UNIVERSITY OF CALIFORNIA

Los Angeles

On Recursive Least-Squares Filtering Algorithms and Implementations

A dissertation submitted in partial satisfaction of the

requirements for the degree

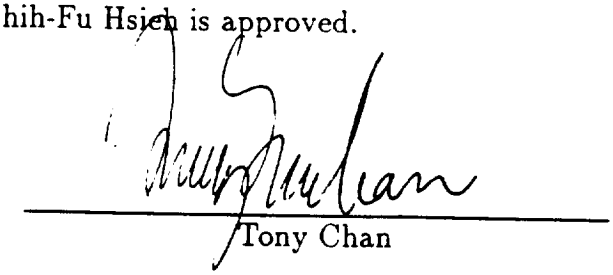
Doctor of Philosophy in Electrical Engineering

by

Shih-Fu Hsieh

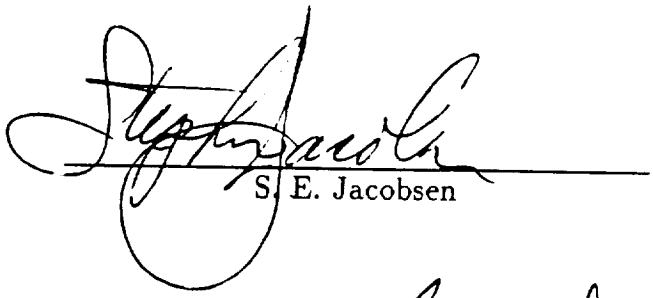
1990

The dissertation of Shih-Fu Hsieh is approved.

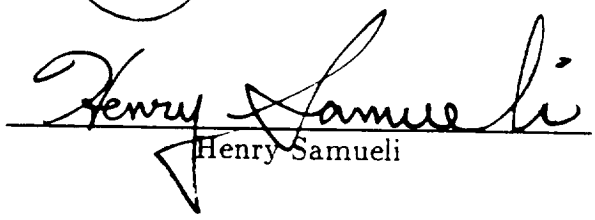


Tony Chan

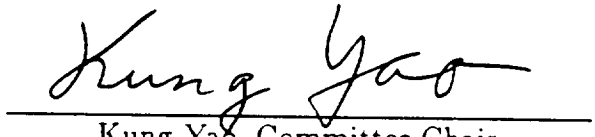
J. S. Gibson



S. E. Jacobsen



Henry Samueli



Kung Yao, Committee Chair

University of California, Los Angeles

1990

DEDICATION

To my Father and Mother

Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

I want to thank Professor Kung Yao for his constant support, and careful and kind guidance on my research. I also want to thank Professors Ezio Biglieri, Tony Chan, J. S. Gibson, S. E. Jacobsen, and Henry Samueli for serving on my committee.

I am very grateful to Dr. K.J.R. Liu for all of his incessant encouragement and fruitful discussions. I especially appreciate Ron and Kay Rogowski for all the pleasant time I shared with them.

This work was partially supported by the NASA/Ames under Grant 2-374, the National Science Foundation under Grant NCR-8814407, and the UC MICRO Grant.

VITA

Shih-Fu Hsieh

[redacted]	Born, [redacted]
June 1984	B.S., Department of Electrical Engineering National Taiwan University Taiwan, R.O.C.
1984 – 1986	Ensign Instructor Naval Communications and Electronics School Taiwan, R.O.C.
December 1987	M.S., Department of Electrical Engineering University of California, Los Angeles
December 1989	Engineer, Electrical Engineering Department University of California, Los Angeles
1987 – 1990	Research Assistant, Electrical Engineering Department University of California, Los Angeles
1989 – 1990	Teaching Assistant, Electrical Engineering Department University of California, Los Angeles
1990	Part-time system engineer LinCom Corporation, Los Angeles

PUBLICATIONS

S.F. Hsieh K.J.R. Liu K. Yao	"A fast and effective algorithm for sinusoidal frequency estimation," *IEEE Int'l Symp. on Information Theory,* San Diego, Jan. 14–19, 1990.

S.F Hsieh
K. Yao

"Hyperbolic Gram-Schmidt pseudo-orthogonalization with applications to sliding window RLS filtering," *24-th Annual Conference on Information Science and System,* Princeton University, Mar. 21-3, 1990.

K.J.R. Liu
S.F. Hsieh
K. Yao

"Recursive LS filtering using block Householder transformations," *Proc. of IEEE Int'l Conf. on ASSP,* pp. 1631-1634, Albuquerque, Apr. 3–6, 1990.

S.F. Hsieh
K.J.R. Liu
K. Yao

"Applications of truncated QR methods to sinusoidal frequency estimation," *Proc. of IEEE Int'l Conf. on ASSP,* pp. 2571 - 2574, Albuquerque, Apr. 3–6 1990.

S.F. Hsieh
K. Yao

"Systolic implementation of windowed recursive LS estimation," *Proc. of IEEE Int'l Symp. on CAS,* pp. 1931 - 1934, New Orleans, May 1–3, 1990.

S.F. Hsieh
K.J.R. Liu
K. Yao

"Comparisons of truncated QR and SVD methods for AR spectral estimations," *Proc. of 2nd Int'l Workshop on SVD and Signal Processing,* pp. 182–188, University of Rhode Island, June 25–27, 1990.

K.J.R. Liu
S.F. Hsieh
K. Yao

"Two-level pipelined implementation of systolic block Householder transformation with application to RLS algorithm," *Int'l Conf. on Application Specific Array Processors,* Princeton, Sept. 5–7, 1990.

K.J.R. Liu
S.F. Hsieh
K. Yao

"Performance comparisons of parallel SVD in VLSI array processors," *IEEE Workshop on VLSI Signal Processing,* San Diego, Nov. 7–9, 1990.

ABSTRACT OF THE DISSERTATION

On Recursive Least-Squares Filtering Algorithms and Implementations

by

Shih-Fu Hsieh

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, August 1990

Professor Kung Yao, Chair

For many real-time signal processing applications, fast and numerically stable algorithms for solving least-squares problems are necessary and important. In particular, under *non-stationary* conditions, these algorithms must be able to adapt themselves to reflect the changes in the system and take appropriate adjustments to achieve optimum performances. Among existing algorithms, the QR-decomposition (QRD)-based *recursive least-squares* (RLS) methods have been shown to be useful and effective for adaptive signal processing in modern communications, radar, and sonar systems.

In order to increase the speed of processing and achieve high throughput rate, many algorithms are being *vectorized* and/or pipelined to facilitate high degrees of parallelism. A time-recursive formulation of RLS filtering employing *block* QRD will be considered first. Several methods including a new non-continuous

windowing scheme based on selectively rejecting contaminated data have been investigated for adaptive processing.

Based on systolic triarrays initiated by Gentleman/Kung and McWhirter, many other forms of systolic arrays are shown to be capable of implementing different algorithms. Various updating and downdating systolic algorithms and architectures for RLS filtering are examined and compared in details, which include Householder reflector, Gram-Schmidt procedure, and Givens rotation. A unified approach encompassing existing square-root-free algorithms is also proposed.

For the sinusoidal spectrum estimation problem, a judicious method of separating the noise from the signal is of great interest. Various truncated QR methods are proposed for this purpose and compared to the truncated SVD method. Computer simulations provided for detailed comparisons show the effectiveness of these methods.

This thesis deals with fundamental issues of numerical stability, computational efficiency, adaptivity, and VLSI implementation for the RLS filtering problems. In all, various new and modified algorithms and architectures are proposed and analyzed; the significance of any of the new method depends crucially on specific application.

Chapter 1

Introduction

For many real-time signal processing applications, such as system identification, channel equalization, adaptive antenna arrays, spectrum estimation, etc., fast and numerically stable algorithms for solving least-squares problems are necessary and important. In particular, under *non-stationary* conditions, these algorithms must be able to adapt themselves to reflect the changes in the system and take appropriate adjustments to achieve the best performance. Among existing algorithms, the QR-decomposition (QRD)-based *recursive least-squares* (RLS) methods have been proven to be very useful and effective towards *adaptive signal processing* in modern communications, radar, and sonar systems [7, 9, 15, 16, 22, 23, 25, 27, 28, 31, 30, 36, 37, 38, 39, 40, 41, 42, 43, 48, 47, 51, 54, 53, 55, 56, 57, 61, 65, 64].

A time-recursive formulation of RLS filtering employing *block* QRD will be given first. To facilitate nonstationary adaptive processing, exponentially growing, fixed-size sliding, and a new non-continuous windowing scheme based on selectively rejecting contaminated equations are introduced and compared.

Various updating and downdating algorithms for RLS filtering are examined

and compared, which include Householder reflector, Gram-Schmidt procedure, and Givens rotation. When the block size of data is reduced to one, these algorithms reduce to existing scalar-type processing methods. A unified approach encompassing existing square-root-free algorithms is proposed.

Systolic arrays promise to be some of the most suitable VLSI architectures for implementing real-time adaptive signal processing algorithms. Based on systolic triarrays initiated by Gentleman/Kung [19] and McWhirter [41], many other forms of systolic arrays are shown to be capable of implementing different algorithms.

For some applications, like sinusoidal spectrum estimation, a judicious method of separating the noise from the signal is always of great interest. This involves rank determination and truncation in the linear system modeling of the problem. Some QR-based methods are proposed for this purpose and compared to other truncation methods. Computer simulations are provided for detailed comparisons.

In this thesis, numerical stability, computational efficiency, suitability for VLSI implementation, and adaptivity are the major considerations for the RLS filtering problems. In all, different methods are provided and analyzed whereupon the choice rests on the application itself.

This thesis will focus on efficient and suitable VLSI algorithms for RLS filtering with applications to real-time signal processing problems. Attention will be focused on *recursive vectorized block* data processing VLSI structure, which generalizes current scalar-type processing structures and possibly reduces the I/O cost. For many architectures, computational speed is much faster compared to data movement; this is the reason why many algorithms are being reformulated into

Table 1.1: Summary of notations.

Symbol	Notation
n	time index
p	order of LS filtering (no. of columns/sensors)
k	block size
X_i	i-th snapshot of matrix data block
$\mathbf{y}_i$	i-th snapshot of desired (R.H.S.) data block
$Q(n)$	orthogonal range space of $X(n)$
$Q^{\perp}(n)$	orthogonal null space of $X(n)$
$\mathbf{w}(n)$	opt. weight vector $\in \mathcal{R}^p$ at time n
$\mathbf{e}(n)$	opt. overall residual vectors, $X(n)\mathbf{w}(n) - \mathbf{y}(n) \in \mathcal{R}^{nk}$, till time n
$\mathbf{e}_n$	opt. residual vector, $X_n\mathbf{w}(n) - \mathbf{y}_n \in \mathcal{R}^k$, at time n
$R(n)$	triangular matrix $\in \mathcal{R}^{p\times p}$ obtained from QRD of $X(n)$
$\mathbf{u}(n)$	projection of $\mathbf{y}(n)$ onto $Q(n)$
$\mathbf{v}(n)$	residual vector by projecting $\mathbf{y}(n)$ onto $Q^{\perp}(n)$
ℓ	fixed-window size (no. of blocks)
λ	block forgetting factor
Λ	block row-weighting matrix
Z	previously triangularized matrix with appended new data block
c, s	cosine and sine values of the planar rotation angle
$\hat{c}, \hat{s}$	hyper-cosine and sine values of the hyperbolic rotation angle
k_1, k_2	normalization factors of row data for sqrt-free algorithms
μ, ν	parameters of different sqrt-free algorithms

vectorized algorithms. Block processing also offers the advantages of reducing arithmetical operations and possibly minimizing the round-off errors occurring in the intermediate stage because of the enlarged wordlength in the internal registers. Table 1.1 provides a list of major symbols and notations used in this thesis.

Finally, future work and a summary of previous and new results will be given.

Chapter 2

Mathematical Formulation of Time Recursive Least-Squares Filtering Problems

For many signal processing applications, time recursive updating forms an important and natural approach to tackle such problems. Consider the adaptive antenna array as shown in Fig. 2.1 [25, 16]. At each time snapshot, a $k \times p$-dimensional data X is collected by the auxiliary antennas while the main antenna receives the data $\mathbf{y}$. Our goal is to find a set of weighting coefficients w_j, $j = 1, \cdots, p$, and possibly its associated residual, such that the Euclidean norm of the overall residual up to time n, $\sqrt{\sum_{i=1}^{n} \|X_i \cdot \mathbf{w} - \mathbf{y}_i\|^2}$ is minimized. If $k = 1$, Gentleman and Kung [19] in 1981 proposed a systolic array to update the optimum weight $\mathbf{w}(n)$ as time n advances. by successively updating the upper-triangular matrix of the QR decomposition of the augmented matrix of $X(n)$ and $\mathbf{y}(n)$. McWhirter [41] in 1983 extended this structure by computing the

most recent scalar residual at each time snapshot without explicitly computing the optimum weight vector $\mathbf{w}(n)$ which entails back substitution.

2.1 Time-Recursive Block QR Algorithms for LS Problems

In this section, recurrence formula to update the optimum weighting and residual vectors in a *block* manner as a function of time are derived. Consider a *time-recursive* least-squares (LS) problem:

$$X(n)\mathbf{w}(n) \approx \mathbf{y}(n)\,, \tag{2.1}$$

where $X(n)$ and $\mathbf{y}(n)$ have growing dimensions in the number of data blocks in rows (growing-window),

$$X(n) = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} \in \mathcal{R}^{nk\times p}, \qquad \mathbf{y}(n) = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{bmatrix} \in \mathcal{R}^{nk}, \tag{2.2}$$

with $X_i \in \mathcal{R}^{k\times p}$, $i = 1, 2, \cdots, n$ and $\mathbf{y}_i \in \mathcal{R}^k$, $i = 1, 2, \cdots, n$. Here we denote k and n as the *block size* and the time index respectively, and p is the order of the LS problem. Capitalized letters (e.g., X) are used to denote matrices; small letters in boldface (e.g., $\mathbf{y}$) vectors, and small letters (e.g., w_i) scalars. A time index n is represented in the parenthesis, e.g., $X(n)$, to denote all of the time span until n, or in the subscript, e.g., X_i, the time epoch n only. For simplicity of notations, we also choose our data as real-valued. It is very easy to extend to the complex-valued cases. The previously considered scalared cases in [19, 41] then have $k = 1$.

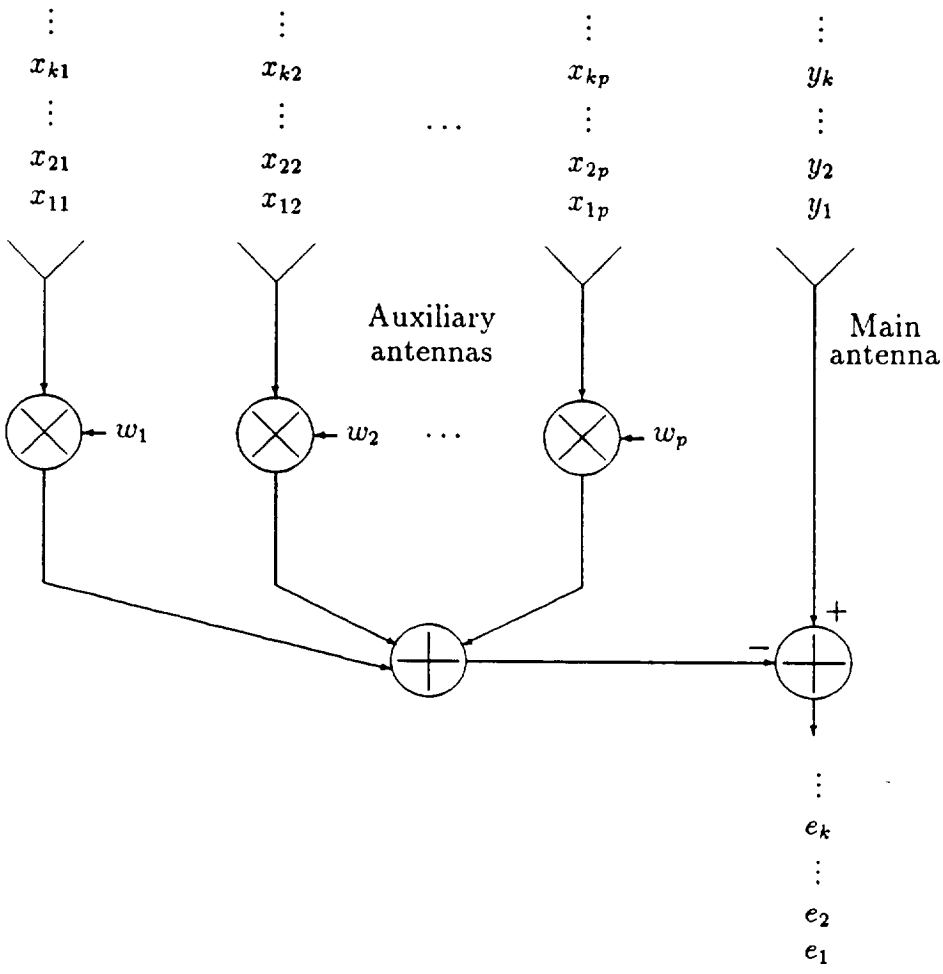


Figure 2.1: Adaptive antenna array

The LS solution $\mathbf{w}(n) \in \mathcal{R}^p$ is computed such that the Euclidean norm of the *residual* vector

$$\mathbf{e}(n) = \begin{bmatrix} \mathbf{e}_1(n) \\ \mathbf{e}_2(n) \\ \vdots \\ \mathbf{e}_n(n) \end{bmatrix} = X(n)\mathbf{w}(n) - \mathbf{y}(n) \in \mathcal{R}^{nk} \tag{2.3}$$

is minimized. All the norms $\|\cdot\|$ of vectors or matrices mentioned are 2-norm, unless otherwise specified. Our interest is to find the recurrence formula for $\mathbf{w}(n)$ and $\mathbf{e}(n)$ as n increases.

Suppose the QR decomposition of the *augmented* matrix $[\ X(n)\ \ \mathbf{y}(n)\ ]$ is known at time n,

$$[\ X(n)\ \ \mathbf{y}(n)\ ] = [\ Q(n)\ \ Q^{\perp}(n)\ ] \begin{bmatrix} R(n) & \mathbf{u}(n) \\ 0 & \mathbf{v}(n) \end{bmatrix}, \tag{2.4}$$

where $Q(n) \in \mathcal{R}^{nk \times p}$ and $Q^{\perp}(n) \in \mathcal{R}^{nk \times (nk-p)}$ represent the orthogonal range and null spaces of the data matrix $X(n)$, and $\mathbf{u}(n) \in \mathcal{R}^p$ is the projection of $\mathbf{y}(n)$ onto $Q(n)$, $\mathbf{v}(n) \in \mathcal{R}^{nk-p}$ is its counterpart projected onto $Q^{\perp}(n)$, and $R(n) \in \mathcal{R}^{p \times p}$ is an upper-triangular matrix and assumed to be full-rank. $R(n)$ is sometimes called the *Cholesky factor* of the covariance matrix of $X(n)$ in that the Cholesky factorization of $X^T(n)X(n)$ can be uniquely expressed as $R^T(n)R(n)$ subject to the signs in each rows of $R(n)$ as long as $X(n)$ has full column rank.

Because an orthogonal transformation preserves the Euclidean norms of of a vector, it can be shown that[35]

$$\|\mathbf{e}(n)\| = \|X(n)\mathbf{w}(n) - \mathbf{y}(n)\| \tag{2.5}$$

$$= \left\| [\ Q(n)\ \ Q^{\perp}(n)\ ] \begin{bmatrix} R(n) & \mathbf{u}(n) \\ 0 & \mathbf{v}(n) \end{bmatrix} \begin{bmatrix} \mathbf{w}(n) \\ -1 \end{bmatrix} \right\| \tag{2.6}$$

$$= \left\| [\, Q(n) \;\; Q^{\perp}(n) \,] \begin{bmatrix} R(n)\mathbf{w}(n) - \mathbf{u}(n) \\ -\mathbf{v}(n) \end{bmatrix} \right\| \tag{2.7}$$

$$= \|Q(n)[R(n)\mathbf{w}(n) - \mathbf{u}(n)]\| + \| - Q^{\perp}(n)\mathbf{v}(n)\| \tag{2.8}$$

$$= \| - Q^{\perp}(n)\mathbf{v}(n)\| \tag{2.9}$$

as long as

$$R(n)\mathbf{w}(n) = \mathbf{u}(n) \; . \tag{2.10}$$

(2.8) follows from the fact that the Euclidean norm of the sum of two or more orthogonal vectors is equal to the sum of Euclidean norms of these vectors. (2.9) means that the residual vector while estimating $\mathbf{y}(n)$ from $X(n)$ must lie in the null space of $X(n)$ which corresponds well with the geometrical interpretation of the orthogonal principle of LS problems.

As the time index n advances by one, i.e., a new data block of size k, $[\; X_{n+1} \;\; \mathbf{y}_{n+1} \;]$, is acquired, we can write the recurrence formula for QRD as follows:

$$[\; X(n+1) \;\; \mathbf{y}(n+1) \;] = \begin{bmatrix} X(n) & \mathbf{y}(n) \\ X_{n+1} & \mathbf{y}_{n+1} \end{bmatrix} \tag{2.11}$$

$$= \begin{bmatrix} Q(n) & Q^{\perp}(n) & 0 \\ 0 & 0 & I_k \end{bmatrix} \begin{bmatrix} R(n) & \mathbf{u}(n) \\ 0 & \mathbf{v}(n) \\ X_{n+1} & \mathbf{y}_{n+1} \end{bmatrix} \tag{2.12}$$

$$= \begin{bmatrix} Q(n) & Q^{\perp}(n) & 0 \\ 0 & 0 & I_k \end{bmatrix} \begin{bmatrix} Q_{n+1} & & Q^{\perp}_{n+1} \\ & I_{nk-p} & \\ \widehat{Q}_{n+1} & & \widehat{Q}^{\perp}_{n+1} \end{bmatrix} \begin{bmatrix} R(n+1) & \mathbf{u}(n+1) \\ 0 & \mathbf{v}(n) \\ 0 & \mathbf{v}_{n+1} \end{bmatrix} \tag{2.13}$$

$$= \begin{bmatrix} Q(n)Q_{n+1} & Q^{\perp}(n) & Q(n)Q^{\perp}_{n+1} \\ \hat{Q}_{n+1} & 0 & \hat{Q}^{\perp}_{n+1} \end{bmatrix} \begin{bmatrix} R(n+1) & \mathbf{u}(n+1) \\ 0 & \mathbf{v}(n) \\ 0 & \mathbf{v}_{n+1} \end{bmatrix} \quad (2.14)$$

$$= [\, Q(n+1) \;\; Q^{\perp}(n+1) \,] \begin{bmatrix} R(n+1) & \mathbf{u}(n+1) \\ 0 & \mathbf{v}(n+1) \end{bmatrix}. \quad (2.15)$$

By defining

$$\tilde{Q}(n+1) \equiv \begin{bmatrix} Q_{n+1} & Q^{\perp}_{n+1} \\ \hat{Q}_{n+1} & \hat{Q}^{\perp}_{n+1} \end{bmatrix} \in \mathcal{R}^{(p+k)\times(p+k)}, \quad (2.16)$$

we note that $\tilde{Q}(n+1)$ constitutes an orthogonal transformation to annihilate the newly appended data block X_{n+1}, and $Q_{n+1} \in \mathcal{R}^{p\times p}$ and $\hat{Q}_{n+1} \in \mathcal{R}^{k\times p}$ represent the operation of modifying the *range* space while $Q^{\perp}_{n+1} \in \mathcal{R}^{p\times k}$, and $\hat{Q}^{\perp}_{n+1} \in \mathcal{R}_{k\times k}$ that of the *null* space. We use a *hat* $\hat{}$ to denote the new dimensional growth due to the appended data. To sum up, we have the following recurrence formula:

$$Q(n+1) = \begin{bmatrix} Q(n)Q_{n+1} \\ \hat{Q}_{n+1} \end{bmatrix} \in \mathcal{R}^{(n+1)k\times p}, \quad (2.17)$$

$$Q^{\perp}(n+1) = \begin{bmatrix} Q^{\perp}(n) & Q(n)Q^{\perp}_{n+1} \\ 0 & \hat{Q}^{\perp}_{n+1} \end{bmatrix} \in \mathcal{R}^{(n+1)k \,\times\, (n+1)k-p}, \quad (2.18)$$

$$\mathbf{v}(n+1) = \begin{bmatrix} \mathbf{v}(n) \\ \mathbf{v}_{n+1} \end{bmatrix} \in \mathcal{R}^{(n+1)k-p}, \quad (2.19)$$

$$\begin{bmatrix} R(n+1) & \mathbf{u}(n+1) \\ 0 & \mathbf{v}_{n+1} \end{bmatrix} = \tilde{Q}(n+1) \begin{bmatrix} R(n) & \mathbf{u}(n) \\ X_{n+1} & \mathbf{y}_{n+1} \end{bmatrix}$$

$$= \begin{bmatrix} Q_{n+1} & Q^{\perp}_{n+1} \\ \hat{Q}_{n+1} & \hat{Q}^{\perp}_{n+1} \end{bmatrix} \begin{bmatrix} R(n) & \mathbf{u}(n) \\ X_{n+1} & \mathbf{y}_{n+1} \end{bmatrix}. \quad (2.20)$$

The desired optimum weighting vector $\mathbf{w}(n+1)$ and the residual vector $\mathbf{e}(n+1)$ are thus given by

$$R(n+1)\mathbf{w}(n+1) = \mathbf{u}(n+1), \tag{2.21}$$

which can be solved by back substitution, and

$$\begin{aligned}
\mathbf{e}(n+1) &= -Q^{\perp}(n+1)\mathbf{v}_{n+1} \qquad \text{(see (2.9))} && (2.22)\\
&= -\begin{bmatrix} Q^{\perp}(n) & Q(n)Q_{n+1}^{\perp} \\ 0 & \hat{Q}_{n+1}^{\perp} \end{bmatrix} \begin{bmatrix} \mathbf{v}(n) \\ \mathbf{v}_{n+1} \end{bmatrix} && (2.23)\\
&= \begin{bmatrix} -Q^{\perp}(n)\mathbf{v}(n) - Q(n)Q_{n+1}^{\perp}\mathbf{v}_{n+1} \\ -\hat{Q}_{n+1}^{\perp}\mathbf{v}_{n+1} \end{bmatrix} && (2.24)\\
&= \begin{bmatrix} \mathbf{e}(n) - Q(n)Q_{n+1}^{\perp}\mathbf{v}_{n+1} \\ -\hat{Q}_{n+1}^{\perp}\mathbf{v}_{n+1} \end{bmatrix} \in \mathcal{R}^{(n+1)k}. && (2.25)
\end{aligned}$$

To see the changes of residuals in each previous data blocks due to a new observation of X_{n+1} and $\mathbf{y}_{n+1}$, we can write down the following lemma.

Lemma 1 *(updating residual)*

$$\mathbf{e}(n+1) = \begin{bmatrix} \mathbf{e}_1(n+1) \\ \mathbf{e}_2(n+1) \\ \vdots \\ \mathbf{e}_n(n+1) \\ \mathbf{e}_{n+1}(n+1) \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1(n) - \hat{Q}_1 Q_2 \cdots Q_n Q_{n+1}^{\perp}\mathbf{v}_{n+1} \\ \mathbf{e}_2(n) - \hat{Q}_2 Q_3 \cdots Q_n Q_{n+1}^{\perp}\mathbf{v}_{n+1} \\ \vdots \\ \mathbf{e}_n(n) - \hat{Q}_n Q_{n+1}^{\perp}\mathbf{v}_{n+1} \\ -\hat{Q}_{n+1}^{\perp}\mathbf{v}_{n+1} \end{bmatrix} \in \mathcal{R}^{(n+1)k} \tag{2.26}$$

Proof.(2.26) can be derived from (2.17) and by noting that $Q(1) = \hat{Q}_1$, *i.e.,*

$$Q(2) = \begin{bmatrix} \hat{Q}_1 Q_2 \\ \hat{Q}_2 \end{bmatrix}$$

$$
\begin{aligned}
Q(3) &= \begin{bmatrix} Q(2)Q_3 \\ \hat{Q}_3 \end{bmatrix} = \begin{bmatrix} \hat{Q}_1 Q_2 Q_3 \\ \hat{Q}_2 Q_3 \\ \hat{Q}_3 \end{bmatrix} \\
&\vdots \\
Q(n) &= \begin{bmatrix} \hat{Q}_1 Q_2 \cdots Q_n \\ \hat{Q}_2 Q_3 \cdots Q_n \\ \vdots \\ \hat{Q}_n \end{bmatrix}
\end{aligned} \tag{2.27}
$$

and substituting $Q(n)$ back into (2.25). ∎

(2.26) explains that the overall residual vector at time $n+1$ comprises of two parts: one of them is equal to $-\hat{Q}_{n+1}^{\perp}\mathbf{v}_{n+1}$, the new dimensional growth due to X_{n+1}, while the other one is equal to the old residual vector at the previous time n, $\mathbf{e}(n)$, offset by $Q(n)Q_{n+1}^{\perp}\mathbf{v}_{n+1}$. Therefore, if we are only interested in $R(n+1)$ and/or $\mathbf{e}_{n+1}$, then we can simply maintain the information of $R(n)$ and $\mathbf{u}(n)$, which is usually the case for many applications such as beamforming[41, 61]. However, if we need to monitor all of those previously block residual vectors $\mathbf{e}_i$, $i = 1, \cdots, n$, then the previously computed range space $Q(n)$ is still required to update those old residual vectors. This monitoring may aid in the determination of some *spurious* observations(rows) such that they can be deleted (downdated) from the LS estimation problem and mitigate the possible bias caused by them. For linear regression [18, 35], this diagnosis in monitoring all the residuals is especially very important. Our method, following the approach first proposed by McWhirter [41], provides a *one-pass* direct way of keeping track of all of the residuals, without explicitly computing $\mathbf{w}(n)$ followed by $X(n)\mathbf{w}(n) - \mathbf{y}(n)$ which requires *two-passes* (involving the use of back substitution twice) and can

be objectionable from the throughput point of view. We will elaborate on this later in Section 2.6.

2.2 Pseudo Cholesky Decomposition

In this section, pseudo orthogonalization will be defined and will be required in explaining the downdating operations. Let Z be an $(m+n) \times p$ matrix cascaded from two real-valued data matrices $A \in \mathcal{R}^{m \times p}$ and $B \in \mathcal{R}^{n \times p}$, i .e.,

$$Z = \begin{bmatrix} A \\ B \end{bmatrix}. \tag{2.28}$$

A $J_{m/n}$-*pseudo sample covariance matrix* of Z is defined as $Z^T J_{m/n} Z$, and its corresponding $J_{m/n}$-*pseudo Cholesky decomposition* is defined as

$$R^T R = Z^T J_{m/n} Z = A^T A - B^T B, \tag{2.29}$$

where

$$J_{m/n} = \begin{bmatrix} I_m & \\ & -I_n \end{bmatrix} \in \mathcal{R}^{(m+n)\times(m+n)} \tag{2.30}$$

is called a *pseudo identity* or *signature* matrix, and the $p \times p$ upper-triangular matrix R (assuming that R exists and has full rank) is called the $J_{m/n}$-*pseudo Cholesky factor* of the pseudo-covariance matrix of Z. For convenience, from now on we will suppress the subscript $(\cdot)_{m/n}$ in J unless it is necessary.

2.2.1 Notations

A J-*pseudo inner product* is defined as

$$< \mathbf{u}, \mathbf{v} >_J = \mathbf{u}^T J \mathbf{v} = < \mathbf{v}, \mathbf{u} >_J, \quad \forall\, \mathbf{u},\ \mathbf{v} \in \mathcal{R}^{m+n}, \tag{2.31}$$

and a *J-pseudo vector norm* is defined as

$$\|\mathbf{u}\|_J = \sqrt{\mathbf{u}^T J \mathbf{u}}, \quad \forall\, \mathbf{u} \in \mathcal{R}^{m+n} \text{ such that } \mathbf{u}^T J \mathbf{u} \geq 0. \tag{2.32}$$

An $(m+n) \times (m+n)$ matrix H is said to be *J-pseudo orthogonal* if $H^T J H = J$, namely,

$$\begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \vdots \\ \mathbf{h}_{m+n}^T \end{bmatrix} J [\, \mathbf{h}_1 \; \mathbf{h}_2 \; \cdots \; \mathbf{h}_{m+n} \,] = \tag{2.33}$$

$$\begin{bmatrix} \|\mathbf{h}_1\|_J^2 & < \mathbf{h}_1, \mathbf{h}_2 >_J & \cdots & < \mathbf{h}_1, \mathbf{h}_{m+n} >_J \\ < \mathbf{h}_1, \mathbf{h}_2 >_J & \|\mathbf{h}_2\|_J^2 & \cdots & < \mathbf{h}_2, \mathbf{h}_{m+n} >_J \\ \vdots & \vdots & \ddots & \vdots \\ < \mathbf{h}_1, \mathbf{h}_{m+n} >_J & < \mathbf{h}_2, \mathbf{h}_{m+n} >_J & \cdots & \|\mathbf{h}_{m+n}\|_J^2 \end{bmatrix} = J.$$

Equivalently, we can say that H has J-pseudo orthogonal columns. This means that for any two columns of H, their J-inner product satisfies

$$< \mathbf{h}_i, \mathbf{h}_j >_J = \begin{cases} 1 & \text{, if } 1 \leq i = j \leq m. \\ -1 & \text{, if } m < i = j \leq m+n. \\ 0 & \text{, otherwise.} \end{cases} \tag{2.34}$$

2.3 Simultaneously Up/Down-dating RLS Problems

Modifications of matrix factorizations have been of great interest in many applications [6, 45, 35]. In particular, recursively up/down-dating QR decomposition

by adding some new and deleting some old data rows will be examined, which will then lead to the systolic implementations in the following chapters.

A time-recursive up/down-dating RLS problem amounts to find $[\ R(n+1), \mathbf{u}(n+1)\ ]$, $\mathbf{v}(n+1)$ and/or $\mathbf{e}(n+1)$ from the knowledge of $[\ R(n), \mathbf{u}(n)\ ]$, $\mathbf{v}(n), \mathbf{e}(n)$, the new data block $[\ X_{n+1}, \mathbf{y}_{n+1}\ ]$, and the old data block $[\ X_{n-\ell+1}, \mathbf{y}_{n-\ell+1}\ ]$, i.e.,

$$\begin{bmatrix} R(n) & \mathbf{u}(n) \\ 0 & \mathbf{v}(n) \\ \boxed{X_{n+1}} & \boxed{\mathbf{y}_{n+1}} \\ \boxed{X_{n-\ell+1}} & \boxed{\mathbf{y}_{n-\ell+1}} \end{bmatrix} \overset{Up/Downdating}{\Longrightarrow} \begin{bmatrix} R(n+1) & \mathbf{u}(n+1) \\ 0 & \mathbf{v}(n) \\ 0 & \overline{\mathbf{v}}_{n+1} \\ 0 & \underline{\mathbf{v}}_{n-\ell+1} \end{bmatrix} \tag{2.35}$$

or symbolically

$$\begin{bmatrix} \times & \times & \times & \cdots & \times & \times \\ & \times & \times & \cdots & \times & \times \\ & & \times & \cdots & \times & \times \\ & & & \ddots & \vdots & \vdots \\ & & & & \times & \times \\ & & & & & \vdots \\ + & + & + & \cdots & + & + \\ - & - & - & \cdots & - & - \end{bmatrix} \overset{Up/Downdating}{\Longrightarrow} \begin{bmatrix} \otimes & \otimes & \otimes & \cdots & \otimes & \otimes \\ & \otimes & \otimes & \cdots & \otimes & \otimes \\ & & \otimes & \cdots & \otimes & \otimes \\ & & & \ddots & \vdots & \vdots \\ & & & & \otimes & \otimes \\ & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \oplus \\ 0 & 0 & 0 & \cdots & 0 & \ominus \end{bmatrix}. \tag{2.36}$$

Following (2.4), then we have

$$\begin{bmatrix} X(n) & \mathbf{y}(n) \\ X_{n+1} & \mathbf{y}_{n+1} \\ X_{n-\ell+1} & \mathbf{y}_{n-\ell+1} \end{bmatrix} = \begin{bmatrix} Q(n) & Q^{\perp}(n) & & \\ & & I_k & \\ & & & I_k \end{bmatrix} \begin{bmatrix} R(n) & \mathbf{y}(n) \\ 0 & \mathbf{v}(n) \\ X_{n+1} & \mathbf{y}_{n+1} \\ X_{n-\ell+1} & \mathbf{y}_{n-\ell+1} \end{bmatrix} \tag{2.37}$$

$$= \begin{bmatrix} Q(n) & Q^{\perp}(n) & & \\ & & I_k & \\ & & & I_k \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ & I & \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} R(n+1) & \mathbf{u}(n+1) \\ 0 & \mathbf{v}(n) \\ 0 & \overline{\mathbf{v}}_{n+1} \\ 0 & \underline{\mathbf{v}}_{n-\ell+1} \end{bmatrix}, \tag{2.38}$$

and the extended up/down-dated residual vector is given by

$$\underline{\overline{\mathbf{e}}}(n+1) = \begin{bmatrix} X(n) & \mathbf{y}(n) \\ X_{n+1} & \mathbf{y}_{n+1} \\ X_{n-\ell+1} & \mathbf{y}_{n-\ell+1} \end{bmatrix} \begin{bmatrix} \mathbf{w}(n+1) \\ -1 \end{bmatrix} \tag{2.39}$$

$$= \begin{bmatrix} -Q^{\perp}(n)\mathbf{v}(n) - Q(n)H_{12}\overline{\mathbf{v}}_{n+1} - Q(n)H_{13}\underline{\mathbf{v}}_{n-\ell+1} \\ -H_{22}\overline{\mathbf{v}}_{n+1} - H_{23}\underline{\mathbf{v}}_{n-\ell+1} \\ -H_{32}\overline{\mathbf{v}}_{n+1} - H_{33}\underline{\mathbf{v}}_{n-\ell+1} \end{bmatrix}. \tag{2.40}$$

2.4 Growing-Window with Exponential Forgetting Factors

If we replace the augmented LS equations in (2.4) by premultiplying a diagonal weighting matrix $\Lambda(n) = diag(\lambda^{n-1}I_k, \cdots, \lambda I_k, I_k)$ to diminish the importance of those previous observations (rows), where $\lambda \in (0, 1]$ is a block forgetting factor, then the exponentially weighted residual in (2.25) will now become

$$\Lambda(n+1)\mathbf{e}(n+1) = \begin{bmatrix} \lambda\mathbf{e}(n) - Q(n)Q_{n+1}^{\perp}\mathbf{v}_{n+1} \\ -\widehat{Q}_{n+1}^{\perp}\mathbf{v}_{n+1} \end{bmatrix}, \tag{2.41}$$

where we can see that the previous residuals are gradually deemphasized by λ. Equivalently, we may consider the weight vector $\mathbf{w}(n)$ is chosen such that the

λ-weighted Euclidean norm of residual vector,

$$\|\mathsf{e}(n)\|_\lambda = \sqrt{\sum_i^n \|\lambda^{n-i}\mathsf{e}_i\|^2} \tag{2.42}$$

is minimized.

2.5 Sliding-Window with Fixed-window Size

A fixed-window or sliding-window RLS filtering needs to incorporate the new data segments (updating) and also remove the influence of the obsolete data (downdating). If we denote ℓ as the number of blocks of the fixed-window size, then for $n \geq \ell$, the fixed-windowed data can be written from the growing-window data given in (2.2) by discarding the oldest $n - \ell$ blocks of data, i.e.,

$$X(n) = \begin{bmatrix} X_{n-\ell+1} \\ \vdots \\ X_n \end{bmatrix} \in \mathcal{R}^{\ell k \times p}, \qquad \mathsf{y}(n) = \begin{bmatrix} \mathsf{y}_{n-\ell+1} \\ \vdots \\ \mathsf{y}_n \end{bmatrix} \in \mathcal{R}^{\ell k}. \tag{2.43}$$

In order to obtain $R(n+1)$ from $R(n)$, we need to *update* (include) X_{n+1} and *downdate* (remove) $X_{n-\ell+1}$ from $R(n)$, i.e,

$$R(n+1)^T R(n+1) = R(n)^T R(n) + X_{n+1}^T X_{n+1} - X_{n-\ell+1}^T X_{n-\ell+1}, \tag{2.44}$$

where we have implicitly noticed that $R(n+1)^T R(n+1) = X(n+1)^T X(n+1)$, $R(n)^T R(n) = X(n)^T X(n)$, and $X(n+1)^T X(n+1) = X(n)^T X(n) + X_{n+1}^T X_{n+1} - X_{n-\ell+1}^T X_{n-\ell+1}$. Therefore, an updating operation in the direct data domain is equivalent to an addition in the second order domain (covariance data), while a downdating operation is equivalent to a subtraction. There are two ways to accomplish this: one is to perform updating and downdating at the same time,

or we can do them one by one consecutively. These will be discussed in details in the following chapters.

Under time-varying conditions, much attention has been focused on schemes employing exponential forgetting factors, while less on fixed-windowed ones. This is partially due to the difficulty of downdating obsolete data encountered in the windowed RLS model. But, fixed-window scheme should not be precluded simply because its computational burden. Other factors, especially fast parameters tracking ability, actually favors this method under some non-stationary conditions. To motivate the need for fixed-window under non-stationary condition, a computer experiment is given to demonstrate the advantage of the faster convergence for the fixed-window method over the method based on an exponential forgetting factor.

2.5.1 An AR model computer simulation

Consider a second order autoregressive (AR) process $\{u(i)\}$ as given in [25, pp. 204-6], where $u(i)+a_1u(i-1)+a_2u(i-2) = v(i), i = 1, \ldots, 100$, with $a_1 = -0.9750$ and $a_2 = 0.9500$ and $u(i) + a_1'u(i-1) + a_2u(i-2) = v(i), i = 101, \ldots, 250$, with $a_1' = -1.5955$. $v(\cdot)$ is a white Gaussian noise with a standard deviation of 0.1, except that from $i = 20$ to 30, $v(i)$'s are intentionally increased by a factor of 20 to account for temporary noisy perturbations. This is equivalent to lowering the SNR by 26 dB during that interval. We note that there is also a step change at iteration 100 in the parameters of the AR model where $a_1 = -0.9750$ jumps to $a_1' = -1.5955$.

To make a fair comparison between the fixed-window scheme with a window size ℓ to the exponentially weighting scheme with a forgetting factor λ in the sense

that both schemes have the same *self noise* (*i.e.*, fluctuation of the estimated parameters with respect to the optimum AR parameters) [40], we choose $\ell = 50$ and $\lambda = \sqrt{(\ell-1)/(\ell+1)}$. 100 simulations with different noise realizations were performed. Fig.2.2 depicts the mean bias versus number of iterations in estimating the AR parameters a_1 and a_1'. For the fixed-window scheme, the transient effect of the changes in the noise (from iteration 20 to 30) is almost completely absent by iteration 80, while its effect on the exponentially weighted window is still present. Even more significant is that for the fixed-window case, after the sudden change in the parameter a_1 to a_1' at iteration 100, convergence is reached with only 50 more iterations, while for the exponentially weighted window case, about 150 iterations are required. This simple example shows that a fixed-window scheme is indeed more suitable for fast parameter tracking under various non-stationary conditions. Since the computational load of a fixed-window scheme is greater(with up and down datings) than that of the exponentially weighted scheme(with only updating), we need to be particularly concerned with its algorithmic and architectural efficiencies. This comparison shows that a fixed-window scheme is indeed necessary and important in speedy tracking parameters under some non-stationary conditions.

Until recently, efficient downdating algorithms have been proposed [2, 48]. But efficient implementations and architectures of fixed-windowed RLS filtering are still rarely considered. In the following chapters, systolic triarrays which are suitable for VLSI design, are proposed to perform fixed-windowed RLS estimation.

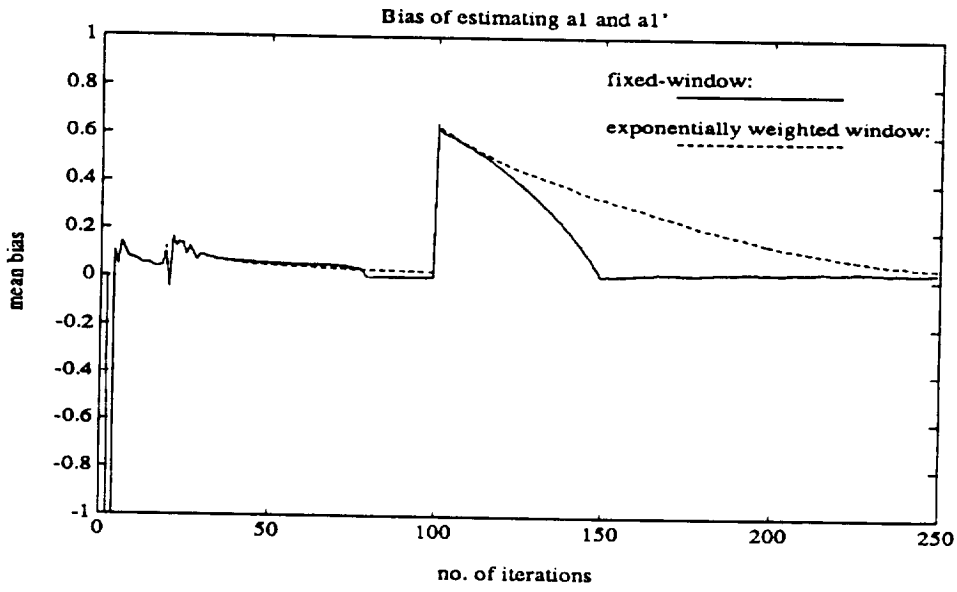


Figure 2.2: Mean bias of estimating non-stationary AR parameter for fixed and exponentially weighted windows.

2.6 Residual-based Selective Window

A new algorithm performing recursive least-squares (RLS) filtering is proposed. Unlike a sliding fixed-window scheme, this new windowing scheme can be *non-*continuous, depending on the estimated level of observation errors(residual). By monitoring the residuals in a recursive manner (see (2.26) in Lemma 1), we can effectively remove those spurious observed data by *downdating* them. This algorithm is most useful when some short-time large interferences perturb the system occasionally. In this respect, it outperforms existing schemes, either exponentially growing or sliding. A computer simulation will be given to justify this.

2.6.1 Robust LS estimations with outlier removal

A conventional LS solution $\mathbf{w}(n) \in \mathcal{R}^p$ for (2.1) is computed such that the Euclidean norm of the residual vector is minimized. Notice that $\mathbf{e}_i(n) \in \mathcal{R}^k$, $1 \leq i \leq n$ denotes the residual of the i-th data block, $X_i\mathbf{w}(n) - \mathbf{y}_i$, when we try to make a best fit from all of X_i, $1 \leq i \leq n$, to $\mathbf{y}_i$, $1 \leq i \leq n$. After obtaining the optimum weight coefficient $\mathbf{w}(n)$ and also the corresponding residual vectors $\mathbf{e}_1(n), \ldots, \mathbf{e}_n(n)$, we can determine an index set $\mathcal{J}$ of outliers based on the criterion, of $\mathcal{J} = \{\, j \,|\, 1 \leq j \leq n, \|\mathbf{e}_j(n)\| > \text{ threshold } \}$ such that those extraneous outliers can be removed. This is the first pass which comprises of: (1). determining the solution of $\mathbf{w}(n)$ which may be obtained from either normal equation, Gaussian elimination or QR decomposition followed by back substitution; (2). finding the associated optimum residual which is obtained by plugging $\mathbf{w}(n)$ into $\mathbf{e}(n) = X(n)\mathbf{w}(n) - \mathbf{y}(n)$. Next, remove all X_j and $\mathbf{y}_j$, $\forall j \in \mathcal{J}$, from (2.1). Then we have

$$X_{\mathcal{J}}(n)\mathbf{w} \approx \mathbf{y}_{\mathcal{J}}(n)\,. \tag{2.45}$$

By using the solution of (2.45), this completes the second pass of a *robust* LS problems.

Figure 2.3 depicts the diagram of the 2-pass robust LS problem with outlier removal[48]. For the two-pass case, a QRD is performed on $[X\ \mathbf{y}]$ followed by back substitution to find the tentative $\mathbf{w}(n)$. After the residual vector is computed, a decision is made to determine which data rows are to be downdated. When all of the outliers are downdated from the Cholesky factor R and the associated $\mathbf{u}$, back substitution is required for the second time to solve for the final optimum $\mathbf{w}(n)$ and the associated optimum residual can be computed.

For the one-pass case, the QRD operator needs not only to orthogonalize the

incoming augmented data matrix, but also to monitor all of the residual vectors, therefore, the computational load is higher in this case. However, by doing so, we have eliminate the need to compute $\mathbf{w}(n)$ by back substitution and the matrix-vector computation of $X(n)\mathbf{w}(n) - \mathbf{y}(n)$. After all of the residual vectors are available, those outliers can be downdated and finally a back substitution is used only one time to compute the optimum $\mathbf{w}(n)$.

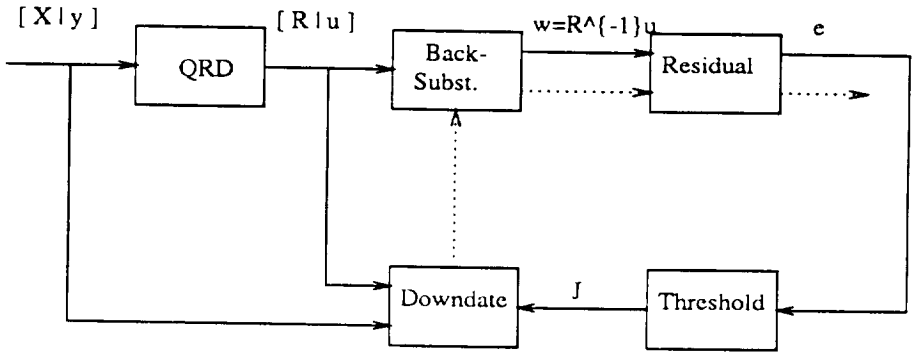
2.6.2 Comparisons of performances of different windows

A second AR model (for details, see the previous simulation case in Section 2.5.1) is used again to demonstrate the advantages of the new window scheme. This time $v(i)$'s are intentionally increased in amplitude by a factor of 30 from $i = 55$ to 57 and also from $i = 155$ to 157 to account for temporary large noisy spikes. This is equivalent to lowering the SNR by about 30 dB during these intervals.

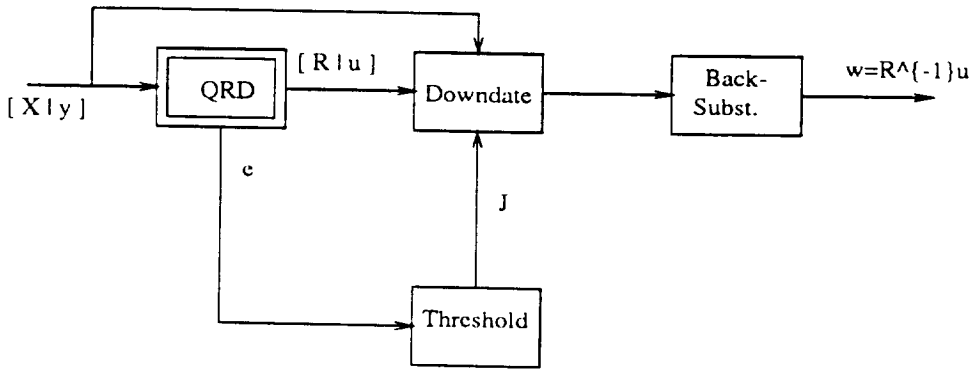
Figs. 2.4 and 2.5 compare the biases of estimating the AR parameters a_1 and a_2. Figs. 2.6 and 2.7 compare the standard deviations of estimating the AR parameters a_1 and a_2. Fig. 2.8 compares the standard deviations of the residuals. Four windowing schemes are compared:

1. no windows are imposed (or equivalently, forgetting factor $\lambda = 1$),
2. exponentially weighted window with forgetting factor $\lambda = \sqrt{49/51}$,
3. fixed-size sliding window with window size $\ell = 50$,
4. selective window with residual threshold = 1.0.

From these figures, we can see that the newly proposed window by selectively rejecting data rows with large residuals gives the least bias in tracking the AR



(A). two-passes



(B). one-pass

Figure 2.3: Two-passes and one-pass diagrams of robust LS estimations with outlier removal

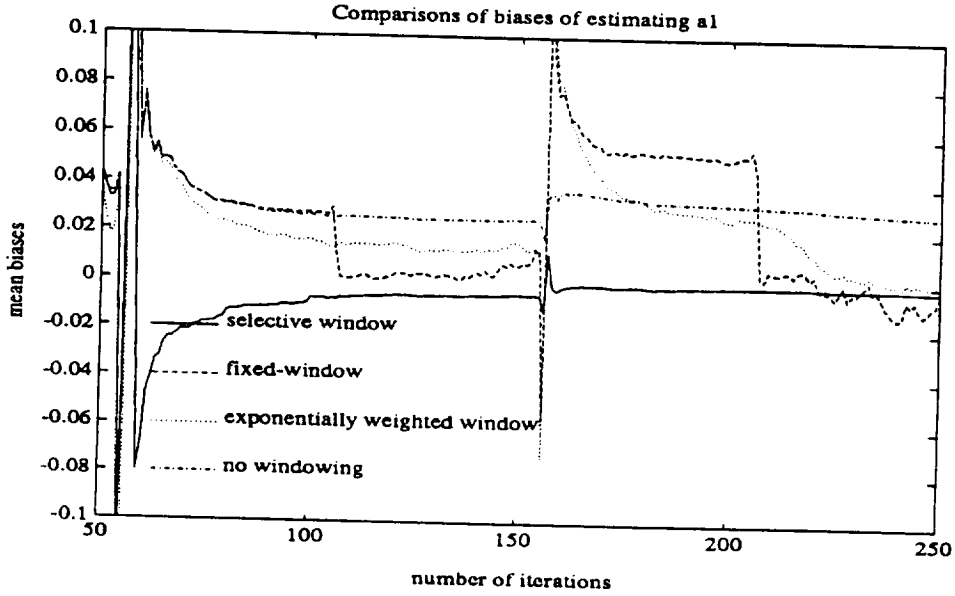


Figure 2.4: Comparisons of mean bias of estimating AR parameter a_1 for various windows under noisy spikes.

parameters and converges most rapidly. This is because this method has discarded those highly perturbed data.

2.7 Other Issues

2.7.1 Mixed-type window schemes

The fixed-window of size ℓ with an exponential block forgetting factor λ has a row-weighting matrix $\Lambda_\ell = (\lambda^{\ell-1}I_k, \cdots, \lambda I_k, I_k)$. We want to modify $\lambda R(n)$ by updating X_{n+1} and downdating $\lambda^\ell X_{n-\ell+1}$, which implies that $R^T(n+1)R(n+1) = \lambda^2 R^T(n)R(n) + X_{n+1}^T X_{n+1} - \lambda^{2\ell} X_{n-\ell+1}^T X_{n-\ell+1}$.

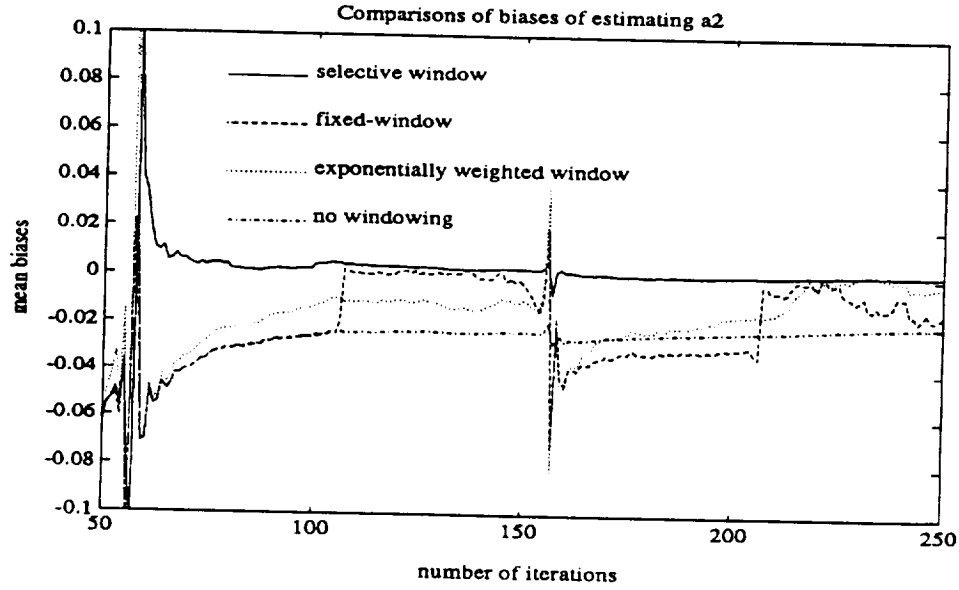


Figure 2.5: Comparisons of mean bias of estimating AR parameter a_2 for various windows under noisy spikes.

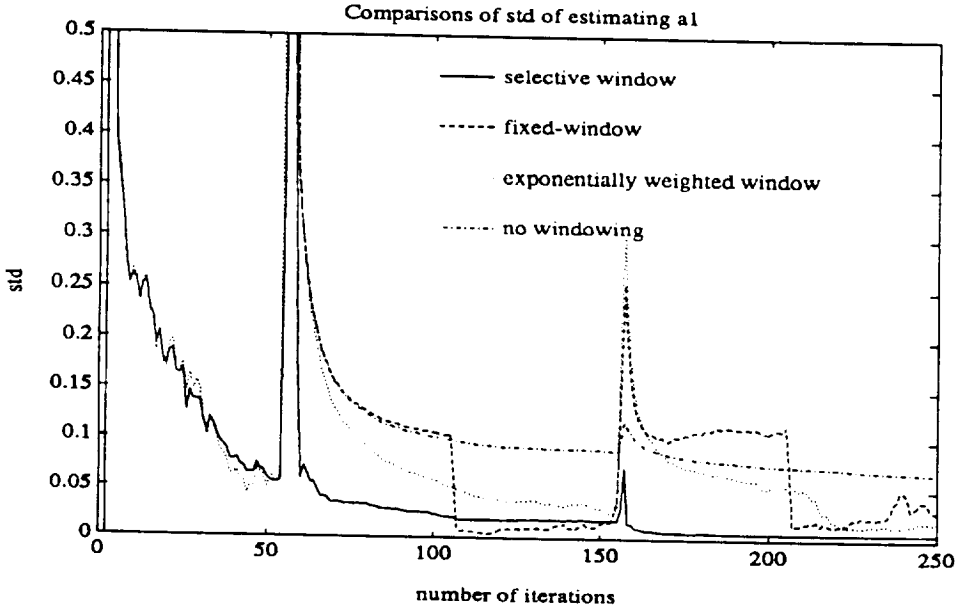


Figure 2.6: Comparisons of standard deviations of estimating AR parameter a_1 for various windows under noisy spikes.

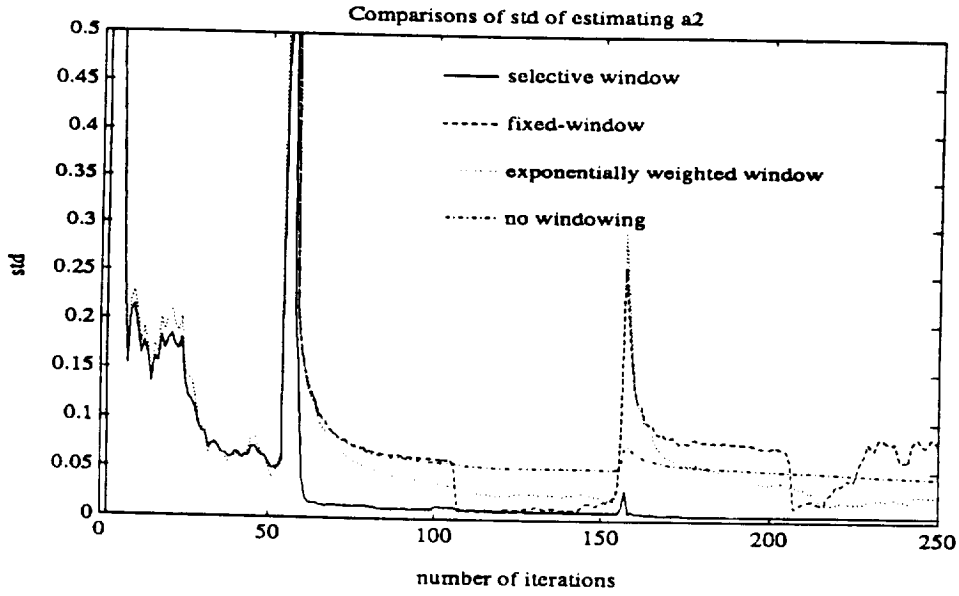


Figure 2.7: Comparisons of standard deviations of estimating AR parameter a_2 for various windows under noisy spikes.

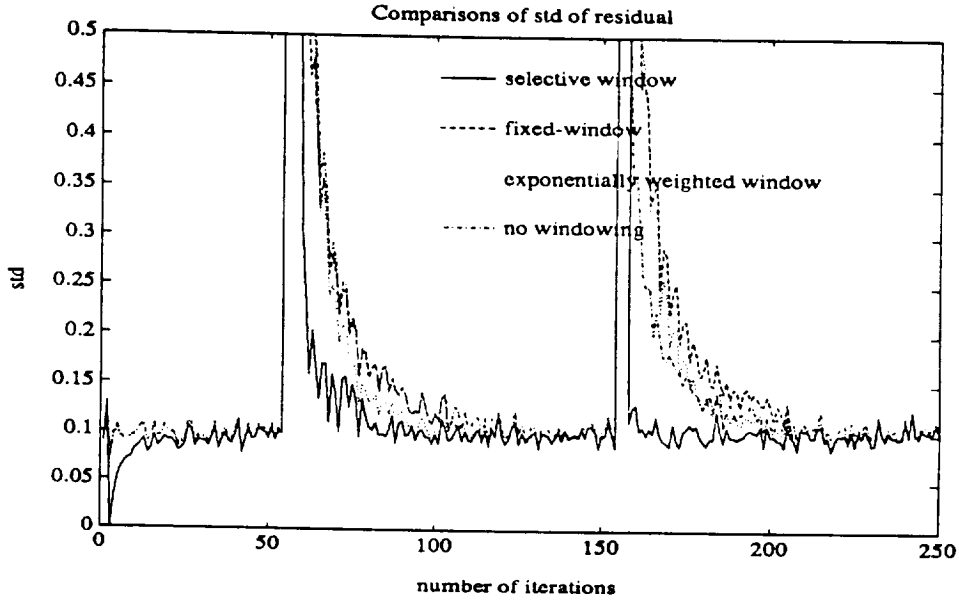


Figure 2.8: Comparisons of standard deviations of residuals of estimating AR parameters for various windows under noisy spikes.

2.7.2 MVDR: least-squares with linear constraints

The above derived algorithms only focus on recursive LS problems with *no constraints*. If a set of *linear constraints* are incorporated, after slight modifications[38], they are still applicable. MVDR beamforming [9, 61, 42, 64] is one of the examples of such applications.

2.8 Conclusions

Up to now, we have assumed that QRD methods are readily available while performing updating/downdating. Explicit algorithms and systolic implementations will be given in the following chapters, which include Householder transformation, (modified) Gram-Schmidt orthogonalization procedure, and Givens rotation methods.

Chapter 3

Block Gram-Schmidt Pseudo-orthogonalization

3.1 Pseudo Orthogonalization Algorithms

We denote a *J-pseudo orthogonal decomposition* of Z as

$$Z = [\ H \quad H^{\perp}\ ] \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{3.1}$$

where $H \in R^{(m+n)\times p}$ and $H^{\perp} \in R^{(m+n)\times(m+n-p)}$ constitute a J-pseudo orthogonalization matrix, and R is a $p \times p$ upper triangular matrix. Notice that R is indeed the J-pseudo Cholesky factor of the J-pseudo sample covariance matrix of Z in that

$$Z^T J Z = [R^T\ 0] H^T J H \begin{bmatrix} R \\ 0 \end{bmatrix} = R^T R. \tag{3.2}$$

Rewriting (3.1) as

$$[\, \mathbf{z}_1\ \mathbf{z}_2\ \cdots\ \mathbf{z}_p\,] = [\, \mathbf{h}_1\ \mathbf{h}_2\ \cdots\ \mathbf{h}_p\,] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} \\ & r_{22} & \cdots & r_{2p} \\ & & \ddots & \vdots \\ & & & r_{pp} \end{bmatrix}, \tag{3.3}$$

leads to

$$\begin{aligned} \mathbf{z}_1 &= r_{11}\mathbf{h}_1 \\ &\vdots \\ \mathbf{z}_i &= r_{1i}\mathbf{h}_1 + r_{2i}\mathbf{h}_2 + \cdots + r_{ii}\mathbf{h}_i = \sum_{j=1}^{i} r_{ji}\mathbf{h}_j \\ &\vdots \\ \mathbf{z}_p &= \sum_{j=1}^{p} r_{ji}\mathbf{h}_j. \end{aligned} \tag{3.4}$$

Multiplying $\mathbf{z}_1^T J$ on the left-hand-side and $r_{11}\mathbf{h}_1^T J$ on the right-hand-side to the first equation of (3.4), and also noticing that $\|\mathbf{h}_1\|_J^2 = 1$, we have

$$r_{11} = \|\mathbf{z}_1\|_J \tag{3.5}$$

and

$$\mathbf{h}_1 = \mathbf{z}_1 / r_{11}. \tag{3.6}$$

Next, *pseudo correlating* or taking the pseudo inner product of the second equation in (3.4) with $\mathbf{h}_1$, i.e., premultiplying it by $\mathbf{h}_1^T J$, we have

$$r_{12} = < \mathbf{h}_1, \mathbf{z}_2 >_J . \tag{3.7}$$

r_{22} can be obtained by taking the pseudo norm of $\mathbf{z}_2 - r_{12}\mathbf{h}_1 = r_{22}\mathbf{h}_2$ which gives

$$r_{22} = \|\mathbf{z}_2 - r_{12}\mathbf{h}_1\|_J, \tag{3.8}$$

and $\mathbf{h}_2$ is also readily known as

$$\mathbf{h}_2 = (\mathbf{z}_2 - r_{12}\mathbf{h}_1)/r_{22}. \tag{3.9}$$

By continuing this procedure, at the $i-th$ step, all of the i nonzero elements in the $i-th$ *subcolumn* of R can be computed by pseudo correlating the $i-th$ equation in (3.4) with $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{i-1}$ respectively, i.e.,

$$\begin{aligned} r_{1i} &= <\mathbf{h}_1, \mathbf{z}_i>_J, \\ &\vdots \\ r_{ji} &= <\mathbf{h}_j, \mathbf{z}_i>_J, \\ &\vdots \\ r_{i-1,i} &= <\mathbf{h}_{i-1}, \mathbf{z}_i>_J, \end{aligned} \tag{3.10}$$

and r_{ii} and $\mathbf{h}_i$ are given as follows:

$$r_{ii} = \|\mathbf{z}_i - r_{1i}\mathbf{h}_i - \cdots - r_{i-1,i}\mathbf{h}_{i-1}\|_J \tag{3.11}$$

and

$$\mathbf{h}_i = (\mathbf{z}_i - r_{1i}\mathbf{h}_i - \cdots - r_{i-1,i}\mathbf{h}_{i-1})/r_{ii}. \tag{3.12}$$

Therefore, a Gram-Schmidt pseudo orthogonalization (GSPO) procedure is derived as follows:

Algorithm 1 (Gram-Schmidt Pseudo Orthogonalization)

for $j = 1, \cdots, p$, **do**

$\mathbf{t} = \mathbf{z}_j$;

for $i = 1, \cdots, j-1$, **do**

$r_{ij} =< \mathbf{h}_i, \mathbf{z}_j >_J$;

$\mathbf{t} = \mathbf{t} - r_{ij}\mathbf{h}_i$;

end;

$r_{jj} = \|\mathbf{t}\|_J$;

$\mathbf{h}_j = \mathbf{t}/r_{jj}$;

end.

To derive the modified Gram-Schmidt pseudo orthogonalization (MGSPO), we first take the pseudo inner products of every equations in (3.4) with $\mathbf{z}_1$ and also notice that $< \mathbf{h}_i, \mathbf{h}_j >_J = \delta_{ij}$, so we have

$$\begin{aligned} \|\mathbf{z}_1\|_J^2 &= r_{11}^2 \qquad \text{and} \qquad \mathbf{h}_1 = \mathbf{z}_1/r_{11} , \\ &\vdots \\ < \mathbf{z}_1, \mathbf{z}_i >_J &= r_{11}r_{1i} , \\ &\vdots \\ < \mathbf{z}_1, \mathbf{z}_p >_J &= r_{11}r_{1p} . \end{aligned} \tag{3.13}$$

Thus, the first row of R, $r_{11}, r_{12}, \ldots, r_{1p}$, and $\mathbf{h}_1$, the first column of H, can be computed. Next, subtract $r_{1i}\mathbf{h}_1$ from $\mathbf{z}_i$, $i = 2, \cdots, p$,

$$\begin{aligned} \hat{\mathbf{z}}_2 &= \mathbf{z}_2 - r_{12}\mathbf{h}_1 = r_{22}\mathbf{h}_2 , \\ &\vdots \\ \hat{\mathbf{z}}_i &= \mathbf{z}_i - r_{1i}\mathbf{h}_1 = \sum_{j=2}^{i} r_{ji}\mathbf{h}_j , \\ &\vdots \\ \hat{\mathbf{z}}_p &= \mathbf{z}_p - r_{1p}\mathbf{h}_1 = \sum_{j=2}^{p} r_{ji}\mathbf{h}_j . \end{aligned} \tag{3.14}$$

Similarly, the second row of R, $r_{22}, \ldots, r_{2p}$, and the second column of H, $\mathbf{h}_2$, are ready to compute as follows:

$$\begin{aligned} \|\hat{\mathbf{z}}_2\|_J^2 &= r_{22}^2 \quad \text{and} \quad \mathbf{h}_2 = \hat{\mathbf{z}}_2 / r_{22} , \qquad (3.15) \\ &\vdots \\ < \hat{\mathbf{z}}_2, \hat{\mathbf{z}}_i >_J &= r_{22} r_{2i} , \qquad (3.16) \\ &\vdots \\ < \hat{\mathbf{z}}_2, \hat{\mathbf{z}}_p >_J &= r_{22} r_{2p} . \end{aligned}$$

Continuing this procedure, R and H can be fully obtained. This process is essentially a *modified Gram-Schmidt* version of pseudo orthogonalization method in that the columns of Z are successively subtracted by those *pseudo projected* vectors determined by pseudo inner products and R is computed row by row while H is determined column by column. A modified Gram-Schmidt pseudo orthogonalization algorithm is thus given as follows:

Algorithm 2 (MGSPO (I))

for $i = 1, \cdots, p$, **do**

$r_{ii} = \|\mathbf{z}_i\|_J$;

$\mathbf{h}_i = \mathbf{z}_i / r_{ii}$;

for $j = i+1, \cdots, p$, **do**

$r_{ij} =< \mathbf{h}_i, \mathbf{z}_j >_J$;

$\mathbf{z}_j = \mathbf{z}_j - r_{ij} \mathbf{h}_i$;

end;

end.

This algorithm requires $(m+n)p^2$ multiply-and-add, $p(p-1)/2$ divisions and p square roots operations, namely, $\mathcal{O}((m+n)p^2)$ flops. It is noted that another MGSPO algorithm where R is computed column by column [26], can also be derived in a similar way, and is given as follows:

Algorithm 3 (MGSPO (II))

for $j = 1, \cdots, p$, **do**

for $i = 1, \cdots, j-1$, **do**

$r_{ij} =< \mathrm{h}_i, \mathbf{z}_j >_J$;

$\mathbf{z}_j = \mathbf{z}_j - r_{ij}\mathrm{h}_i$;

end;

$r_{jj} = \|\mathbf{z}_j\|_J$;

$\mathrm{h}_j = \mathbf{z}_j / r_{jj}$;

end.

Based on the procedures above, we have the following theorem.

Theorem 2 (Pseudo Orthogonal Decomposition) *For any* $Z \in \mathcal{R}^{(m+n)\times p}$, $J = \begin{bmatrix} I_m & \\ & -I_n \end{bmatrix}$, *an* J*-pseudo orthogonal decomposition of* Z, $Z = HR$, *exists and is unique subject to the signs of each row in the upper-triangular matrix* R *and the signs in the columns of the pseudo orthogonal matrix* H, *if and only if* $Z^T J Z$ *is positive definite.*

3.2 Square-Root-Free Triangularization Algorithms

For many computations, it is advantageous to reduce the number of square root operations or even eliminate them altogether. To this end, we can consider the inverse of the diagonal elements of R and decompose R into

$$
\begin{aligned}
R &= D_R^{1/2} R_D \\
&= \begin{bmatrix} 1/r_{11} & & & \\ & 1/r_{22} & & \\ & & \ddots & \\ & & & 1/r_{pp} \end{bmatrix} \begin{bmatrix} r_{11}^2 & r_{11}r_{12} & \cdots & r_{11}r_{1p} \\ & r_{22}^2 & \cdots & r_{22}r_{2p} \\ & & \ddots & \vdots \\ & & & r_{pp}^2 \end{bmatrix}.
\end{aligned} \tag{3.17}
$$

It is noted that the operation of $D_R^{1/2}$ is only stated here for symbolic purpose; our interest is essentially to find R_D(D_R can be obtained from the diagonal elements of R_D), or, r_{ij}^2, for $1 \leq i \leq p; i \leq j \leq p$. A square-root-free MGSPO procedure to obtain the upper triangular matrix R, or equivalently R_D, is given below.

Algorithm 4 (Sqrt-Free MGSPO)

for $i = 1, \ldots, p$, **do**

  for $j = i, \ldots, p$, **do**

    $r_{ii}r_{ij} = < \mathbf{z}_i, \mathbf{z}_j >_J$;

    $\mathbf{z}_j = \mathbf{z}_j - (r_{ii}r_{ij}/r_{ii}^2)\mathbf{z}_i$;

  end;

end.

3.3 Downdating the Cholesky Factor

In adaptive signal processing, it is often necessary to keep the Cholesky factor only, and then successively update/downdate this upper-triangular matrix as new/old data rows become available. Updating via orthogonal transformations such as Householder transformation or Givens rotation method are well known. Here we only consider using the Gram-Schmidt method to perform pseudo orthogonalization of 2.28) which now becomes

$$Z = \begin{bmatrix} R \\ D \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} \\ & r_{22} & \cdots & r_{2p} \\ & & \ddots & \vdots \\ & & & r_{pp} \\ \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_p \end{bmatrix} \in \mathcal{R}^{(p+n)\times p}, \tag{3.18}$$

with $R \in \mathcal{R}^{p\times p}$ being upper-triangular and $D \in \mathcal{R}^{n\times p}$ the appended data rows to be discarded. We are interested in $\tilde{R} \in \mathcal{R}^{p\times p}$ such that $\tilde{R}^T\tilde{R} = Z^TJZ = R^TR - D^TD$. The signature matrix becomes $J_{p/n}$. If we denote $Z^{(0)} = Z$, then the sqrt-free MGSPO can be rewritten as follows:

for $i = 1, \cdots, p$, **do**

$\tilde{r}_{ii}^2 = \|\mathbf{z}_i^{(i-1)}\|_J^2$;

for $j = i+1, \cdots, p$, **do**

$$\begin{aligned} \tilde{r}_{ii}\,\tilde{r}_{ij} &= < \mathbf{z}_i^{(i-1)}, \mathbf{z}_j^{(i-1)} >_J ; \\ \mathbf{z}_j^{(i)} &= \mathbf{z}_j^{(i-1)} - \tfrac{\tilde{r}_{ij}\tilde{r}_{ii}}{\tilde{r}_{ii}^2}\,\mathbf{z}_i^{(i-1)} ; \end{aligned}$$

end;

end.

Notice that

$$Z^{(i)} \equiv [\, \mathbf{z}_{i+1}^{(i)} \cdots ; \mathbf{z}_{p}^{(i)} \,] \in \mathcal{R}^{(p+n)\times(p-i)} , \quad i = 1, \cdots, p-1 . \tag{3.19}$$

In this scheme, even though we can take advantages of the zeros already in R to reduce the computational cost by half, the number of flops is still of the order of p to the third power (where p is the number of the columns of R) , and is given by

$$\sum_{i=1}^{p}[(n+i)(p-i+1)+(n+i)(p-i)] = \frac{1}{3}[p^3 + 3np^2 + \frac{3p^2}{2} + \frac{p}{2}]. \tag{3.20}$$

This arises from the computational load in obtaining the lower right $\tilde{R}$. To see this, it is noticed that the work to compute $\tilde{r}_{ij}, j = i, \cdots, p$, grows linearly with the index i because pseudo inner products of size $n+i$ are required in computing the i-th row of $\tilde{R}$. This *load imbalance* among row computations while downdating(or updating) a Cholesky factor makes the MGS methods less favorable especially under massive-data(very large p) parallel computing conditions. Another drawback of this unmodified MGSPO(same for GSPO) is the difficulty in implementing an efficient VLSI architecture to accomplish downdating, although the sqrt-free computation is very attractive. To circumvent these difficulties, the previous algorithms must be reformulated to reduce the order of computation and hopefully also to facilitate VLSI processing.

Next, we will reexamine the operations involved in computing $\tilde{r}$'s and successive modification of the appended data block D. Then an improved algorithm can be derived. At each step, one row of the updated Cholesky factor will be obtained. The key idea is to represent the modified data in the old Cholesky factor in terms of the modified appended data. To clarify this idea, we will derive this algorithm in the following discussion step by step.

Step 1.

$$Z^{(0)} \equiv \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} \\ & r_{22} & \cdots & r_{2p} \\ & & \ddots & \vdots \\ & & & r_{pp} \\ \mathbf{d}_1^{(0)} & \mathbf{d}_2^{(0)} & \cdots & \mathbf{d}_p^{(0)} \end{bmatrix}$$

$$\Downarrow \{\tilde{r}_{11}, \cdots, \tilde{r}_{1p}\} \tag{3.21}$$

$$\begin{bmatrix} r_{12}^{(1)} & \cdots & r_{2p}^{(1)} \\ r_{22} & \cdots & r_{2p} \\ & \ddots & \vdots \\ & & r_{pp} \\ \mathbf{d}_2^{(1)} & \cdots & \mathbf{d}_p^{(1)} \end{bmatrix} \equiv Z^{(1)}$$

$\{ \tilde{r}_{11}, \cdots, \tilde{r}_{1p} \}$ can be computed, while $\{r_{12}^{(1)}, \cdots, r_{1p}^{(1)}\}$ and $\{\mathbf{d}_2^{(1)}, \cdots, \mathbf{d}_p^{(1)}\}$ are modified as follows:

$$\tilde{r}_{11}^2 = r_{11}^2 - \mathbf{d}_1^{(0)T}\mathbf{d}_1^{(0)}, \tag{3.22}$$

$$\tilde{r}_{11}\tilde{r}_{1j} = r_{11}r_{1j} - \mathbf{d}_1^{(0)T}\mathbf{d}_j^{(0)}, \quad j = 2, \cdots, p, \tag{3.23}$$

$$\mathbf{d}_j^{(1)} = \mathbf{d}_j^{(0)} - \frac{\tilde{r}_{11}\tilde{r}_{1j}}{\tilde{r}_{11}^2}\mathbf{d}_1^{(0)}, \quad j = 2, \cdots, p, \tag{3.24}$$

$$r_{1j}^{(1)} = r_{1j} - \frac{\tilde{r}_{11}\tilde{r}_{1j}}{\tilde{r}_{11}^2}r_{11}, \quad j = 2, \cdots, p. \tag{3.25}$$

What we need now is to express $r_{1j}^{(1)}$ in (3.25) in terms of $\mathbf{d}_j^{(1)}$ in (3.24). With (3.22) and (3.23) substituting in (3.25), it can be shown that

$$r_{1j}^{(1)} = \frac{1}{r_{11}}\mathbf{d}_1^{(0)^T}\mathbf{d}_j^{(0)} = \mathbf{f}_1^T\mathbf{d}_j^{(0)}, \quad j = 2, \cdots, p, \tag{3.26}$$

with

$$\mathbf{f}_1 \equiv \frac{1}{r_{11}}\mathbf{d}_1^{(0)}. \tag{3.27}$$

Step 2.

$$Z^{(1)} \equiv \begin{bmatrix} \mathbf{f}_1^T\mathbf{d}_2^{(1)} & \mathbf{f}_1^T\mathbf{d}_3^{(1)} & \cdots & \mathbf{f}_1^T\mathbf{d}_p^{(1)} \\ r_{22} & r_{23} & \cdots & r_{2p} \\ & r_{33} & \cdots & r_{3p} \\ & & \ddots & \vdots \\ & & & r_{pp} \\ \mathbf{d}_2^{(1)} & \mathbf{d}_3^{(1)} & \cdots & \mathbf{d}_p^{(1)} \end{bmatrix}$$

$$\Downarrow \{\tilde{r}_{22}, \cdots, \tilde{r}_{2p}\} \tag{3.28}$$

$$\begin{bmatrix} \mathbf{f}_1^T\mathbf{d}_3^{(2)} & \cdots & \mathbf{f}_1^T\mathbf{d}_p^{(2)} \\ r_{23}^{(2)} & \cdots & r_{2p}^{(2)} \\ r_{33} & \cdots & r_{3p} \\ & \ddots & \vdots \\ & & r_{pp} \\ \mathbf{d}_3^{(2)} & \cdots & \mathbf{d}_p^{(2)} \end{bmatrix} \equiv Z^{(2)}$$

$\{ \tilde{r}_{22}, \cdots, \tilde{r}_{2p} \}$ are computed and $\{r_{13}^{(2)}, \cdots, r_{1p}^{(2)}\}$, $\{r_{23}^{(2)}, \cdots, r_{2p}^{(2)}\}$ and $\{\mathrm{d}_3^{(2)}, \cdots, \mathrm{d}_p^{(2)}\}$ are modified as follows:

$$\begin{aligned}
\tilde{r}_{22}^2 &= r_{22}^2 - \mathrm{d}_2^{(1)^T}\mathrm{d}_2^{(1)} + \mathrm{d}_2^{(1)^T}\mathrm{f}_1\mathrm{f}_1^T\mathrm{d}_2^{(1)} \\
&= r_{22}^2 - \mathrm{d}_2^{(1)^T}(I - \mathrm{f}_1\mathrm{f}_1^T)\mathrm{d}_2^{(1)}, \qquad (3.29)
\end{aligned}$$

$$\begin{aligned}
\tilde{r}_{22}\tilde{r}_{2j} &= r_{22}r_{2j} - \mathrm{d}_2^{(1)^T}\mathrm{d}_j^{(1)} + \mathrm{d}_2^{(1)^T}\mathrm{f}_1\mathrm{f}_1^T\mathrm{d}_j^{(1)} \\
&= r_{22}r_{2j} - \mathrm{d}_2^{(1)^T}(I - \mathrm{f}_1\mathrm{f}_1^T)\mathrm{d}_j^{(1)}, j=3,\cdots,p, \qquad (3.30)
\end{aligned}$$

$$\mathrm{d}_j^{(2)} = \mathrm{d}_j^{(1)} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}\mathrm{d}_2^{(1)}, \quad j = 3,\cdots,p, \qquad (3.31)$$

$$\begin{aligned}
r_{1j}^{(2)} &= r_{1j}^{(1)} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}r_{12}^{(1)} = \mathrm{f}_1^T(\mathrm{d}_j^{(1)} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}\mathrm{d}_2^{(1)}) \\
&= \mathrm{f}_1^T\mathrm{d}_j^{(2)}, \quad j = 3,\cdots,p, \qquad (3.32)
\end{aligned}$$

$$r_{2j}^{(2)} = r_{2j} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}r_{22}, \quad j = 3,\cdots,p. \qquad (3.33)$$

Again, by using (3.29, 3.30, 3.31), $r_{2j}^{(2)}$ in (3.33) can be expressed in terms of $\mathrm{d}_j^{(2)}$ in (3.31), namely,

$$\begin{aligned}
r_{2j}^{(2)} &= \frac{1}{r_{22}}[r_{22}r_{2j} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}r_{22}^2] \\
&= \frac{1}{r_{22}}\{\tilde{r}_{22}\tilde{r}_{2j} + \mathrm{d}_2^{(1)^T}(I - \mathrm{f}_1\mathrm{f}_1^T)\mathrm{d}_j^{(1)} \\
&\quad - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}[\tilde{r}_{22}^2 + \mathrm{d}_2^{(1)^T}(I - \mathrm{f}_1\mathrm{f}_1^T)\mathrm{d}_2^{(1)}]\} \\
&= \frac{1}{r_{22}}[\mathrm{d}_2^{(1)^T}(I - \mathrm{f}_1\mathrm{f}_1^T)][\mathrm{d}_j^{(1)} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}\mathrm{d}_j^{(2)}] \\
&= \frac{1}{r_{22}}\mathrm{d}_2^{(1)^T}(I - \mathrm{f}_1\mathrm{f}_1^T)\mathrm{d}_j^{(1)} \\
&= \mathrm{f}_2^T\mathrm{d}_j^{(1)}, \quad j = 3,\cdots,p, \qquad (3.34)
\end{aligned}$$

with

$$\mathrm{f}_2 \equiv \frac{1}{r_{22}}(I - \mathrm{f}_1\mathrm{f}_1^T)\mathrm{d}_2^{(1)}. \qquad (3.35)$$

Step 3.

$$Z^{(2)} \equiv \begin{bmatrix} \mathrm{f}_1^T\mathrm{d}_3^{(2)} & \mathrm{f}_1^T\mathrm{d}_4^{(2)} & \cdots & \mathrm{f}_1^T\mathrm{d}_p^{(2)} \\ \mathrm{f}_2^T\mathrm{d}_3^{(2)} & \mathrm{f}_2^T\mathrm{d}_4^{(2)} & \cdots & \mathrm{f}_2^T\mathrm{d}_p^{(2)} \\ r_{33} & r_{34} & \cdots & r_{3p} \\ & r_{44} & \cdots & r_{4p} \\ & & \ddots & \vdots \\ & & & r_{pp} \\ \mathrm{d}_3^{(2)} & \mathrm{d}_4^{(2)} & \cdots & \mathrm{d}_p^{(2)} \end{bmatrix}$$

$$\Downarrow \{\tilde{r}_{33}, \cdots, \tilde{r}_{3p}\} \tag{3.36}$$

$$\begin{bmatrix} \mathrm{f}_1^T\mathrm{d}_4^{(3)} & \cdots & \mathrm{f}_1^T\mathrm{d}_p^{(3)} \\ \mathrm{f}_2^T\mathrm{d}_4^{(3)} & \cdots & \mathrm{f}_2^T\mathrm{d}_p^{(3)} \\ \mathrm{f}_3^T\mathrm{d}_4^{(3)} & \cdots & \mathrm{f}_3^T\mathrm{d}_p^{(3)} \\ r_{44} & \cdots & r_{4p} \\ & \ddots & \vdots \\ & & r_{pp} \\ \mathrm{d}_4^{(3)} & \cdots & \mathrm{d}_p^{(3)} \end{bmatrix} \equiv Z^{(3)}$$

$\{\ \tilde{r}_{33}\ ,\cdots,\ \tilde{r}_{3p}\ \}$ are computed and $\{r_{14}^{(3)}\ ,\cdots,\ r_{1p}^{(3)}\ \}$,$\{r_{24}^{(3)}\ ,\cdots,\ r_{2p}^{(3)}\ \}$,$\{r_{34}^{(3)}\ ,\cdots,\ r_{3p}^{(3)}\ \}$ and $\{\mathrm{d}_4^{(3)}\ ,\cdots,\ \mathrm{d}_p^{(3)}\ \}$ are updated as follows:

$$\begin{aligned} \tilde{r}_{33}^2 &= r_{33}^2 - \mathrm{d}_3^{(2)^T}\mathrm{d}_3^{(2)} + \mathrm{d}_3^{(2)^T}\mathrm{f}_1\mathrm{f}_1^T\mathrm{d}_3^{(2)} + \mathrm{d}_3^{(2)^T}\mathrm{f}_2\mathrm{f}_2^T\mathrm{d}_3^{(2)} \\ &= r_{33}^2 - \mathrm{d}_3^{(2)^T}(I - \mathrm{f}_1\mathrm{f}_1^T - \mathrm{f}_2\mathrm{f}_2^T)\mathrm{d}_3^{(2)}, \end{aligned} \tag{3.37}$$

$$\tilde{r}_{33}\tilde{r}_{3j} = r_{33}r_{3j} - \mathrm{d}_3^{(2)^T}\mathrm{d}_j^{(2)} + \mathrm{d}_3^{(2)^T}\mathrm{f}_1\mathrm{f}_1^T\mathrm{d}_j^{(2)}$$

$$+\mathrm{d}_3^{(2)^T}\mathrm{f}_2\mathrm{f}_2^T\mathrm{d}_j^{(2)}$$

$$= r_{33}r_{3j} - \mathrm{d}_3^{(2)^T}(I - \mathrm{f}_1\mathrm{f}_1^T - \mathrm{f}_2\mathrm{f}_2^T)\mathrm{d}_j^{(2)} \tag{3.38}$$

$$\mathrm{d}_j^{(3)} = \mathrm{d}_j^{(2)} - \frac{\tilde{r}_{3j}}{\tilde{r}_{33}}\mathrm{d}_3^{(2)}\ ,\quad j = 4, \cdots, p, \tag{3.39}$$

$$r_{1j}^{(3)} = \mathrm{f}_1^T(\mathrm{d}_j^{(2)} - \frac{\tilde{r}_{3j}}{\tilde{r}_{33}}\mathrm{d}_3^{(2)})$$

$$= \mathrm{f}_1^T\mathrm{d}_j^{(3)}\ ,\ j = 4, \cdots, p, \tag{3.40}$$

$$r_{2j}^{(3)} = \mathrm{f}_2^T(\mathrm{d}_j^{(2)} - \frac{\tilde{r}_{3j}}{\tilde{r}_{33}}\mathrm{d}_3^{(2)})$$

$$= \mathrm{f}_2^T\mathrm{d}_j^{(3)}\ ,\ j = 4, \cdots, p, \tag{3.41}$$

$$r_{3j}^{(3)} = r_{3j} - \frac{\tilde{r}_{3j}}{\tilde{r}_{33}}r_{33}\ ,\quad j = 4, \cdots, p. \tag{3.42}$$

Now, $r_{3j}^{(3)}$ in (3.42) can be written as

$$r_{3j}^{(3)} = \frac{1}{r_{33}}\mathrm{d}_3^{(2)^T}\,(I - \mathrm{f}_1\mathrm{f}_1^T - \mathrm{f}_2\mathrm{f}_2^T)\,\mathrm{d}_j^{(3)} \tag{3.43}$$

$$= \mathrm{f}_3^T\mathrm{d}_j^{(3)},\ j = 4, \cdots, p, \tag{3.44}$$

where

$$\mathrm{f}_3 \equiv \frac{1}{r_{33}}(I - \mathrm{f}_1\mathrm{f}_1^T - \mathrm{f}_2\mathrm{f}_2^T)\mathrm{d}_3^{(2)}. \tag{3.45}$$

Proceeding in this way, it can be shown that

Step i

$$\tilde{r}_{ii}^2 = r_{ii}^2 - \mathrm{d}_i^{(i-1)^T}(I - \mathrm{f}_1\mathrm{f}_1^T - \cdots - \mathrm{f}_{i-1}\mathrm{f}_{i-1}^T)\mathrm{d}_i^{(i-1)}\ , \tag{3.46}$$

$$\tilde{r}_{ii}\tilde{r}_{ij} = r_{ii}r_{ij} - \mathrm{d}_i^{(i-1)^T}(I - \mathrm{f}_1\mathrm{f}_1^T - \cdots - \mathrm{f}_{i-1}\mathrm{f}_{i-1}^T)\mathrm{d}_j^{(i-1)}\ , \tag{3.47}$$

$$\mathrm{d}_j^{(i)} = \mathrm{d}_j^{(i-1)} - \frac{\tilde{r}_{ij}}{\tilde{r}_{ii}}\mathrm{d}_i^{(i-1)}\ , \tag{3.48}$$

$$r_{kj}^{(i)} = \mathrm{f}_k^T\mathrm{d}_j^{(i)} \quad k = 1, \cdots, i;\quad j = i+1, \cdots, p, \tag{3.49}$$

$$\mathrm{f}_i \equiv \frac{1}{r_{ii}}(I - \mathrm{f}_1\mathrm{f}_1^T - \cdots - \mathrm{f}_{i-1}\mathrm{f}_{i-1}^T)\mathrm{d}_i^{(i-1)}. \tag{3.50}$$

By further defining two new quantities,

$$\mathbf{g}_i = r_{ii}\mathbf{f}_i \in \mathcal{R}^n, \quad i = 1, \cdots, p \tag{3.51}$$

$$G_i = I - \mathbf{f}_1\mathbf{f}_1^T - \cdots - \mathbf{f}_i\mathbf{f}_i^T \quad i = 1, \cdots, p, \tag{3.52}$$

with $G_0 \equiv I_n$, a recursion of formula can be easily derived from (3.51) and (3.50),

$$\mathbf{g}_i = G_{i-1}\mathbf{d}_i^{(i-1)}, \tag{3.53}$$

$$G_i = G_{i-1} - \mathbf{f}_i\mathbf{f}_i^T = G_{i-1} - \frac{\mathbf{g}_i\mathbf{g}_i^T}{r_{ii}^2} \in \mathcal{R}^{n\times n}, \tag{3.54}$$

hence (3.46) and (3.47) become

$$\tilde{r}_{ii}^2 = r_{ii}^2 - \mathbf{d}_i^{(i-1)^T}\mathbf{g}_i\,, \tag{3.55}$$

$$\tilde{r}_{ii}\tilde{r}_{ij} = r_{ii}r_{ij} - \mathbf{d}_j^{(i-1)^T}\mathbf{g}_i\,. \tag{3.56}$$

A new MGSPO algorithm with rank-n downdating is thus given below.

Algorithm 5 *New MGSPO rank-n downdating algorithm*

Initialization:

$\mathbf{d}_i^{(0)} = \mathbf{d}_i\,, \quad i = 1, \cdots, p.$

$G_0 = I_n.$

Recursion:

for $i = 1, \cdots, p$, **do**

$\mathbf{g}_i = G_{i-1}\,\mathbf{d}_i^{(i-1)}$;

$\tilde{r}_{ii}^2 = r_{ii}^2 - \mathbf{d}_i^{(i-1)^T}\,G_{i-1}\,\mathbf{d}_i^{(i-1)} = r_{ii}^2 - \mathbf{d}_i^{(i-1)^T}\mathbf{g}_i$;

$G_i = G_{i-1} - \frac{\mathbf{g}_i\,\mathbf{g}_i^T}{r_{ii}^2}$;

for $j = i+1, \cdots, p$, **do**

$\tilde{r}_{ii}\,\tilde{r}_{ij} = r_{ii}\,r_{ij} - \mathbf{d}_i^{(i-1)^T}\,G_{i-1}\,\mathbf{d}_j^{(i-1)}$

$$= r_{ii}r_{ij} - \mathbf{d}_j^{(i-1)^T}\mathbf{g}_i \ ;$$

$$\mathbf{d}_j^{(i)} = \mathbf{d}_j^{(i-1)} - \frac{\tilde{r}_{ii}\,\tilde{r}_{ij}}{\tilde{r}_{ii}^2}\,\mathbf{d}_i^{(i-1)} \ ;$$

end;

end.

It can be shown that this algorithm requires

$$\sum_{i=1}^{p}\{n^2 + n + (n^2 + n) + [\sum_{j=i+1}^{p}(n + n)]\} = np(p + 2n + 1) \tag{3.57}$$

flops. If $p \gg n$, this new method needs about $\mathcal{O}(np^2)$ flops, while the unimproved MGSPO method needs about $\mathcal{O}(np^2 + p^3/3)$ flops. Therefore, asymptotically when $2p^2 + 3p > 12n^2 + 6n$, it is more efficient to use this new method. As for HGR [2], it can be shown that it needs

$$n \cdot \sum_{i=1}^{p}[2 + 4(p - i)]$$

multiply-and-add, np square-root and $2np$ division operations, or, $\mathcal{O}(np^2 + 1.5np)$ flops. HHT [48] requires

$$\sum_{i=1}^{p}[(n + 3) + 2(n + 1)(p - i)]$$

multiply-and-add, p square-root and $(p^2 - p)/2$ division operations, or, $\mathcal{O}(np^2 + 1.5p^2)$ flops. Our newly reformulated MGSPO becomes very attractive especially when p is much larger then n among existing methods.

3.4 Simultaneously Up/Down-dating Multiple Rows

Similar to the hyperbolic Householder transformations(HHT) proposed by Rader and Steinhardt [48], our MGS pseudo orthogonalization can also perform rank-k updating and rank-ℓ downdating *simultaneously*.

Let

$$Z = \begin{bmatrix} R \\ X_{new} \\ X_{old} \end{bmatrix}$$

and

$$J = \begin{bmatrix} I_p & & \\ & I_k & \\ & & -I_\ell \end{bmatrix},$$

then an algorithm for simultaneously up/down-dating the Cholesky factor R can be derived in the similar way.

3.5 Block Systolic Triarray Using MGSPO

Fig.3.1 is a block MGSPO systolic array without square roots. The boundary and regular processor elements (PEs) of a block MGSPO systolic array without square-roots work as follows:

Boundary PE:

$$\begin{aligned} \mathbf{g}_i &= G_{i-1}\mathbf{d}_i; \\ G_i &= G_{i-1} - \frac{\mathbf{g}_i\mathbf{g}_i^T}{r_{ii}^2}; \end{aligned}$$

$$r_{ii}'^{\,2} = r_{ii}^2 - \mathbf{d}_i^T \mathbf{g}_i.$$

Regular PE:

$$\begin{aligned} r_{ii}' r_{ij}' &= r_{ii} r_{ij} - \mathbf{d}_j^T \mathbf{g}_i; \\ \mathbf{d}_j' &= \mathbf{d}_j - r_{ii}' r_{ij}' \frac{\mathbf{d}_i}{r_{ii}'^{\,2}} \end{aligned}$$

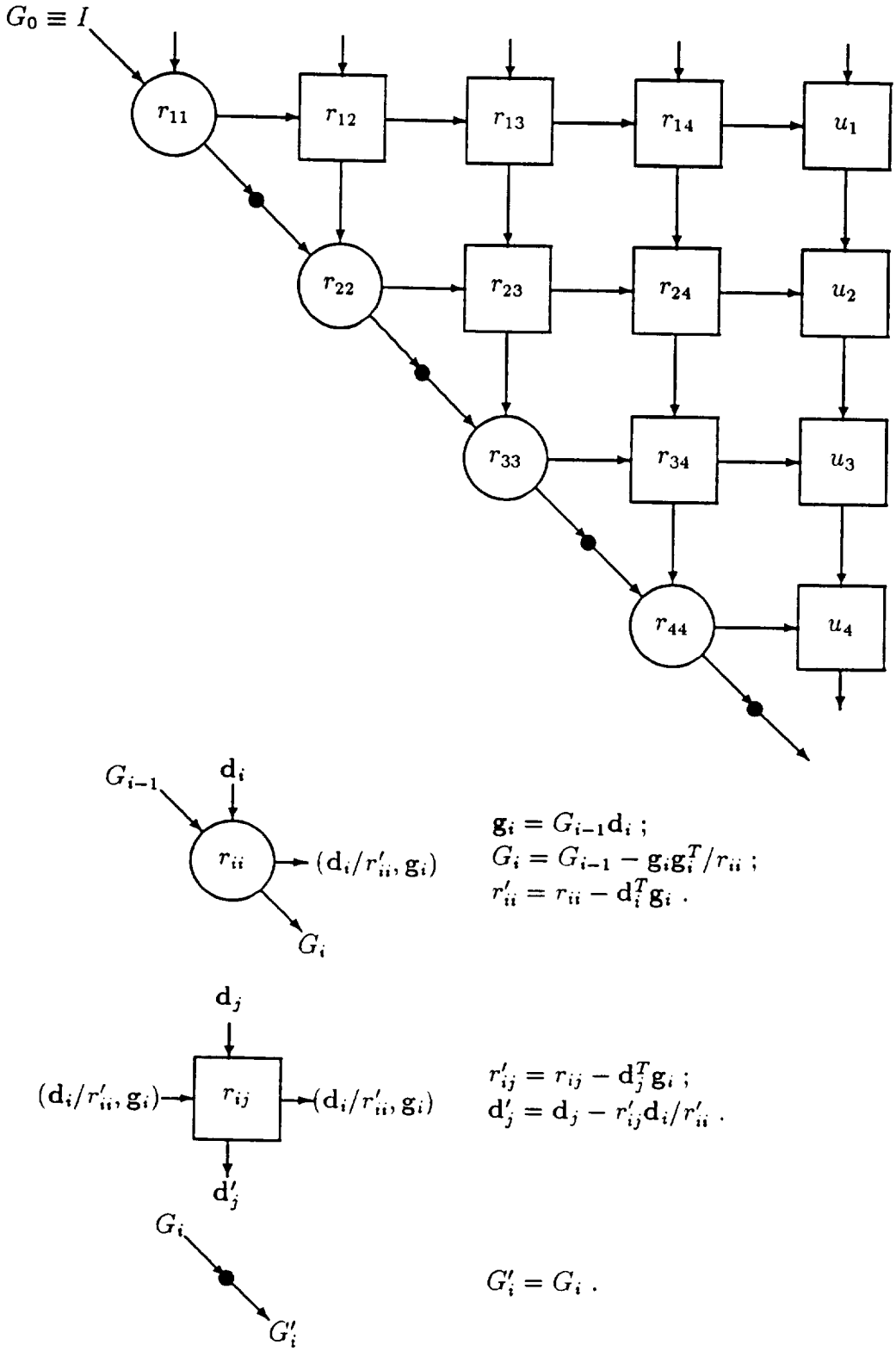


Figure 3.1: Block MGSPO systolic array without square roots

Chapter 4

Block Hyperbolic Householder Transformation

For vector-valued processing, Householder transformation is more desirable as compared to the scalar-valued Givens rotation when performing orthogonal transformations either from the viewpoint of computational flops counts or the accumulated roundoff errors under finite computations [17, 46, 58, 63].

Gentleman and Kung [19] in 1981 proposed a systolic triarray to perform QRD using Givens rotation; later McWhirter [41] in 1983 extended this structure by propagating data along the diagonal cells to obtain the most recent residual data. Kalson/Yao [31] in 1985 and Ling/Proakis [37] in 1986 derived a similar systolic structure using the MGS method. These systolic triarrays required $\mathcal{O}(p^2)$ processors, which may be objectionable for VLSI design especially when p, the order (i.e., the no. of columns) of the LS problem, is very large. Rader [47] in 1988 proposed a wafer-scale linear systolic array to reduce the number of processors down to $p/2$. All of these works only deal with recursive updating QRD.

Now, a class of systolic implementations performing up/down-dating are proposed. They are all based on block-data processing, hence generalize all previous works in this area.

4.1 Hyperbolic Householder Transformations

Rader & Steinhardt [48] in 1986 proposed a hyperbolic Householder transformation (HHT) to simultaneously perform up/down-dating. We will propose a systolic architecture implementing HHT in Sec. 4.2. To begin with, let us define a $J-$hyperbolic Householder matrix H_J as follows

$$H_J = J - 2\mathbf{h}\mathbf{h}^T/\|\mathbf{h}\|_J^2 \,, \tag{4.1}$$

where $\mathbf{h}$ is a column vector, J is a pseudo identity matrix and $< \cdot, \cdot >_J$ is the J-pseudo vector norm as defined in (2.30) and (2.32). We note that H_J is Hermitian and J-pseudo orthogonal, namely,

$$H_J = H_J^T \tag{4.2}$$

and

$$H_J^T \, J \, H_J \;=\; J \,. \tag{4.3}$$

We can compress all of the J-pseudo energy of a vector $\mathbf{a}$ into its j-th entry by premultiplying it (performing pseudo orthogonal transformation) by H_J and choosing

$$\mathbf{h} = J\mathbf{a} + \alpha \mathbf{u}_j \tag{4.4}$$

with

$$\alpha = (\pm a_j/\|a_j\|)\|\mathbf{a}\|_J \,. \tag{4.5}$$

Here $\mathbf{u}_j$ is a unit vector with all zeros except for its j-th entry. Then we have

$$H_J \mathbf{a} = -\alpha \mathbf{u}_j \, . \tag{4.6}$$

4.1.1 Hyperbolic Householder transformation with up/down-dating

An algorithm using HHT to update $A = [\mathbf{a}_1, \ldots, \mathbf{a}_p]$ and downdate $B = [\mathbf{b}_1, \ldots, \mathbf{b}_p]$ from the Cholesky factor R is presented below.

Algorithm 6 *HHT Up/downdating*

for $i = 1, \cdots, p$, *do*

$\tilde{r}_{ii} = \sqrt{r_{ii}^2 + \mathbf{a}_i^T \mathbf{a}_i - \mathbf{b}_i^T \mathbf{b}_i}$;

if $r_{ii} < 0$, $\tilde{r}_{ii} = -\tilde{r}_{ii}$;

for $j = (i+1), \cdots, p$, *do*

$\tilde{r}_{ij} = (r_{ii} r_{ij} + \mathbf{a}_i^T \mathbf{a}_j - \mathbf{b}_i^T \mathbf{b}_j)/\tilde{r}_{ii}$;

$\mathbf{a}_j = \mathbf{a}_j - \frac{\tilde{r}_{ij} + r_{ij}}{\tilde{r}_{ii} + r_{ii}} \mathbf{a}_i$;

$\mathbf{b}_j = \mathbf{b}_j - \frac{\tilde{r}_{ij} + r_{ij}}{\tilde{r}_{ii} + r_{ii}} \mathbf{b}_i$;

if $r_{ii} < 0$, $\mathbf{a}_j = -\mathbf{a}_j$; $\mathbf{b}_j = -\mathbf{b}_j$;

end;

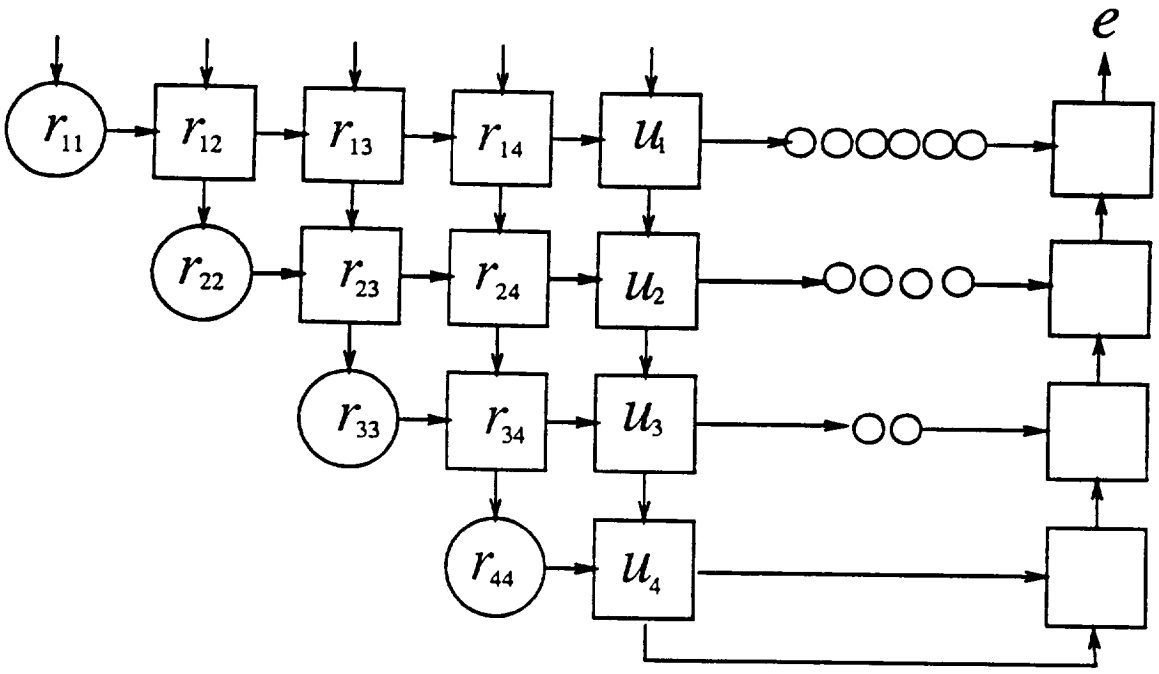
end.

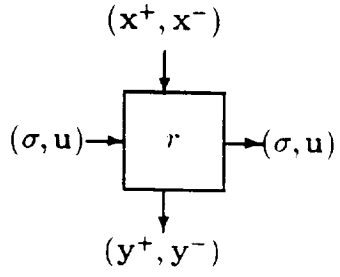
4.2 Block Hyperbolic Householder Systolic Array

Because of the nature of the hyperbolic Householder triangularization procedure, it can be shown [38] that $\widehat{Q}^{\perp} = \widehat{H}_J^{(1)^{\perp}} \cdots \widehat{H}_J^{(p)^{\perp}}$, where $\widehat{H}_J^{(j)^{\perp}} \in \mathcal{R}^{2k \times 2k}$ is the

lower right submatrix of the hyperbolic Householder reflection matrix in zeroing out the j^{th} column of appended data $\begin{bmatrix} X^+ & \mathrm{y}^+ \\ X^- & \mathrm{y}^- \end{bmatrix}$, with $[X^+ \ \mathrm{y}^+]$ and $[X^- \ \mathrm{y}^-]$ representing the new and old data block to be up/downdated respectively. The residual vector $\mathbf{e}$ therefore can be written as $\mathbf{e} = -\widehat{H}_J^{(1)^\perp} \cdots \widehat{H}_J^{(p)^\perp} \mathbf{v}$, which can be computed by a series of backward matrix-vector multiplications[38]. This systolic structure will be considered below. We note that If the block size k is equal to 1, then it reduces to that of Gentleman & Kung's [19] and McWhirter's [41] Givens algorithm.

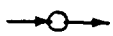
A block HHT systolic array for RLS filtering is given in Fig. 4.1. A modification suggested by Tsao in 1975 can be used to slightly reduce the flop counts and roundoff errors as well as make a two-level pipelined implementation become feasible [38]. Fig. 4.2 depicts the modified boundary and regular processors.





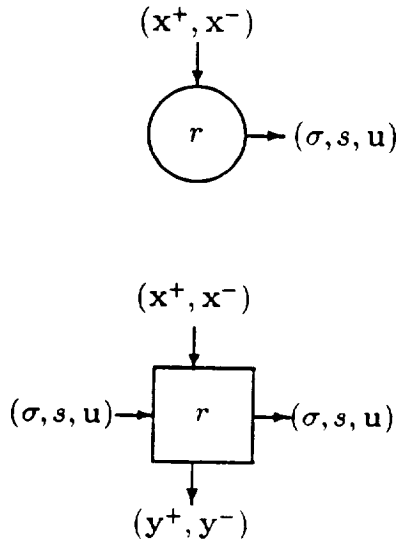
$$t = \frac{1}{\sigma} \cdot \mathbf{u}^T \cdot \begin{bmatrix} r \\ \mathbf{x}^+ \\ -\mathbf{x}^- \end{bmatrix},$$

$$\begin{bmatrix} r \\ \mathbf{y}^+ \\ \mathbf{y}^- \end{bmatrix} = \begin{bmatrix} r \\ \mathbf{x}^+ \\ \mathbf{x}^- \end{bmatrix} - t \cdot \mathbf{u}\ .$$



unit delay

Figure 4.1: Block hyperbolic Householder systolic array



$$S = r^2 + \|\mathbf{x}^+\|_2^2 - \|\mathbf{x}^-\|_2^2 ,$$
$$s = \sqrt{S} ,$$
$$\sigma = r + s ,$$
$$\mathbf{u} = \begin{bmatrix} r \\ \mathbf{x}^+ \\ \mathbf{x}^- \end{bmatrix} ,$$
$$r = s .$$

$$S = \mathbf{u}^T \cdot \begin{bmatrix} r \\ \mathbf{x}^+ \\ -\mathbf{x}^- \end{bmatrix} ,$$
$$t = -S/s ,$$
$$d = \frac{r-t}{\sigma} ,$$
$$y_i^+ = x_i^+ - d \cdot u_{i+1}, \ i = 1, \ldots, k .$$
$$y_i^- = x_i^- - d \cdot u_{k+i+1}, \ i = 1, \ldots, k .$$

Figure 4.2: Modified processing cells for Block HHT systolic array

Chapter 5

Planar and Hyperbolic Rotations

A *dual-state systolic* structure is proposed to perform joint up/down-dating operations encountered in windowed recursive least-squares (RLS) estimation problems. It is based on successively performing Givens rotations for updating and hyperbolic rotations for downdating. Due to data independency, a series of Givens and hyperbolic rotations can be interleaved and parallel processing can be achieved by alternatively performing updating and downdating both in time and space. This flip-flop nature of up/down-dating characterizes the feature of the dual-state systolic triarray. To further reduce the complexity and increase the throughput rate, Cordic cells can be used to mimic the operations of row-broadcasting where only one sign bit is propagated along each row of processors. Efficient implementation on the evaluation of optimal residuals and a transformation of the hyperbolic rotation to an algebraically equivalent orthogonal operation

to provide a more stable implementation are also considered. This systolic architecture is promising for the VLSI implementation of fixed size sliding-window recursive least-squares estimations.

5.1 Introduction

Consider a least-squares (LS) problem at time n given by,

$$X(n)w(n) \approx y(n), \tag{5.1}$$

where $X(n)$ is an $\ell \times p$ fixed-windowed data matrix with real-valued elements taken either from a single or p time-indexed multichannel data sequences,

$$X(n)=\begin{bmatrix} x_{n-\ell+1,1} & x_{n-\ell+1,2} & \cdots & x_{n-\ell+1,p} \\ x_{n-\ell+2,1} & x_{n-\ell+2,2} & \cdots & x_{n-\ell+2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{n-\ell+1}^T \\ \mathbf{x}_{n-\ell+2}^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \in \Re^{\ell \times p}, \tag{5.2}$$

and $y(n)$ is the real-valued $\ell \times 1$ desired response vector,

$$y(n) = \begin{bmatrix} y_{n-\ell+1} \\ y_{n-\ell+2} \\ \vdots \\ y_n \end{bmatrix} \in \Re^{\ell}. \tag{5.3}$$

We denote ℓ as the window size, p as the order of the system (possibly the number of sensors in a multichannel filtering problem) and n is the time index($n \geq \ell$ is assumed). The LS problem is to find a $p \times 1$ optimum coefficient vector $\hat{w}(n) \in \Re^p$, such that the Euclidean norm of its associated residual

$$e(n) = X(n)w(n) - y(n) \tag{5.4}$$

is minimized. If $X(n)$ has full column rank, then it is well known that from the normal equation (NE) approach $\hat{w}(n)$ is given by

$$\hat{w}(n) = (X^T(n)X(n))^{-1}X^T(n)y(n). \tag{5.5}$$

The increased dynamic range requirement (due to the squaring of the condition number [21]) precludes the NE method for some critical applications in modern digital signal processing. Therefore, in order to achieve the same computational precision, direct matrix factorization methods employing orthogonalization to preserve the condition number (such as the QR decomposition (QRD)), are preferred especially when it is likely that the numerical instability may arise due to ill-conditioning. Furthermore, the NE method of (5.5) is limited to block operations, while the QRD method is amenable to recursive operations implementable with various parallel and systolic architectures.

Recently, some efficient up/downdating algorithms have been proposed [2, 48, 8]. But work on efficient implementations and architectures for a fixed-windowed RLS filtering with such up/downdating is still fragmentary. In this paper, we propose two systolic arrays[33, 44, 34], which are suitable for VLSI designs, to perform fixed-windowed RLS estimation. The first one is denoted as the *dual-state* systolic triarray, which resembles Gentleman and Kung's triarray [19] with the same hardware complexity, except the clock rate of the processor is set two times higher. The second one is realized by using Cordic cells to reduce the hardware complexity. Efficient schemes to obtain optimal residual have not been fully addressed for the windowed RLS estimation. Along this direction, we consider the feasibilities and limitations based on systolic implementations. A transformation of the hyberbolic rotation to a more stable orthogonal operation is also considered in this chapter.

In Section 5.2, the basic up/downdating RLS estimation is considered, followed by the dual-state systolic architecture in Section 5.3, and Cordic processor implementations in Section 5.4. In Section 5.5, we consider the recursive estimation of optimal residual with systolic implementation. Finally, in Section 5.6, a transformation of the hyperbolic rotation to a more stable orthogonal operation is derived. Conclusions are then given in Section 5.7.

5.2 Windowed RLS Estimation

Suppose at time n, the QRD of $[X(n) \vdots y(n)]$, where $X(n)$ and $y(n)$ are given by (5.2) and (5.3) respectively, is available. Then

$$Q(n)[X(n)\vdots y(n)] = \begin{bmatrix} R(n) & \vdots & u(n) \\ 0 & \vdots & v(n) \end{bmatrix}, \tag{5.6}$$

where $Q(n) \in \Re^{\ell \times \ell}$ is orthogonal and $R(n) \in \Re^{p \times p}$ is upper triangular. Thus the optimum $\hat{w}(n)$ is given [21] by

$$R(n)\hat{w}(n) = u(n). \tag{5.7}$$

$R(n)$ is called the Cholesky factor of $X^T(n)X(n)$ in that $R^T(n)R(n) = X^T(n)X(n)$. The Cholesky factor can be obtained by computing the $p \times p$ *sample covariance matrix* $X^T(n)X(n)$ first, followed by Cholesky decomposition. But, this method squares the condition number in forming the covariance matrix. A numerically more stable approach is to perform QRD directly on the data matrix $X(n)$ and in this way the condition number of the LS problem is maintained.

Now at time $n+1$, we want to obtain $R(n+1), u(n+1)$ and hence $w(n+1)$ with the minimum effort. If the window size is growing, then we can simply

update $R(n)$ by p Givens orthogonal transformations to zero out $\mathbf{x}_{n+1}^T$ and obtain $R(n+1)$ [41]. But with a fixed sliding window scheme, in addition to zeroing out the new data row $\mathbf{x}_{n+1}^T$ by orthogonalization, it is still necessary to *downdate* the obsolete data row, $\mathbf{x}_{n-\ell+1}^T$. We define *updating* as a series of Givens rotation operations such that an *additive* rank-one modification of the Cholesky factor is accomplished, and *downdating* as hyperbolic rotation operations such that a *subtractive* rank-one modification is made. It is noticed that at time $n+1$, the data matrix

$$X(n+1) = \begin{bmatrix} \mathbf{x}_{n-\ell+2}^T \\ \vdots \\ \mathbf{x}_n^T \\ \mathbf{x}_{n+1}^T \end{bmatrix} \tag{5.8}$$

is obtained by adding a new row data $\mathbf{x}_{n+1}^T$ and removing an old row data $\mathbf{x}_{n-\ell+1}^T$ from $X(n)$ as can be seen by comparing (5.2) with (5.8). Since $R^T(n+1)R(n+1) = X^T(n+1)X(n+1)$, we have

$$R^T(n+1)R(n+1) = R^T(n)R(n) + \mathbf{x}_{n+1}\mathbf{x}_{n+1}^T - \mathbf{x}_{n-\ell+1}\mathbf{x}_{n-\ell+1}^T. \tag{5.9}$$

Rader and Steinhardt [48] proposed a hyperbolic Householder transformation to update multiple new data rows and downdate multiple undesired ones simultaneously. Alexander *et al.* [2] suggested performing orthogonal rotations for updating followed by hyperbolic rotations for downdating. It can be shown that hyperbolic rotation is merely a degenerate case of hyperbolic Householder transformation, if we do not distinguish a rotation matrix from a reflection matrix [21]. This is analogous to a Givens rotation can be considered as a special case of a Householder transformation. To facilitate systolic array processing, we will adopt the latter approach for windowed RLS filtering which involves only

scalar computations.

5.2.1 Up/down-dating Cholesky factor

The basic up/downdating of the Cholesky factor is considered in this section. Given $[R(n)\vdots u(n)]$, we can obtain $[R(n+1)\vdots u(n+1)]$ by first updating

$$\begin{bmatrix} R(n) & \vdots & u(n) \\ \mathbf{x}_{n+1}^T & \vdots & y_{n+1} \\ \mathbf{x}_{n-\ell+1}^T & \vdots & y_{n-\ell+1} \end{bmatrix} \tag{5.10}$$

via p Givens rotations, *i.e.*,

$$\begin{aligned} & G_{p,p+1}\cdots G_{2,p+1}G_{1,p+1}\begin{bmatrix} R(n) & \vdots & u(n) \\ \mathbf{x}_{n+1}^T & \vdots & y_{n+1} \\ \mathbf{x}_{n-\ell+1}^T & \vdots & y_{n-\ell+1} \end{bmatrix} \\ &= \begin{bmatrix} \tilde{R}(n+1) & \vdots & \tilde{u}(n+1) \\ 0 & \vdots & v_1(n+1) \\ \mathbf{x}_{n-\ell+1}^T & \vdots & y_{n-\ell+1} \end{bmatrix}, \end{aligned} \tag{5.11}$$

then downdating the right-hand-side via p hyperbolic rotations, *i.e.*,

$$\begin{aligned} & H_{p,p+2}\cdots H_{2,p+2}H_{1,p+2}\begin{bmatrix} \tilde{R}(n+1) & \vdots & \tilde{u}(n+1) \\ 0 & \vdots & v_1(n+1) \\ \mathbf{x}_{n-\ell+1}^T & \vdots & y_{n-\ell+1} \end{bmatrix} \\ &= \begin{bmatrix} R(n+1) & \vdots & u(n+1) \\ 0 & \vdots & v_1(n+1) \\ 0 & \vdots & v_2(n+1) \end{bmatrix}. \end{aligned} \tag{5.12}$$

Here a $(p+2)\times(p+2)$ Givens rotation matrix $G_{i,p+1}$ is used to zero out the $(p+1,i)$-th element of the matrix in (5.10), *i.e.*,

$$G_{i,p+1}\begin{bmatrix}\vdots\\ \alpha_i\\ \vdots\\ \alpha_{p+1}\\ \cdot\end{bmatrix}=\begin{bmatrix} I & & & \\ & c_i & s_i & \\ & & I & \\ & -s_i & c_i & \\ & & & 1\end{bmatrix}\begin{bmatrix}\vdots\\ \alpha_i\\ \vdots\\ \alpha_{p+1}\\ \cdot\end{bmatrix}=\begin{bmatrix}\vdots\\ \sqrt{\alpha_i^2+\alpha_{p+1}^2}\\ \vdots\\ 0\\ \cdot\end{bmatrix}, \tag{5.13}$$

where

$$c_i=\alpha_i/\sqrt{\alpha_i^2+\alpha_{p+1}^2}\quad\text{and}\quad s_i=\alpha_{p+1}/\sqrt{\alpha_i^2+\alpha_{p+1}^2}. \tag{5.14}$$

Similarly, a $(p+2)\times(p+2)$ hyperbolic rotation matrix $H_{i,p+2}$ is used to zero out the $(p+2,i)$-th element of the matrix in (5.11),

$$H_{i,p+2}\begin{bmatrix}\vdots\\ \alpha_i\\ \vdots\\ \alpha_{p+2}\end{bmatrix}=\begin{bmatrix} I & & & \\ & \tilde{c}_i & -\tilde{s}_i & \\ & & I & \\ & -\tilde{s}_i & \tilde{c}_i & \end{bmatrix}\begin{bmatrix}\vdots\\ \alpha_i\\ \vdots\\ \alpha_{p+2}\end{bmatrix}=\begin{bmatrix}\vdots\\ \sqrt{\alpha_i^2-\alpha_{p+2}^2}\\ \vdots\\ 0\end{bmatrix}, \tag{5.15}$$

where

$$\tilde{c}_i=\alpha_i/\sqrt{\alpha_i^2-\alpha_{p+2}^2}\quad\text{and}\quad \tilde{s}_i=\alpha_{p+2}/\sqrt{\alpha_i^2-\alpha_{p+2}^2}. \tag{5.16}$$

Since $G_{i,p+1}$ only affects the i-th and $(p+1)$-th rows of the matrix in (5.10), and $H_{i,p+2}$ affects the i-th and $(p+2)$-th rows, we can combine (5.11) and (5.12) in the following manner,

$$H_{p,p+2}\cdots H_{2,p+2}H_{1,p+2}G_{p,p+1}\cdots G_{1,p+1}\begin{bmatrix} R(n) & \vdots & u(n)\\ \mathbf{x}_{n+1}^T & \vdots & y_{n+1}\\ \mathbf{x}_{n-\ell+1}^T & \vdots & y_{n-\ell+1}\end{bmatrix}$$

$$= \begin{bmatrix} R(n+1) & \vdots & u(n+1) \\ 0 & \vdots & v_1(n+1) \\ 0 & \vdots & v_2(n+1) \end{bmatrix}. \tag{5.17}$$

5.3 Dual-State Systolic Triarray

Similar to the systolic QRD triarray proposed by Gentleman and Kung [19], which only performs updating, a *dual-state* systolic triarray performing both updating and downdating is given in Fig. 5.1. In a multi-channel filtering problem, for every sensor (*i.e.*, column of the data matrix) there is a delay buffer of window size ℓ to queue up the data. Therefore each data will be first fetched and processed (updated) and then stays in the queuing buffer for ℓ data clocks and finally will be reprocessed (downdated) by the triarray. Before the skewed data rows enter the arrays, there is an array of selection switches that alternatively take in new data and old data. The clock rate for the processors is set at twice the input data rate so that both new and old data can be processed within one data clock. We use a black circle • to denote a processor working on a Givens rotation(updating) and a white circle o to denote a hyperbolic rotation(downdating). We also note that only one control bit is required in determining whether updating or downdating operation needs to be performed.

To this dual-state systolic triarray, data rows are skewed with updating and downdating data interleaved to form a sequence of up/down-dating wavefronts which will then impact upon this triarray sequentially. All of the wavefronts are consistent, *i.e.*, the involved processors will all perform updating or downdating according to the underlying wavefront. As one updating wavefront finds its way along the triarray, one downdating wavefront follows immediately behind, and

then followed by another updating wavefront, and so forth.

Every processor, after experiencing one updating wavefront, will switch from updating to downdating operation as the next downdating wavefront will pass through it immediately following the previous updating wavefront. Therefore, all processors perform updating and downdating successively. Thus they are doing flip-flops in *time*, which characterizes the temporal duality of this systolic triarray.

A spatial duality can be also observed as follows. While a processor is performing updating, all its adjacent processors, either vertical or horizontal , but not diagonal neighbors, are performing downdating. As an example, consider the $(2,3)$ processor in Fig. 5.1. When this internal cell is performing updating, its right neighbor, the $(2,4)$ internal cell, and its lower neighbor, the $(3,3)$ boundary cell, are being impacted by the downdating wavefront just *before* the updating wavefront that impacts upon the $(2,3)$ internal cell (recall that up/downdating wavefronts occur consecutively). Similarly, its left neighbor, the $(2,2)$ boundary cell, and its upper neighbor, the $(1,3)$ internal cell, must be performing downdating, too, as these two neighbor cells are confronting the downdating wavefront which follows right *after* the updating wavefront associated with this $(2,3)$ cell. We therefore say that this triarray also performs flip-flops in *space*.

In all, for each time snapshot, we see all processors are doing updating and downdating evenly distributed over the entire triarray, and for the next snapshot, they change their roles. The phenomenon of flip-flops both in time and space characterizes the dual-state systolic triarrays. The wavefronts for the updating and downdating then propagate pairwise toward the lower-right direction in the triarray.

5.4 Cordic Processors

Cordic (*co*ordinate *r*otation *di*gital *c*omputation) processors [62, 1, 14, 47] have been shown to be able to efficiently perform Givens and hyperbolic rotations with simple operations like `add`, `subtract` and `shift`, and one fixed-number multiplication.

5.4.1 Givens rotations

First consider the determination of the rotational angle θ, such that a vector $[a, b]^T$ is rotated into $[\frac{a}{|a|}\sqrt{a^2+b^2}, 0]^T$, *i.e.*,

$$\begin{bmatrix} \frac{a}{|a|}\sqrt{a^2+b^2} \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}. \tag{5.18}$$

We can split θ approximately into N predetermined minirotational angles with the proper choice of the directions of these angles, such that each minirotation only involves additions and binary shifts. To see this, a recurrence of minirotations can be written as

$$\begin{aligned} \begin{bmatrix} a_{i+1} \\ b_{i+1} \end{bmatrix} &= \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} a_i \\ b_i \end{bmatrix} \\ &= \cos\theta_i \begin{bmatrix} 1 & \tan\theta_i \\ -\tan\theta_i & 1 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \end{bmatrix} \\ &= \cos\theta_i \begin{bmatrix} a_i + \rho_i 2^{-i} b_i \\ -\rho_i 2^{-i} a_i + b_i \end{bmatrix}, i = 0, 1, \cdots, N-1, \end{aligned} \tag{5.19}$$

where

$$\begin{bmatrix} a_0 \\ b_0 \end{bmatrix} \equiv \begin{bmatrix} a \\ b \end{bmatrix} \tag{5.20}$$

and

$$\tan\theta_i = \rho_i 2^{-i}. \tag{5.21}$$

The planar sign bit ρ_i is determined by

$$\rho_i = \begin{cases} 1\,, & \text{if } a_i b_i \geq 0, \\ -1\,, & \text{otherwise,} \end{cases} \tag{5.22}$$

and the intentional choice of the minirotaion angle θ_i in (5.21) renders the **shift-by-i bits** (multiplied by 2^{-i}) operations.

If the number of minirotation stages N is large enough, it can be shown [62] that

$$\begin{bmatrix} a_N \\ b_N \end{bmatrix} = \left(\prod_{i=0}^{N-1} \cos\theta_i\right) \left(\prod_{i=0}^{N-1} \begin{bmatrix} 1 & \tan\theta_i \\ -\tan\theta_i & 1 \end{bmatrix}\right) \begin{bmatrix} a \\ b \end{bmatrix} \tag{5.23}$$

$$\approx \begin{bmatrix} \frac{a}{|a|}\sqrt{a^2+b^2} \\ 0 \end{bmatrix}. \tag{5.24}$$

$\mathcal{K}_c = \prod_{i=0}^{N-1} \cos\theta_i \approx 0.60725$ is called a planar rotation correction factor and is usually independent of N when N is large enough. The rotational angle is thus uniquely determined by the planar sign bits ρ_i's,

$$\theta \approx \sum_{i=0}^{N-1} \rho_i \tan^{-1} 2^{-i}. \tag{5.25}$$

In our updating scheme, it is necessary to apply the same rotation to all the subsequent data on the two involved data rows (*i.e*, with one being in the triarray and the other the new data row being updated). In fact, it is *not* necessary to wait until all the minirotation planar sign bits ρ_i's are generated from the boundary cell. In order to take advantage of the fact that all the subsequent data on these two rows of data are to be rotated in the same manner as that in the boundary

cell, we can pipeline these minirotaion angles to the internal cells as soon as they become available. Therefore every time a planar sign bit is generated by the boundary cell, it can propagate to the rest of its right-hand-side internal cells such that the others can start doing minirotations as soon as possible. Thus along the horizontal direction, the clock rate of the mini-clock of the Cordic cells is $N+1$ times the rate of the vertical direction, which is set equal to twice the incoming data rate. Because the systolic miniclock rate along the horizontal data rows is much faster than the incoming data rate, we can consider the Givens rotation as almost simultaneously applied to every data on these data rows, or, the rotational angles are being broadcast along the remaining internal cells in the same row.

5.4.2 Hyperbolic rotations

For the same reason as above, a sequence of mini-hyperbolic rotations can be found to accomplish a hyperbolic rotation as follows:

$$\begin{bmatrix} \frac{a}{|a|}\sqrt{a^2-b^2} \\ 0 \end{bmatrix} = \begin{bmatrix} \cosh\phi & -\sinh\phi \\ -\sinh\phi & \cosh\phi \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}. \tag{5.26}$$

Now, a recurrence of mini-hyperbolic rotations are given as

$$\begin{aligned} \begin{bmatrix} a_{i+1} \\ b_{i+1} \end{bmatrix} &= \begin{bmatrix} \cosh\phi_i & -\sinh\phi_i \\ -\sinh\phi_i & \cosh\phi_i \end{bmatrix} \begin{bmatrix} a_i \\ b_i \end{bmatrix} \\ &= \cosh\phi_i \begin{bmatrix} 1 & -\tanh\phi_i \\ -\tanh\phi_i & 1 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \end{bmatrix} \\ &= \cosh\phi_i \begin{bmatrix} a_i - \sigma_i 2^{-i-1} b_i \\ -\sigma_i 2^{-i-1} a_i + b_i \end{bmatrix}, \quad i = 0, \cdots, N-1, \end{aligned} \tag{5.27}$$

where

$$\begin{bmatrix} a_0 \\ b_0 \end{bmatrix} \equiv \begin{bmatrix} a \\ b \end{bmatrix} \tag{5.28}$$

and

$$\tanh \phi_i = \sigma_i 2^{-i-1}, \tag{5.29}$$

and the hyperbolic sign bit σ_i is determined by

$$\sigma_i = \begin{cases} 1\,, & \text{if } a_i b_i \geq 0, \\ -1\,, & \text{otherwise.} \end{cases} \tag{5.30}$$

If N is large enough, it can be shown [1] that

$$\begin{bmatrix} a_N \\ b_N \end{bmatrix} = \left(\prod_{i=0}^{N-1} \cosh \phi_i \right) \left(\prod_{i=0}^{N-1} \begin{bmatrix} 1 & -\tanh \phi_i \\ -\tanh \phi_i & 1 \end{bmatrix} \right) \begin{bmatrix} a \\ b \end{bmatrix} \tag{5.31}$$

$$\approx \begin{bmatrix} \frac{a}{|a|}\sqrt{a^2 - b^2} \\ 0 \end{bmatrix}. \tag{5.32}$$

We call $\mathcal{K}_h = \prod_{i=0}^{N-1} \cosh \phi_i \approx 1.2051$ the hyperbolic rotation correction factor.

5.4.3 Cordic cells

The Cordic implementation of the dual-state systolic array has the same general triarray structure as that in Fig. 5.1. Since we have split a rotation into N minirotations and one correction factor multiplication, the time needed to perform a basic processor operation in Fig. 5.1 will be divided into $N+1$ miniclocks, too. We notice that rotating two rows of data needs not take place sequentially from one column to another, which is the case considered in Fig. 5.1. In fact we can broadcast the rotational parameters $((c, s)$ or $(\tilde{c}, \tilde{s})$ from the boundary cell

to all of its right-hand-side internal cells such that their rotations can be completed simultaneously. Unfortunately, in implementing VLSI circuits, the routing difficulty incurred due to non-local connections prohibits such a broadcast operation. This is the reason why a systolic array processing algorithm is much favored from the VLSI viewpoints in addition to its massive parallelism and regularity. However, by using Cordic cells, we are able to mimic the row broadcasting of rotational parameters with only local connections.

Cordic rotations distinguish themselves by performing minirotations sequentially. The minirotational parameters are carried solely in a stream of sign bits. These minirotational bits can be sequentially passed along without waiting for the availability of the entire bit stream. Therefore, the right-hand-side internal cells can start doing minirotations as soon as these sign bits are available. The stream of sign bits are propagated horizontally along the right-hand-side internal cells. By doing this, new data are skewed with only one miniclock in between, instead of one processor clock. We also observe that the Cordic implementation reduces the wavefronts of skewed data from an tilting slope of 1 to $1/(N+1)$. If N is sufficiently large, we can say the data is essentially not skewed and a rotation is taking place simultaneously on each row of the triarray.

Fig. 5.3 (a) and (b) depict the boundary and internal Cordic processing cells, while Fig. 5.3 (c) and (d) describe the associated sign bit generator and shift register operation. To differentiate between the following updating and downdating operations, all the downdating parameters are enclosed in the parentheses. We also use a solid arrow $\downarrow$ to represent a data movement at a system clock rate and a dashed arrow $--\!>$ at a miniclock rate in these figures. Along the horizontal direction, instead of passing the rotational parameters $c, s(\tilde{c}, \tilde{s})$ at a system

clock rate (which is the same as the clock rate along the vertical direction), the minirotational sign bits ρ_i's (σ_i's) move towards the right at a faster miniclock rate.

The boundary cell of Fig. 5.3 (a) is responsible for determining the sign bits ρ_i's (σ_i's). It has an internal memory to store the diagonal element r in the upper triangular matrix. At the first miniclock, $i = 0$ (i denotes the miniclock index of a complete rotational cycle), it fetches data y from above. During the following miniclocks $0 < i < N$, r and y are cyclically fedback to the minirotator to successively generate the minirotation sign bit of $\rho_i(\sigma_i)$ and propagate it to the right-hand-side internal cell. In the last minirotation stage of $i = N$, the internal data r is multiplied by a correction factor $\mathcal{K}_c(\mathcal{K}_h)$ and restored to its local memory. This completes a rotational cycle of the boundary Cordic cell.

As to the internal Cordic cell in Fig. 5.3 (b), it also takes in external data from above in the first miniclock, then successively feedbacks data and rotates according to incoming sign bits. In the meantime, these sign bits are also propagated to the right. In the last miniclock, both data r and y on two feedback arms are multiplied by the correction factors, with one restored to its local memory and the other output downward for further processing.

Both boundary and internal Cordic cells share many architectural similarities. The differences between updating and downdating in Fig. 5.3 (a) and (b) are: (1). the correction factors; (2). the shift register indices differ by one; (3). the signs at the lower left adder input of the minirotators.

5.5 Recursive Estimation of Optimal Residuals

We have considered the recursive evaluation of coefficient vector $\hat{w}(n)$ in Section 2. In many applications such as beamformation, array processing and filtering, and communication, the optimal weight coefficient vector may not be of direct interest. Instead, we may be interested in the newest optimal residual $\hat{e}_n$ which is the last (*i.e.*, the ℓ-th) element of $\hat{e}(n)$ in (5.4). Information is then extracted from the optimal residual for various applications. In this section, we consider an efficient implementation to obtain the newest residuals under the up/downdating operations.

From (5.6), we can separate $Q(n)$ into two terms as

$$Q(n) = \begin{bmatrix} Q_1(n) \\ Q_2(n) \end{bmatrix}, \tag{5.33}$$

where $Q_1(n) \in \Re^{p \times l}$, $Q_2(n) \in \Re^{(l-p) \times l}$, such that

$$Q_1(n)X(n) = R(n),$$

$$Q_2(n)X(n) = 0.$$

Also from (5.4), (5.6), and (5.17), we can rewrite the optimal residual vector as

$$\hat{e}(n+1) = -Q_2^T(n+1) \begin{bmatrix} v_1(n+1) \\ v_2(n+1) \end{bmatrix}. \tag{5.34}$$

Thus, a basic issue is the efficient recursive evaluation of $Q_2(n+1)$. Define

$$\bar{Q}(n+1) = \begin{bmatrix} Q_1(n) & & \\ & 1 & \\ & & 1 \end{bmatrix}, \tag{5.35}$$

then

$$\bar{Q}(n+1)[X(n+1) \vdots y(n+1)] = \begin{bmatrix} R(n) & \vdots & u(n) \\ \mathbf{x}_{n+1}^T & \vdots & y_{n+1} \\ \mathbf{x}_{n-l+1}^T & \vdots & y_{n-l+1} \end{bmatrix}. \tag{5.36}$$

From (5.17) and discussions in Section 2, denote

$$H(n+1) = H_{p,p+2} \cdots H_{2,p+2} H_{1,p+2},$$

$$G(n+1) = G_{p,p+2} \cdots G_{2,p+2} G_{1,p+2}.$$

Then we have

$$Q(n+1) = H(n+1)G(n+1)\bar{Q}(n+1) \tag{5.37}$$

if updating is performed first and

$$Q(n+1) = G(n+1)H(n+1)\bar{Q}(n+1) \tag{5.38}$$

if downdating is performed first. Suppose updating is performed first, then we have

$$Q(n+1) = H(n+1)Q_u(n+1), \tag{5.39}$$

where $Q_u(n+1) = G(n+1)\bar{Q}(n+1)$ is defined as the Q matrix associated with updating only. It can be shown that G is of the form

$$G(n+1) = \begin{bmatrix} Z(n+1) & h(n+1) & 0 \\ k^T(n+1) & \prod_{i=1}^{p} c_i & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.40}$$

where $Z(n+1)$ is a $p \times p$ matrix, and therefore Q_u is of the form

$$Q_u(n+1) = \begin{bmatrix} Z(n+1)Q_1(n) & h(n+1) & 1 \\ k^T(n+1)Q_1(n) & \prod_{i=1}^{p} c_i & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.41}$$

It can also be shown that H is of the form

$$H(n+1) = \begin{bmatrix} \tilde{Z}(n+1) & 0 & \tilde{h}(n+1) \\ 0 & 1 & 0 \\ \tilde{k}^T(n+1) & 0 & \prod_{i=1}^{p} \tilde{c}_i \end{bmatrix}, \tag{5.42}$$

and therefore $Q(n+1)$ is of the form

$$Q(n+1) = \begin{bmatrix} \tilde{Z}(n+1)Z(n+1)Q_1(n) & \tilde{Z}(n+1)h(n+1) & \tilde{h}(n+1) \\ k^T(n+1)Q_1(n) & \prod_{i=1}^{p} c_i & 0 \\ \tilde{k}^T(n+1)Z(n+1)Q_1(n) & \tilde{k}^T(n+1)h(n+1) & \prod_{i=1}^{p} \tilde{c}_i \end{bmatrix}. \tag{5.43}$$

From (5.34) and (5.41), we can obtain the residual vector when the updating wavefront passes through the array and is given by

$$\hat{e}_u(n+1) = \begin{bmatrix} \tilde{e}_u(n+1) \\ e_{u_1}(n+1) \\ e_{u_2}(n+1) \end{bmatrix} = \begin{bmatrix} -Q_1^T(n)k(n+1)v_1(n+1) \\ -\prod_{i=1}^{p} c_i \cdot v_1(n+1) \\ -v_2(n+1) \end{bmatrix}, \tag{5.44}$$

where e_{u_1} and e_{u_2} are the newest residuals associated with the updating and downdating respectively at time $n+1$. Since at this point the downdating has not yet been performed, $e_{u_2}(n+1)$ is not considered as a residual.

From (5.34) and (5.43), we can obtain the residual vector when the downdating wavefront passes through the triarray. Again, we are only interested in the newest residuals (the last two elements) and are given by

$$\begin{bmatrix} e_1(n+1) \\ e_2(n+1) \end{bmatrix} = \begin{bmatrix} -\prod_{i=1}^{p} c_i \cdot v_1(n+1) - h^T(n+1)\tilde{k}(n+1)v_2(n+1) \\ -\prod_{i=1}^{p} \tilde{c}_i \cdot v_2(n+1) \end{bmatrix}, \tag{5.45}$$

where e_1 and e_2 are the residuals associated with updating and downdating respectively. From (5.17), it can be seen that $v_1(n+1)$ and $v_2(n+1)$ can be

obtained naturally from the up/downdating operations in the triarray. If the updating parameters c_i's are propagated down to the diagonal boundary cells and are cumulatively multiplied as in [41], when the updating wavefront passes through the triarray, the term $\prod c_i$ in (5.44) can be obtained. A multiplier call is then used to obtain $e_{u_1}(n+1) = -\prod_{i+1}^{p} c_i \cdot v_1(n+1)$ as in [41]. In fact, although the window size is l as described in (5.2), the residual $e_{u_1}(n+1)$ is estimated from $[\mathbf{x}_{n-l+1}, \cdots, \mathbf{x}_n, \mathbf{x}_{n+1}]^T$ and $[y_{n-l+1}, \cdots, y_n, y_{n+1}]^T$ of window size $\ell+1$ since downdating of $\mathbf{x}_{n-l+1}$ has not yet been performed. That is

$$e_{u_1}(n+1) = \mathbf{x}_{n+1}^T \hat{w}_{[n-l+1,n+1]} - y_{n+1}, \tag{5.46}$$

where $\hat{w}_{[n-l+1,n+1]}$ denotes the optimal coefficient vector estimated from data $[\mathbf{x}_{n-l+1}, \cdots, \mathbf{x}_n, \mathbf{x}_{n+1}]^T$ and $[y_{n-l+1}, \cdots, y_n, y_{n+1}]^T$.

Also, when the downdating wavefront passes through the triarray, if $\tilde{c}_i$'s are propagated down to the diagonal boundary cells and are cumulatively multiplied. From (5.45), the downdating residual can be obtained easily. It is estimated from $[\mathbf{x}_{n-l+2}, \cdots, \mathbf{x}_n, \mathbf{x}_{n+1}]^T$ of window size ℓ. That is,

$$e_2(n+1) = \mathbf{x}_{n-l+1}^T \hat{w}_{[n-l+2,n+1]} - y_{n-l+1}. \tag{5.47}$$

Obviously, the residual at time $n-l+1$ is *post estimated* by data from $n-l+2$ to $n+1$ and appears at time $n+1$. This kind of property may or may not be of practical interest in real-life applications. As to the updating residual $e_1(n+1)$, due to the term $h^T(n+1)\tilde{k}(n+1)v_2(n+1)$ which is not available from the systolic implementation, we are unable to extract $e_1(n+1)$ from the triarray. However, (5.45) provides a simple relation for this updating residual before and after the downdating. That is,

$$e_1(n+1) = e_{u_1}(n+1) - h^T(n+1)\tilde{k}(n+1)v_2(n+1). \tag{5.48}$$

If downdating is performed first, then by the same argument as above, we can obtain

$$e_{d_2}(n+1) = -\prod_{i=1}^{p} \tilde{c}_i \cdot v_2(n+1) = \mathbf{x}_{n-l+1}^T \hat{w}_{[n-l+2,n]} - y_{n-l+1}, \tag{5.49}$$

and

$$e_1(n+1) = -\prod_{i=1}^{p} c_i \cdot v_1(n+1) = \mathbf{x}_{n+1}^T \hat{w}_{[n-l+2,n+1]} - y_{n+1}. \tag{5.50}$$

From (5.4) and (5.8), it is obvious that this $e_1(n+1)$ is the exact residual we are looking for. However, a drawback for this scheme is that downdating first before the updating may incur a numerical stability problem [2]. A dual-state up/downdating systolic array for the recursive residual estimation is also shown in Fig. 5.1.

5.6 Tradeoffs Between Numerical Stability and Hardware Complexity

From the numerical stability point of view, the usage of hyperbolic rotation for downdating may be objectionable, even though it has been shown to be forward (weakly) stable in [3]. One of the reasons is that, from Fig. 5.2, $\tilde{c}$ and $\tilde{s}$ generated by the boundary cell can be very large. Once these $\tilde{c}$ and $\tilde{s}$ are sent to the internal cells for further processing, the computations may involve two other parameters r and z which are not scaled (*i.e.*, they can also be very large). Therefore, z' and the updated r may suffer large amount of roundoff errors, or even overflows. To stabilize this problem, we consider an algebraically equivalent set of orthogonal parameters to replace $\tilde{c}$ and $\tilde{s}$. Along this line, let us first consider the relation between updating and downdating.

Suppose we have a known $p \times p$ upper triangular matrix R and want to downdate a vector $\mathbf{x}$ to obtain an upper triangular matrix $\tilde{R}$. That is,

$$\tilde{R}^T\tilde{R} = R^TR - \mathbf{x}\mathbf{x}^T. \tag{5.51}$$

If we know $\tilde{R}$ instead of R, then

$$\tilde{R}^T\tilde{R} + \mathbf{x}\mathbf{x}^T = R^TR \tag{5.52}$$

becomes an updating problem.

To downdate R, we use a sequence of hyperbolic rotations to zero out $\mathbf{x}$ as

$$\begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix} = H_{p,p+1}\cdots H_{2,p+1}H_{1,p+1}\begin{bmatrix} R \\ \mathbf{x}^T \end{bmatrix}. \tag{5.53}$$

On the other hand, to update $\tilde{R}$, we use a sequence of Givens rotations to zero out $\mathbf{x}$ as

$$G_{p,p+1}\cdots G_{2,p+1}G_{1,p+1}\begin{bmatrix} \tilde{R} \\ \mathbf{x}^T \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix}. \tag{5.54}$$

Suppose now, for this updating problem, we know R instead of $\tilde{R}$, and $k-1$ updating has been done. That is

$$G_{k-1,p+1}\cdots G_{1,p+1}\begin{bmatrix} \tilde{R} \\ \mathbf{x}^T \end{bmatrix} = \begin{bmatrix} R^{k-1} \\ \tilde{R}_{p-k+1} \\ 0,\cdots,0,x_k^{(k-1)},\cdots,x_p^{(k-1)} \end{bmatrix}, \tag{5.55}$$

where R^{k-1} denotes the first $k-1$ rows of R, $\tilde{R}_{p-k+1}$ denotes the last $p-k+1$ rows of $\tilde{R}$, and $x^{(k)}$ denotes an element of the k-th updated vector $\mathbf{x}$. At the k-th rotation, let us focus only on the k-th and the last rows, then we have

$$\begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix}\begin{bmatrix} 0, & \cdots, & 0, & \tilde{r}_{k,k}, & \cdots, & \tilde{r}_{k,p} \\ 0, & \cdots, & 0, & x_k^{(k-1)}, & \cdots & x_p^{(k-1)} \end{bmatrix}$$

$$= \begin{bmatrix} 0, & \cdots, & 0, & r_{k,k} & \cdots & \cdots, & r_{k,p} \\ 0, & \cdots, & 0, & 0 & x_{k+1}^{(k)}, & \cdots & x_p^{(k)} \end{bmatrix}, \tag{5.56}$$

where $\tilde{r}_{i,j}$ and $r_{i,j}$ are the (i,j) elements of $\tilde{R}$ and R respectively,

$$r_{k,k} = \sqrt{\tilde{r}_{k,k}^2 + x_k^{(k-1)^2}}, \tag{5.57}$$

$$c_k = \frac{\tilde{r}_{k,k}}{r_{k,k}}, s_k = \frac{x_k^{(k-1)}}{r_{k,k}},$$

and for $j = k+1, \cdots, p$,

$$\begin{cases} r_{k,j} = c_k \tilde{r}_{k,j} + s_k x_j^{(k-1)}, \\ x_j^{(k)} = -s_k \tilde{r}_{k,j} + c_k x_j^{(k-1)}. \end{cases} \tag{5.58}$$

From (5.57) and (5.58), we can solve $\tilde{r}$ and $x^{(k)}$ easily and the downdating can now be achieved by using Givens rotation parameters given by

$$\tilde{r}_{k,k} = \sqrt{r_{k,k}^2 - x_k^{(k-1)^2}}, \tag{5.59}$$

$$c_k = \frac{\tilde{r}_{k,k}}{r_{k,k}}, s_k = \frac{x_k^{(k-1)}}{r_{k,k}},$$

for $j = k+1, \cdots, p$,

$$\begin{cases} \tilde{r}_{k,j} = (r_{k,j} - s_k x_j^{(k-1)})/c_k, \\ x_j^{(k)} = -s_k \tilde{r}_{k,j} + c_k x_j^{(k-1)}. \end{cases} \tag{5.60}$$

Since c_k and s_k are bounded parameters (*i.e.*, ≤ 1), even though $\tilde{r}_{k,j}$ can be very large when computing $x_j^{(k)}$, it is only multiplied with s_k which is bounded. That is to say, $\tilde{r}_{k,j}$ will never be magnified during the computation. Thus, this scheme is more stable than hyperbolic rotation. The operations of the processing cells for the systolic implementation are shown in Fig. 5.4.

With this transformation, the operations of the downdating part are different from that of the updating part. However, both provide stable numerical

results especially under finite precision computations. If we want to make both operations identical (except fro sign differences) for implementational efficiency. Transformations on c and s of Fig. 5.2 can be performed such that new $\hat{c}$ and $\hat{s}$ are given by

$$\hat{c} = \frac{1}{c}, \qquad \hat{s} = \frac{s}{c}. \tag{5.61}$$

Therefore we have

$$\begin{aligned} r &= (r + \hat{s}y)/\hat{c}, \\ y' &= -\hat{s}r + \hat{c}y, \end{aligned} \tag{5.62}$$

which share the same form as that of the corresponding downdating operation. However, it loses the numerical property of an orthogonal transformation and hence is less desirable under finite precision computations. The operations of this updating transformation are shown in Fig. 5.4.

5.7 Conclusions

A dual-state systolic triarray performing up/down-dating operations for fixed-window RLS filtering has been proposed. While previous researches have been centered on QRD-based systolic triarray with exponentially forgetting factors to perform updating, no suitable VLSI architecture (such as that based on systolic array), has been proposed to perform fixed-window RLS filtering.

Due to the inherent similarity between updating and downdating, they can use the same hardware and alternatively pipelined to achieve parallelism in this dual-state systolic triarray. A flip-flop systolic behavior of this array is observed both in temporal and spatial domains. We also proposed Cordic cells to mimic

broadcasting along the rows in the triarray and thus eliminated the propagation of word-level rotational parameters along the rows by using minirotational sign bit streams. The hardware complexity of the Cordic cells is quite simple and the involved computations comprise of simple arithmetics. Square root and division operations are not required. We have also considered some basic systolic implementational issues related to the solution of the optimal residuals. To remedy potential round-off errors associated with downdating, a modified transformation has been considered. The issues of numerical stability and pipelined computations in the stabilized transformation have also been addressed.

We have investigated efficient algorithms and architectures for performing fixed-window RLS filtering problems that extended previously known results from updating to up/down-dating operations. The proposed new dual-state systolic triarray architectures appear promising for VLSI implementation.

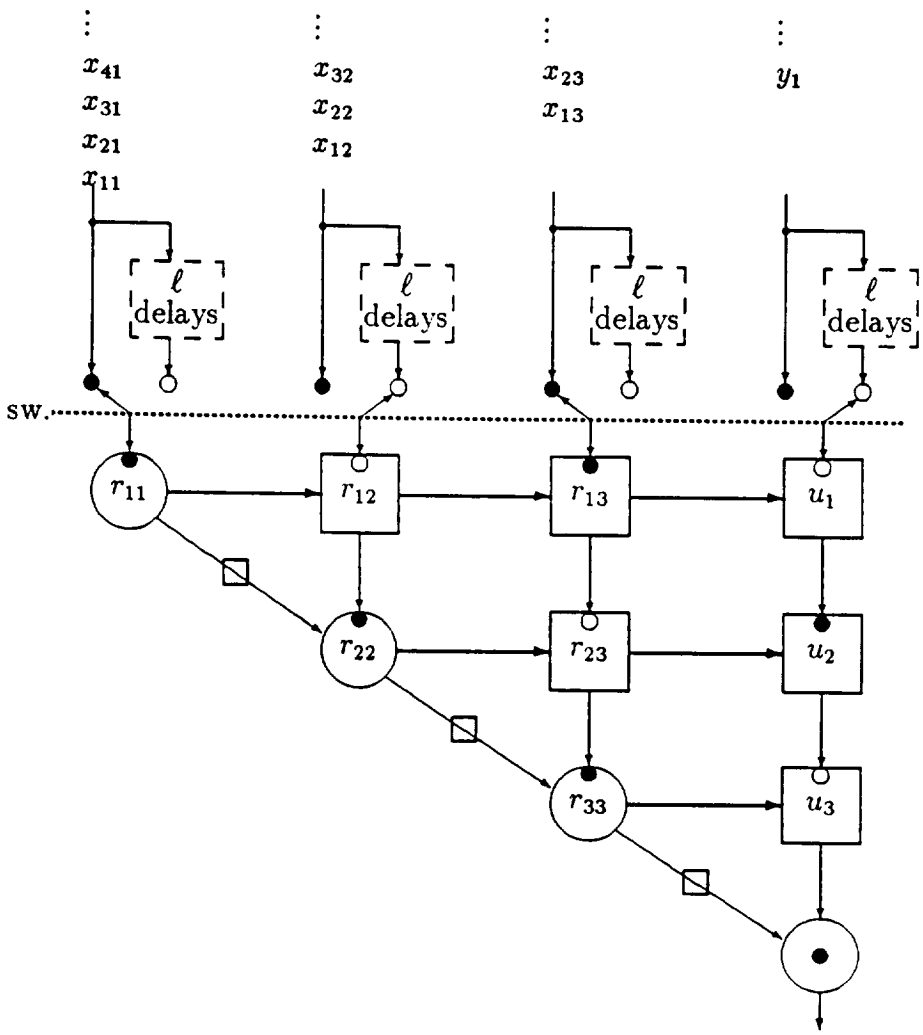


Figure 5.1: Dual-State Systolic Array for Windowed-RLS Problems.

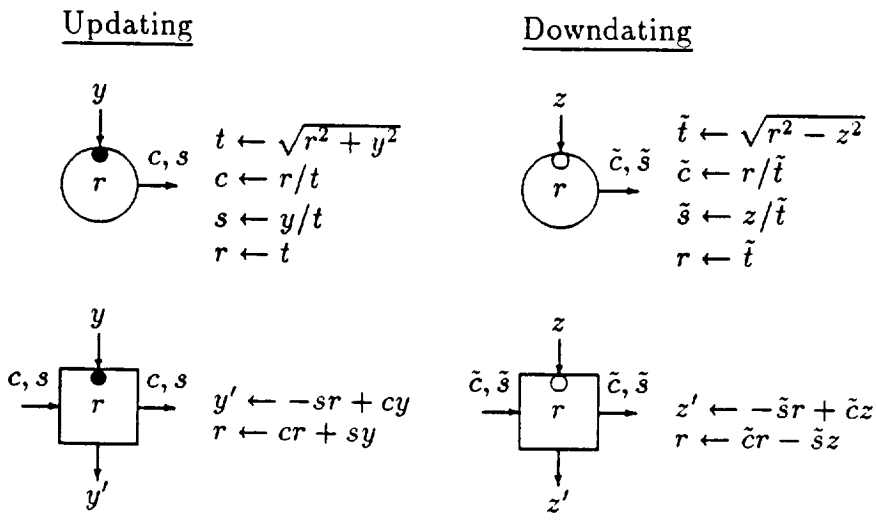


Figure 5.2: Boundary and internal cells.

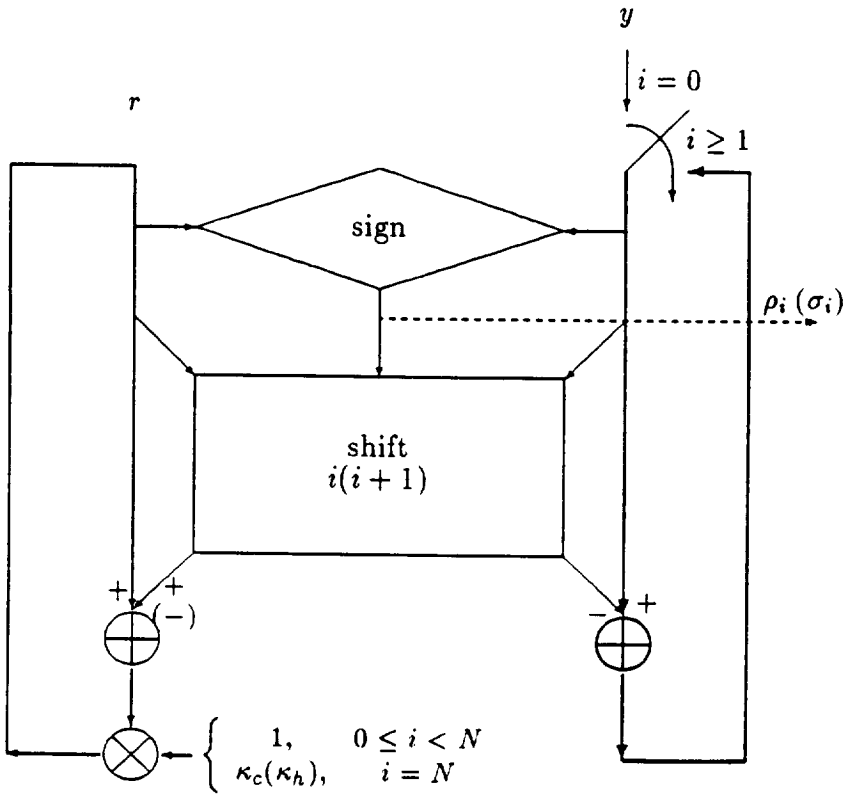


Fig. 4 (a). Boundary Cordic processor

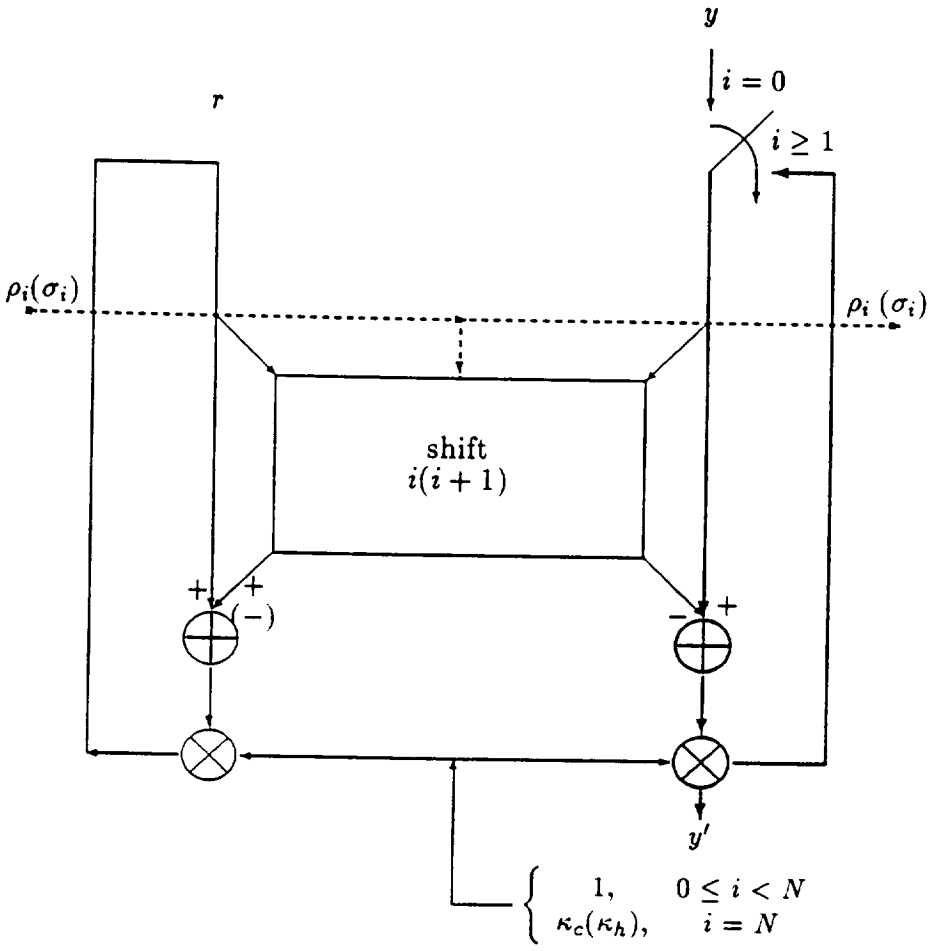


Fig. 4 (b). Internal Cordic processor

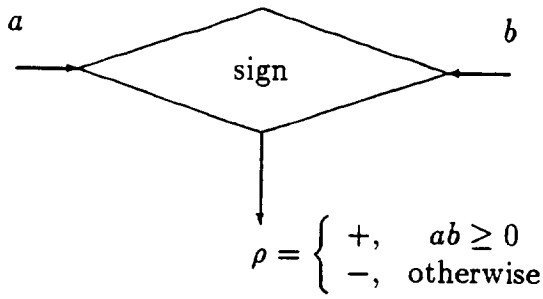


Fig. 4 (c). Sign bit

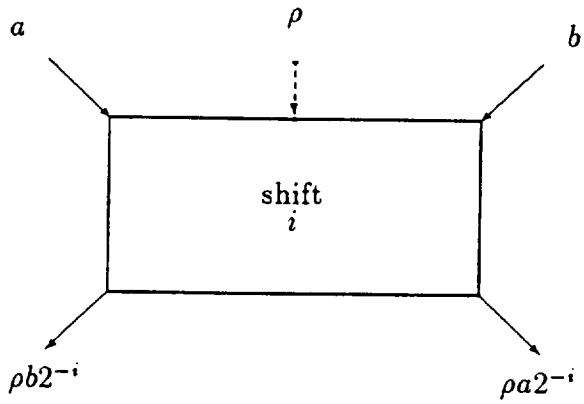


Fig. 4 (d). Shift-i operation

Figure 5.3: Cordic cells

C-2

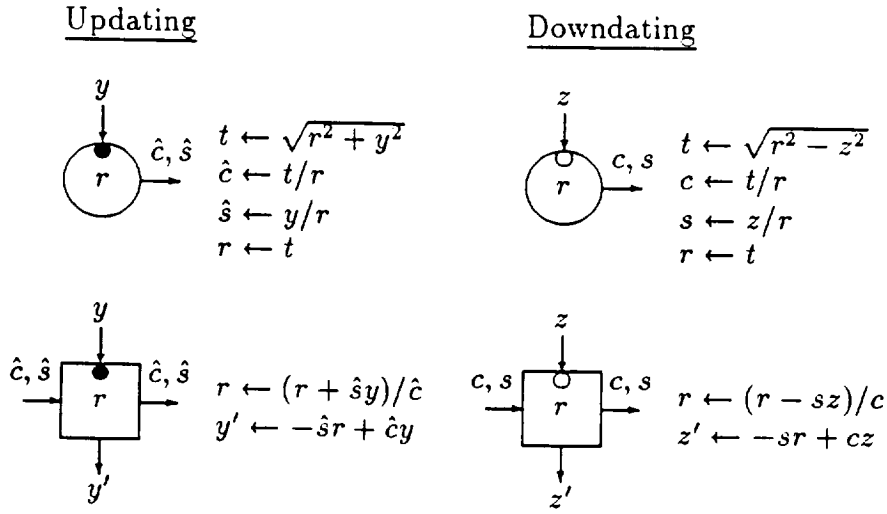


Figure 5.4: Transformed boundary and internal cells.

Chapter 6

Unified Square-Root-Free Rank-1 Up/Down-dating

Planar (Givens) and hyperbolic rotations are the most commonly used methods in performing QRD up/downdating. But the generic formula for these rotations require explicit square-root (sqrt) computations, which are quite undesirable from the practical VLSI circuit design point of view. Since the sqrt operation takes up much area and its computational time is long (due to many iterations), the associated area/time efficiency is poor.

By setting the block size k equal to 1, we can reduce all the vector operations down to scalar operations. It can be easily seen that all currently available scalar-type algorithms are merely special cases of the block-type algorithms proposed in the previous chapters. When $k = 1$, all the 3 major QRD methods, namely, HT, MGS, and Givens, can have square-root-free operations and share many similarities. However, if the block size k is not equal to 1, it's very difficult to give a square-root-free version for the HT method, while MGS still enjoys such a

fast version as can be seen in Chapter 3.

After introducing planar (Givens) and hyperbolic rotations, a generalized approach to eliminate square-root operations is derived and then it becomes clear, that all 2×2 orthogonal and pseudo-orthogonal HT, MGS, or Givens rotation fall into this generalization.

6.1 Planar and Hyperbolic Rotations

A 2×2 *planar* (Givens) rotation matrix is the most fundamental orthogonal matrix in performing QR decomposition. A *planar* (Givens) rotation is given by $\begin{bmatrix} c & s \\ -s & c \end{bmatrix}$, and is used to premultiply a two-row matrix

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_p \\ \beta_1 & \beta_2 & \cdots & \beta_p \end{bmatrix}$$

to introduce a zero element in the $(2,1)$ location such that it becomes

$$\begin{bmatrix} \alpha_1' & \alpha_2' & \cdots & \alpha_p' \\ 0 & \beta_2' & \cdots & \beta_p' \end{bmatrix},$$

where

$$c = \alpha_1 / \sqrt{\alpha_1^2 + \beta_1^2}\,, \tag{6.1}$$

$$s = \beta_1 / \sqrt{\alpha_1^2 + \beta_1^2}\,, \tag{6.2}$$

$$\alpha_1' = \sqrt{\alpha_1^2 + \beta_1^2}\,, \tag{6.3}$$

$$\alpha_j' = c\alpha_j + s\beta_j\,, \tag{6.4}$$

$$j = 2, \cdots, p.$$

$$\beta_j' = -s\alpha_j + c\beta_j\,, \tag{6.5}$$

Similarly, a *hyperbolic* rotation matrix is given by $\begin{bmatrix} \hat{c} & -\hat{s} \\ -\hat{s} & \hat{c} \end{bmatrix}$, and the corresponding parameters satisfy:

$$\hat{c} = \alpha_1/\sqrt{\alpha_1^2 - \beta_1^2}\,, \tag{6.6}$$

$$\hat{s} = \beta_1/\sqrt{\alpha_1^2 - \beta_1^2}\,, \tag{6.7}$$

$$\alpha_1' = \sqrt{\alpha_1^2 - \beta_1^2}\,, \tag{6.8}$$

$$\alpha_j' = \hat{c}\alpha_j - \hat{s}\beta_j\,, \tag{6.9}$$

$$\beta_j' = -\hat{s}\alpha_j + \hat{c}\beta_j\,, \tag{6.10}$$

$$j = 2, \cdots, p.$$

6.2 Prototypes of Generalized SQRT-Free Algorithms

In VLSI circuit design, square-root operation is expensive, because it takes up much area or is slow (due to many iterations). Therefore, it is advantageous to avoid square-root operations or minimize the required number of such operations. We will focus on how to meet this goal for the 2×2 planar and hyperbolic rotations.

By taking out a scaling factor from each row, the two rows under consideration before and after the orthogonal transformations are given by

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_p \\ \beta_1 & \beta_2 & \cdots & \beta_p \end{bmatrix} = \begin{bmatrix} \sqrt{k_1} & 0 \\ 0 & \sqrt{k_2} \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_p \\ b_1 & b_2 & \cdots & b_p \end{bmatrix} \tag{6.11}$$

and

$$\begin{bmatrix} \alpha_1' & \alpha_2' & \cdots & \alpha_p' \\ 0 & \beta_2' & \cdots & \beta_p' \end{bmatrix} = \begin{bmatrix} \sqrt{k_1'} & 0 \\ 0 & \sqrt{k_2'} \end{bmatrix} \begin{bmatrix} a_1' & a_2' & \cdots & a_p' \\ 0 & b_2' & \cdots & b_p' \end{bmatrix}. \tag{6.12}$$

Now, our task is to find the expressions for $k_1', k_2', a_1', \{(a_j', b_j'),\ j = 2, \cdots, p\}$, in terms of $k_1, k_2, \{(a_j, b_j),\ j = 1, \cdots, p\}$, such that *NO* square-root operation is actually needed. The square-root expressions of $\sqrt{k_1}$, $\sqrt{k_2}$, $\sqrt{k_1'}$, and $\sqrt{k_2'}$ in (6.11) and (6.12) are used for representational purposes only and are not actually performed.

For simplicity, let us focus on planar rotations only, similar derivations for the square-root-free hyperbolic rotation can also be obtained by replacing k_2 with $-k_2$ and k_2' with $-k_2'$. Replacing $\alpha_j = \sqrt{k_1}a_j,\ \beta_j = \sqrt{k_2}b_j,\ \alpha_j' = \sqrt{k_1'}a_j',\ \beta_j' = \sqrt{k_2}b_j'$, $j = 1, \ldots, p$, in (6.1) - (6.5) leads to

$$c = \sqrt{k_1}\, a_1 \,/\, \sqrt{k_1 a_1^2 + k_2 b_1^2}\,, \tag{6.13}$$

$$s = \sqrt{k_2}\, b_1 \,/\, \sqrt{k_1 a_1^2 + k_2 b_1^2}\,, \tag{6.14}$$

$$\sqrt{k_1'}\, a_1' = \sqrt{k_1 a_1^2 + k_2 b_1^2}\,, \tag{6.15}$$

$$\sqrt{k_1'}\, a_j' = \frac{k_1\, a_1}{\sqrt{k_1 a_1^2 + k_2 b_1^2}}\, a_j + \frac{k_2\, b_1}{\sqrt{k_1 a_1^2 + k_2 b_1^2}}\, b_j\,, \qquad j = 2, \cdots, p. \tag{6.16}$$

$$\sqrt{k_2'}\, b_j' = \frac{-\sqrt{k_1 k_2}\, b_1}{\sqrt{k_1 a_1^2 + k_2 b_1^2}}\, a_j + \frac{\sqrt{k_1 k_2}\, a_1}{\sqrt{k_1 a_1^2 + k_2 b_1^2}}\, b_j\,, \tag{6.17}$$

After simplifications we have

$$a_1' = \sqrt{\frac{k_1 a_1^2 + k_2 b_1^2}{k_1'}}\,, \tag{6.18}$$

$$a_j' = \frac{1}{\sqrt{k_1'}\,\sqrt{k_1 a_1^2 + k_2 b_1^2}}[k_1 a_1 a_j + k_2 b_1 b_j]\,, \qquad j = 2, \cdots, p. \tag{6.19}$$

$$b_j' = \frac{\sqrt{k_1 k_2}}{\sqrt{k_2'}\,\sqrt{k_1 a_1^2 + k_2 b_1^2}}[-b_1 a_j + a_1 b_j]\,, \tag{6.20}$$

To avoid square-roots, we need to determine k_1' and k_2' such that a_1', a_j' and b_j' will not require square-root operations. Let us express k_1' and k_2' as

$$k_1' = \frac{k_1 a_1^2 + k_2 b_1^2}{\mu^2}\,, \tag{6.21}$$

$$k_2' = \frac{k_1 k_2}{\nu^2(k_1 a_1^2 + k_2 b_1^2)}, \tag{6.22}$$

where μ and ν will be determined later to be any square-root-free functions of k_1, k_2, a_1, and b_1. Indeed, with (6.21) - (6.22), (6.18) - (6.20) can be computed with no square-roots, and we have the following updating formulas without square-roots:

$$k_1' = \frac{k_1 a_1^2 + k_2 b_1^2}{\mu^2}, \tag{6.23}$$

$$k_2' = \frac{k_1 k_2}{\nu^2(k_1 a_1^2 + k_2 b_1^2)} = \frac{k_1 k_2}{\mu^2 \nu^2 k_1'}, \tag{6.24}$$

$$a_1' = \mu, \tag{6.25}$$

$$a_j' = \frac{\mu}{k_1 a_1^2 + k_2 b_1^2}[k_1 a_1 a_j + k_2 b_1 b_j] = \frac{k_1 a_1 a_j + k_2 b_1 b_j}{\mu k_1'}, \quad j = 2, \cdots, p, \tag{6.26}$$

$$b_j' = \nu[-b_1 a_j + a_1 b_j], \tag{6.27}$$

$$c = \frac{a_1}{\mu}\sqrt{\frac{k_1}{k_1'}}, \tag{6.28}$$

$$s = \frac{b_1}{\mu}\sqrt{\frac{k_2}{k_1'}}. \tag{6.29}$$

Notice that square-root operations disappear in our formulas of (6.23) - (6.27) needed in the planar and hyperbolic rotations. The use of the rotation parameter c in (6.28) (with a square-root operation) will be further considered in the next section when the optimum residual e is desired. Later work will show that it is possible to obtain e without any square-root operation where the explicit computation of the rotation parameter c can be bypassed. To avoid repetitive computations and take the advantage of previous computed results, (6.24), (6.26), (6.28) and (6.29) use the newly updated k_1' of (6.23). As stated earlier, we are still free to choose those two parameters μ and ν. Different choices of μ and ν

will affect the number of multiplications and divisions, as well as the numerical stability and parallelism of computations.

It can be shown that this newly derived approach generalizes *all* of the previously known researches on the square-root-free algorithms via an proper choice of μ and ν. Among them are Gentleman('73) [20], Hammarling('74) [24], Bareiss('82) [4], Kalson and Yao('85) [31], Ling, *etc.*('86) [37], Barlow and Ispen('87) [5], Chen and Yao('88) [13], Götze and Schwiegelshohn('89) [23]. Table 6.1 lists various square-root-free algorithms and the corresponding choices of μ and ν.

6.3 SQRT-Free Triangular Array Updating and Optimum Residual Acquisition

In this section, we will apply the prototypes of sqrt-free rotations developed above to QRD-based RLS filtering problems. To be specific, we are interested in updating from

$$\begin{bmatrix} R & \mathbf{u} \\ \mathbf{x}^T & y \end{bmatrix} \tag{6.30}$$

to

$$\begin{bmatrix} R' & \mathbf{u}' \\ \mathbf{0}^T & v \end{bmatrix} \tag{6.31}$$

as given earlier in (2.12). It can be shown [41] that the $p \times p$ upper triangular matrix R' can be obtained through a sequence of p Givens rotations and the optimum residual e for the newly appended data $[\,\mathbf{x}^T : y\,]$ is given by

$$e = -(\prod_{i=1}^{p} c_i)\, v, \tag{6.32}$$

with c_i representing the cosine value of the i-th rotation angles.

Table 6.1: Choices of μ and ν for various sqrt-free Givens rotations

μ	ν	author(year)	comment
1	1	Gentleman('73)	$a_1 = 1$
$\frac{k_1a_1^2+k_2b_1^2}{k_2b_1}$	$\frac{-1}{b_1}$	"	
$\frac{k_1a_1^2+k_2b_1^2}{k_1a_1}$	$\frac{1}{a_1}$	"	
$\frac{k_1a_1^2+k_2b_1^2}{k_1a_1}$	$\frac{1}{a_1}$	Hammarling('74)	
$\frac{k_1a_1^2+k_2b_1^2}{k_1a_1}$	$\frac{-1}{b_1}$	"	
$\frac{k_1a_1^2+k_2b_1^2}{k_1a_1}$	$\frac{k_2b_1}{k_1a_1^2}$	"	
1	$\frac{1}{a_1}$	Bareiss('82)	
$a_1 + k_2b_1^2$	$\frac{1}{a_1+k_1b_1^2}$	Ling('86),Kalson/Yao('85)	$k_1a_1 = 1$
$\frac{k_1a_1^2+k_2b_1^2}{k_1a_1}$	$\frac{1}{a_1}$	Chen/Yao ('88)	$k_1a_1 = 1$
$\frac{k_1a_1^2+k_2b_1^2}{k_1k_2}$	1	Götze/Schwiegelshohn('89)	
$2^{r_1}(k_1a_1^2 + k_2b_1^2)$	2^{r_2}	Barlow/Ispen('87)	Scaled

Factoring out the scaling constants into the pre-multiplying diagonal matrix leads (6.30) to the form of

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} & u_1 \\ & r_{22} & \cdots & r_{2p} & u_2 \\ & & \ddots & \vdots & \vdots \\ & & & r_{pp} & u_p \\ x_1 & x_2 & \cdots & x_p & y \end{bmatrix} \tag{6.33}$$

$$= \begin{bmatrix} \sqrt{k_1} & & & & \\ & \sqrt{k_2} & & & \\ & & \ddots & & \\ & & & \sqrt{k_p} & \\ & & & & \sqrt{k_q} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} & a_{1,p+1} \\ & a_{22} & \cdots & a_{2p} & a_{2,p+1} \\ & & \ddots & \vdots & \vdots \\ & & & a_{pp} & a_{p,p+1} \\ b_1 & b_2 & \cdots & b_p & b_{p+1} \end{bmatrix} . \tag{6.34}$$

Unlike the previously developed formula, where we are only interested in updating k_i, a_{ij}, to k'_i, a'_{ij} and zeroing out all the b_i's, this time we do also need to know the cosine values explicitly as required in the optimum residual given in (6.32).

After the first rotation, b_1 will be zeroed-out and we have

$$k'_1 = \frac{k_1 a_{11}^2 + k_q b_1^2}{\mu_1^2} , \tag{6.35}$$

$$k_q^{(1)} = \frac{k_1 k_q}{\mu_1^2 \nu_1^2 k'_1} , \tag{6.36}$$

$$a'_{11} = \mu_1 , \tag{6.37}$$

$$a'_{1j} = \frac{1}{\mu_1 k'_1} [\, k_1 a_{11} a_{1j} + k_q b_1 b_j \,] , \qquad j = 2, \cdots, p+1 , \tag{6.38}$$

$$b_j^{(1)} = \nu_1 [\, -b_1 a_{1j} + a_{11} b_j \,] , \tag{6.39}$$

$$c_1 = \frac{a_{11}}{\mu_1} \sqrt{\frac{k_1}{k'_1}} , \tag{6.40}$$

with (μ_1, ν_1) being the parameter pair which are still free to be chosen later. Note the close analogy of (6.35) - (6.40) to those of (6.23) - (6.28). Similarly, after the i-th rotation $(1 < i \leq p)$, we have

$$k_i' = \frac{k_i a_{ii}^2 + k_q^{(i-1)} b_i^{(i-1)2}}{\mu_i^2}, \tag{6.41}$$

$$k_q^{(i)} = \frac{k_i k_q^{(i-1)}}{\mu_i^2 \nu_i^2 k_i'}, \tag{6.42}$$

$$a_{ii}' = \mu_i, \tag{6.43}$$

$$a_{ij}' = \frac{1}{\mu_i k_i'} [\, k_i a_{ii} a_{ij} + k_q^{(i-1)} b_1^{(i-1)} b_j^{(i-1)} \,], \quad j = 2, \cdots, p+1, \tag{6.44}$$

$$b_j^{(i)} = \nu_i [\, -b_i^{(i-1)} a_{ij} + a_{ii} b_j^{(i-1)} \,], \tag{6.45}$$

$$c_i = \frac{a_{ii}}{\mu_i} \sqrt{\frac{k_i}{k_i'}}. \tag{6.46}$$

After p rotations are finished, (6.34) becomes

$$\begin{bmatrix} \sqrt{k_1'} & & & & \\ & \sqrt{k_2'} & & & \\ & & \ddots & & \\ & & & \sqrt{k_p'} & \\ & & & & \sqrt{k_q'} \end{bmatrix} \begin{bmatrix} a_{11}' & a_{12}' & \cdots & a_{1p}' & a_{1,p+1}' \\ & a_{22}' & \cdots & a_{2p}' & a_{2,p+1}' \\ & & \ddots & \vdots & \vdots \\ & & & a_{pp}' & a_{p,p+1}' \\ 0 & 0 & \cdots & 0 & b_{p+1}^{(p)} \end{bmatrix} \tag{6.47}$$

which has the form of $\begin{bmatrix} R' & \mathrm{u}' \\ 0^T & v \end{bmatrix}$ in (6.31). The optimum residual e in (6.32) now becomes

$$e = -\left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i} \sqrt{\frac{k_i}{k_i'}} \right) \sqrt{k_q^{(p)}}\, b_{p+1}^{(p)}. \tag{6.48}$$

To further simplify the expression in (6.48), we notice that $k_q^{(p)}$ can be computed recursively as follows,

$$k_q^{(p)} = \left(\frac{\sqrt{k_p/k'_p}}{\mu_p\nu_p}\right)^2 k_q^{(p-1)} \tag{6.49}$$

$$= \left(\frac{\sqrt{k_p/k'_p}}{\mu_p\nu_p}\right)^2 \left(\frac{\sqrt{k_{p-1}/k'_{p-1}}}{\mu_{p-1}\nu_{p-1}}\right)^2 k_q^{(p-2)} \tag{6.50}$$

$$\vdots$$

$$= \left[\prod_{i+1}^{p} \left(\frac{\sqrt{k_i/k'_i}}{\mu_i\nu_i}\right)^2\right] k_q \,, \tag{6.51}$$

where (6.42) is used in the recursion.

With (6.51) substituted into (6.48), we have

$$e = -\left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i^2\,\nu_i} \frac{k_i}{k'_i}\right) \sqrt{k_q}\, b_{p+1}^{(p)} \,. \tag{6.52}$$

Because $\sqrt{k_q}[b_1, b_2, \ldots, b_p, b_{p+1}] = [x_1, x_2, \ldots, x_p, y]$ is the appended new data row, we are certainly free to choose $k_q = 1$ to reduce the arithmetic complexity and simplify the expression in (6.52). Therefore, a lemma on obtaining the sqrt-free optimum residual is given below.

Lemma 3 *(Sqrt-free optimum residual)*

The optimum residual e can be computed with no square-root operations as follows:

$$e = -\left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i^2\,\nu_i} \frac{k_i}{k'_i}\right) b_{p+1}^{(p)} \,. \tag{6.53}$$

McWhirter[41] successfully employed Gentleman's proposition[20] in computing the residual e without sqrt operations. By choosing

$$\mu_i = \nu_i = a_{ii} = 1, \quad 1 \le i \le p \,, \tag{6.54}$$

the optimum residual can be reduced to

$$e = -\left(\prod_{i=1}^{p} \frac{k_i}{k_i'}\right) b_{p+1}^{(p)} \qquad \text{(Gentleman/McWhirter).} \tag{6.55}$$

This result was first observed by McWhirter.

Another example can be taken from Hammarling's suggestion[24] as follows,

$$\mu_i = \frac{k_i a_{ii}^2 + k_q^{(i-1)} b_i^{(i-1)^2}}{k_i a_{ii}}, \qquad i = 1, \cdots, p. \tag{6.56}$$

$$\nu_i = 1 / a_{ii}, \tag{6.57}$$

then it follows that $k_i' = k_i a_{ii}/\mu_i$ and the optimum residual is given by

$$e = -\left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i^2 \nu_i} \frac{\mu_i}{a_{ii}}\right) b_{p+1}^{(p)} \tag{6.58}$$

$$= -\left(\prod_{i=1}^{p} \frac{1}{\mu_i \nu_i}\right) b_{p+1}^{(p)} \tag{6.59}$$

$$= -\left(\prod_{i=1}^{p} \frac{a_{ii}}{a_{ii}'}\right) b_{p+1}^{(p)} \qquad \text{(Hammarling).} \tag{6.60}$$

6.4 Conclusions

Planar (Givens) and hyperbolic rotations are the most commonly used methods in performing QRD up/downdating. Most of these rotation-based methods require explicit square-root computations, which are undesirable from the practical VLSI circuit design point of view. Since the square-root operation takes up much area and its computational time is slow (due to many iterations), the associated area/time efficiency is poor. This is the first effort to establish the basic understanding toward all known square-root-free QRD algorithms, from which the basic criterion is seen to be simple. This unified approach also provides a fundamental framework for the square-root-free RLS algorithms which are essential for practical VLSI implementations.

Chapter 7

Truncated Least-Squares Problems and Applications

In resolving closely spaced frequencies from limited amount of data samples, Tufts and Kumaresan (1982) proposed a SVD-based method to solve the forward-backward linear prediction(FBLP) least-squares problem. By imposing an excessive order in the FBLP model and then truncating small singular values to zero, this truncated SVD (TSVD) method yields a very low SNR threshold and greatly suppresses spurious frequencies. However, the massive computation required by SVD makes it unsuitable for *real time* super-resolution applications. We propose to use truncated QR methods which are amenable to VLSI implementations, such as systolic arrays, with slightly degraded performances as compared to the TSVD method. Three truncated QR methods for sinusoidal frequency estimation will be considered: (1) truncated QR without column pivoting (TQR); (2) truncated QR with re-ordered columns (TQRR); and (3) truncated QR with column pivoting (TQRP). It is demonstrated that the benefit of the TSVD method for high

frequency resolution is achievable under the truncated QR methods with much lower computational cost. Other attractive features of the proposed methods include the ease of updating, which is difficult for the SVD method, and numerical stability. Thus, the TQR methods offer efficient ways for identifying sinusoids closedly clustered in frequencies under stationary and non-stationary conditions. Some results based on the truncated normal equation approach as well as on sufficient conditions for perfect truncations based on truncated QR and SVD methods are considered. Based on the FBLP model, computer simulations and comparisons are provided for different truncation methods under various SNR's. Comparisons of asymptotic performance with large data samples are also given.

7.1 Introduction

In recent years, there is much interest in seeking efficient and effective algorithms for resolving closely spaced sinusoids in the frequency domain as well as in the spatial domain [52, 59, 10, 32, 49, 51, 29, 50, 60]. Generally, a "good" algorithm for spectral estimation should comprise of several factors, such as: high frequency resolution capability; computational efficiency; updating and downdating capability; and implementable parallel processing structure so that fast real-time applications are possible. Different methods may perform well in some aspects but suffer in the others. While the SVD-based method is well known for its robustness in resolving closely clustered sinusoids, it is not attractive from the other desirable feature points of view. In this paper, we consider several other promising approaches based on the truncated QR and least-squares techniques.

In the pioneering paper of Tufts and Kumaresan [59], a SVD-based method for

solving the forward-backward linear prediction(FBLP) least-squares (LS) problem was used to resolve the frequencies of closely spaced sinusoids from limited amount of data samples. By imposing an excessive order in the FBLP model and then truncating small singular values to zero, this truncated SVD method yields a low SNR threshold and greatly suppresses spurious frequencies. However, the massive computations required by SVD makes it unsuitable for *real time* super-resolution applications. We propose to use truncated QR and LS methods which are more amenable to VLSI implementations, such as on systolic arrays[19], with insignificantly degraded performances as compared to the TSVD method. Three different truncated QR methods will be considered, depending on the ordering of the columns of the data matrix. The first one is the truncated QR method without column shuffling (TQR). This method does not change the structure of the data matrix. A QR decomposition (QRD) of the data matrix is followed by the truncation of the lower right rank-weakly submatrix of the upper-triangular matrix. The second one is the truncated QR method with reordered columns (TQRR). The reordering of the columns is determined in an *a priori* manner [51]. Here truncation is performed on the QRD of the column-reordered data matrix. The computational cost of this TQRR method is the same as that of the first method, except for the column reshuffling. The last one is called truncated QR with column pivoting (TQRP) [21]. This method entails a series of *dynamic* swapping of columns while performing QRD. An additional computational cost is required to monitor the norms of the remaining columns in the dimension-shrinking submatrix such that the first column is replaced by the one with the largest norm in the remaining submatrix. The processing overhead of successive column swapping may be nontrivial and prohibitive in implementing a VLSI structure. All these

three truncated QR methods only involve a finite number of computations, while for the TSVD method, the number of iterations required cannot be specified in an exact manner. Based upon MATLAB computations, SVD requires about 5 to 6 times the number of flops as compared to QRD for a dense 50×50 matrix. Furthermore, we should note that QRD only requires a small number of flops for updating when new data are successively appended, while updating SVD is generally much more intractable [11]. A truncated normal equation approach will be shown to be equivalent to the TQR method except for the increased roundoff errors under finite precision computations.

A FBLP model for estimating sinusoidal frequencies is formulated first, followed by an introduction of different truncation methods and the minimum-norm solutions. Finally, comparisons of these three QR and the LS methods to the TSVD method are given based on computer simulations.

7.2 FBLP Model

Consider a complex-valued data sequence of length n,

$$\tilde{x}_i = \sum_{k=1}^{p} c_k e^{j2\pi f_k i} + w_i \equiv x_i + w_i, \quad i = 1, 2, \cdots, n, \tag{7.1}$$

where p is the number of sinusoids , complex-valued c_k comprises the amplitudes and phases of each sinusoid, and w_i is an additive white Gaussian noise. We define the signal-to-noise ratio (SNR) as

$$\text{SNR (dB)} = 20 \log(\|x\|_2 \,/\, \|w\|_2). \tag{7.2}$$

It can be shown [59] that under noise-free conditions, the frequency locations can be obtained by finding the roots of

$$S(z) = 1 - \sum_{k=1}^{\ell} g_k z^{-k} = 0 \,, \tag{7.3}$$

on the unit circle, where the complex-valued coefficients $g_k's$, $k = 1, 2, \cdots, \ell$, satisfy the following system of FBLP equations

$$\begin{bmatrix} x_\ell & x_{\ell-1} & \cdots & x_1 \\ x_{\ell+1} & x_\ell & \cdots & x_2 \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1} & x_{n-2} & \cdots & x_{n-\ell} \\ x_2^* & x_3^* & \cdots & x_{\ell+1}^* \\ x_3^* & x_4^* & \cdots & x_{\ell+2}^* \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-\ell+1}^* & x_{n-\ell+2}^* & \cdots & x_n^* \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_\ell \end{bmatrix} = \begin{bmatrix} x_{\ell+1} \\ x_{\ell+2} \\ \vdots \\ x_n \\ x_1^* \\ x_2^* \\ \vdots \\ x_{n-\ell}^* \end{bmatrix} \tag{7.4}$$

with $\ell \geq p$ representing the order of the prediction model, and $*$ the complex conjugate. We will assume that $2(n-\ell) > \ell$. For simplicity, denote (7.4) as

$$Ag = b \,, \tag{7.5}$$

where the data matrix A and the right-hand-side vector b are constructed from the data sequence $\{x_i \mid i = 1, \ldots, n\}$ in a FBLP manner. Symbolically, this will be denoted by $[A \vdots b] \equiv \{x_i \mid i = 1, \ldots, n\}_{\text{FBLP}}$. It is noted that the top half of A represents the forward prediction and is of Toeplitz form while the bottom half represents the backward prediction and is known as a Hankel form. The rank of A is p if $\min\{2(n-\ell), \ell\} \geq p$. When the noise is present, we use an $\tilde{}$ on A and b, i.e., $\tilde{A} = A + E$ and $\tilde{b} = b + e$, to denote the noise-corrupted FBLP model with

the additive noise given by $[E \vdots e] \equiv \{w_i \mid i = 1, \ldots, n\}_{\text{FBLP}}$. (7.5) now becomes the FBLP LS problem of

$$\tilde{A}g \cong \tilde{b}, \tag{7.6}$$

where $\tilde{A}$ usually has full rank due to the perturbation of the noise. One standard approach [59] is to use the TSVD method on (7.6) to obtain a rank-p approximation of the FBLP matrix $\tilde{A}$, denoted by $\tilde{A}^{(p)}_{SVD}$, followed by solving for a minimum norm LS solution of g given by

$$\tilde{A}^{(p)}_{SVD}\, g \;\cong \tilde{b}\,. \tag{7.7}$$

Then the frequencies can be computed by finding the phases of the roots of (7.3) close to the unit circle or searching for the peaks on the pseudo-spectrum $1/|S(exp(j2\pi f)|^2$, $-0.5 \leq f \leq 0.5$. Notice that the proper choice of the prediction order ℓ depends on p, the number of sinusoids, which may or may not be known in advance. Fig. 7.1 depicts a flowchart diagram summarizing the estimation of harmonics frequencies based on the FBLP model.

7.3 Truncation Methods

In this section, we consider the rank-p approximation of the FBLP matrix $\tilde{A}$ and subsequently solve for the minimum-norm solution $\tilde{g}^{(p)}$. For many LS problems, ill-conditioning can be troublesome, and truncation methods are known to be useful in stablizing the solutions at the cost of slightly increased residual errors. The rationale is that the condition number[21] of a matrix, defined as the ratio of the largest to the smallest singular values, can be used to characterize a worst case bound on the LS solution when the underlying matrix is subject to some unknown

perturbation or round-off errors. When the smaller singular values are discarded, their ill effects on the LS solution are reduced(i.e., stabilized), or we may say that the new condition number of the truncated system is decreased(i.e., stabilized). However, this truncated LS solution, although stablized, will be different from the one that gives the minimum residual; instead, its associated residual with respect to the original *un*truncated linear system will be larger. Therefore, the tradeoff for the truncated linear system lies between an increased stability versus a decreased residual.

Let

$$\tilde{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^H = [\tilde{U}_1\ \tilde{U}_2]\begin{bmatrix} \tilde{\Sigma}_1 & 0 \\ 0 & \tilde{\Sigma}_2 \end{bmatrix}\begin{bmatrix} \tilde{V}_1^H \\ \tilde{V}_2^H \end{bmatrix} \tag{7.8}$$

and

$$\tilde{A}\Pi = \tilde{Q}\tilde{R} = [\tilde{Q}_1\ \tilde{Q}_2]\begin{bmatrix} \tilde{R}_{11} & \tilde{R}_{12} \\ 0 & \tilde{R}_{22} \end{bmatrix} \tag{7.9}$$

be the SVD and QRD of the $2(n-\ell) \times \ell$ complex-valued matrix $\tilde{A}$ respectively, where ^{H} denotes the Hermitian of a complex-valued matrix or vector and Π is a column-permutation matrix and will be explained later. $\tilde{\Sigma}_1 = diag(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_p)$ and $\tilde{\Sigma}_2 = diag(\tilde{\sigma}_{p+1}, \ldots, \tilde{\sigma}_\ell)$ represent nonincreasing singular values. $\tilde{R}_{11} \in \mathcal{C}^{p \times p}$, $\tilde{R}_{12} \in \mathcal{C}^{p \times (\ell-p)}$, and $\tilde{R}_{22} \in \mathcal{C}^{\ell-p \times (\ell-p)}$, while $\tilde{R}$ is an upper-triangular matrix.

$$\tilde{U}=[\tilde{U}_1\ \tilde{U}_2] = [\tilde{u}_1, \ldots, \tilde{u}_p,\ \tilde{u}_{p+1}, \ldots, \tilde{u}_\ell] \in \mathcal{C}^{2(n-\ell)\times\ell}, \tag{7.10}$$

$$\tilde{V}=[\tilde{V}_1\ \tilde{V}_2] = [\tilde{v}_1, \ldots, \tilde{v}_p,\ \tilde{v}_{p+1}, \ldots, \tilde{v}_\ell] \in \mathcal{C}^{\ell\times\ell}, \tag{7.11}$$

and

$$\tilde{Q} = [\tilde{Q}_1\ \tilde{Q}_2] = [\tilde{q}_1, \ldots, \tilde{q}_p,\ \tilde{q}_{p+1}, \ldots, \tilde{q}_\ell] \in \mathcal{C}^{2(n-\ell)\times\ell} \tag{7.12}$$

all have orthonormal columns, i.e., $\tilde{u}_i^H\tilde{u}_j = \tilde{v}_i^H\tilde{v}_j = \tilde{q}_i^H\tilde{q}_j = \delta_{ij}$.

In the absence of noise, $\tilde{\Sigma}_2 = \tilde{R}_{22} = 0$. Here the permutation matrix $\Pi = [\pi_1, \ldots, \pi_\ell]$ is used to represent different methods of performing QRD with column interchanges. Now we want to preserve as much of the energy as possible (with respect to the Frobenius norm defined below) in the trapezoidal matrix $[\tilde{R}_{11}\ \tilde{R}_{12}]$ of (7.9). Equivalently, we want to leave as little as possible the energy residing in the lower right submatrix $\tilde{R}_{22}$, which will be truncated. This approach amounts to selecting the columns of $\tilde{A}$ in an order such that the column with the largest linear independency will be selected first. This procedure is repeated for the shrinking submatrix.

There are at least 3 possible methods for determining the permutation matrix Π while performing QRD, which are:

1. For QRD with no pivoting, Π is simply an identity matrix.

2. QRD with pre-ordered columns [51] determines Π according to a column index maximum-difference bisection rule. Here we select the first and the ℓ^{th} columns, followed by the column $\lceil \frac{1+\ell}{2} \rceil$ halfway between 1 and ℓ. Then we pick the columns that lie in the midway of those ones which are already selected, i.e., $\lceil (1 + \lceil \frac{1+\ell}{2} \rceil)/2 \rceil$, $\lceil (\lceil \frac{1+\ell}{2} \rceil)/2 \rceil + \ell)/2 \rceil$, and so on. This selection rule does not depend on the real-time data in $\tilde{A}$. The underlying reason for this *ad hoc* fixed-ordering scheme is to provide the selected columns with a possibly maximum differences or minimum linear dependency among these columns. This scheme was motivated due to the nature of the matrix $\tilde{A}$ arranged in the form of (7.4) consisting of perturbed sums of harmonic sinusoids. As an example, suppose there are 5 columns, then the pre-ordering strategy leads to $[1, 5, 3, 2, 4]$. Thus we have $\Pi = [e_1, e_5, e_3, e_2, e_4]$,

where e_i is a dimension ℓ column vector with all zero components except for an one at the i^{th} position.

3. As for QRD with column pivoting [21, p. 233], Π is determined during the QRD process, where $\pi_1 = e_{d_1}$ and $d_1 \in [1, \ell]$ is the index such that $\tilde{a}_{d_1}$ has the largest norm. Continuing with this column-pivoting process on the lower right submatrix yet to be triangularized, we can determine the permutation matrix Π which yields an optimum QRD column ordering strategy in the sense of preserving most energy in the upper trapezoidal submatrix. However, this Π is data-dependent and the extra cost for this pivoting may make it less desirable for some applications.

After forcing those rank-weakly quantities to be zero and preserving the most significant p-rank, we can obtain a rank-p approximate of $\tilde{A}$. These rank-weakly quantities are those entries in the factorized matrix that contribute least significantly to the matrix, or possess the smallest portion of the energy (square of Frobenius norm) of the associated matrix. For TSVD, $\tilde{\Sigma}_2$ is discarded and

$$\tilde{A}^{(p)}_{TSVD} = \tilde{U}_1 \tilde{\Sigma}_1 \tilde{V}_1^H . \tag{7.13}$$

Similarly, for TQR, the lower-right submatrix $\tilde{R}_{22}$ is discarded and

$$\tilde{A}^{(p)}_{TQR}\, \Pi = \tilde{Q}_1 [\tilde{R}_{11}\ \tilde{R}_{12}]. \tag{7.14}$$

To account for the effect due to truncation, we define the *fractional truncated F-norm* as

$$\mathcal{F}^{(p)} = 1 - \|\tilde{A}^{(p)}\|_F / \|\tilde{A}\|_F, \tag{7.15}$$

where $\|\cdot\|_F$ is the Frobenius norm given by

$$\|A\|_F = \sqrt{\sum_i \sum_j |a_{i,j}|^2} = \sqrt{\text{trace } (A^H A)}. \tag{7.16}$$

Thus we have

$$\mathcal{F}_{TSVD}^{(p)} = \sqrt{\Sigma_{j=p+1}^{\ell} \tilde{\sigma}_j^2 \,/\, \Sigma_{j=1}^{\ell} \tilde{\sigma}_j^2} \tag{7.17}$$

and

$$\mathcal{F}_{TQR}^{(p)} = 1 \;-\; \left(\frac{\|\tilde{R}_{11}\|_F^2 \;+\; \|\tilde{R}_{12}\|_F^2}{\|\tilde{R}_{11}\|_F^2 \;+\; \|\tilde{R}_{12}\|_F^2 \;+\; \|\tilde{R}_{22}\|_F^2} \right)^{1/2} . \tag{7.18}$$

While

$$0 \leq \mathcal{F}_{TSVD}^{(p)} \leq \mathcal{F}_{TQRP}^{(p)} \leq 1 \tag{7.19}$$

is valid analytically[21], from extensive computations we also observed the relationships among truncated QR methods to satisfy

$$\mathcal{F}_{TQRP}^{(p)} \leq \mathcal{F}_{TQRR}^{(p)} \leq \mathcal{F}_{TQR}^{(p)} \leq 1. \tag{7.20}$$

Therefore from the point of view of preserving the Frobenius norm(square root of energy) of a matrix, SVD provides the optimum truncation, with TQRP being next, while TQR and TQRR truncate even more.

7.4 Perturbation of Matrix Decomposition and Perfect Truncation

In this section, we examine the effects of the noise matrix E on the decompositions of $\tilde{A} = A + E$, and associated sufficient conditions for perfect truncations based on truncated QR and SVD methods. For simplicity, we only consider the QRD method without pivoting. Since the noise-free data matrix A has rank p, we have

$$A \;=\; [Q_1 \;\vdots\; Q_2] \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \tag{7.21}$$

$$\;=\; [Q_1 R_{11} \;\vdots\; Q_1 R_{12}] \,. \tag{7.22}$$

The noise-perturbed data matrix $\tilde{A}$ can be written as

$$\tilde{A} = A + E \tag{7.23}$$

$$= [Q_1 \vdots Q_2] \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} + [Q_1 \vdots Q_2] \begin{bmatrix} E_{11}^* & E_{12}^* \\ E_{21}^* & E_{22}^* \end{bmatrix} \tag{7.24}$$

$$= [Q_1 \vdots Q_2] \begin{bmatrix} R_{11} + E_{11}^* & R_{12} + E_{12}^* \\ E_{21}^* & E_{22}^* \end{bmatrix} \tag{7.25}$$

$$= [Q_1 \vdots Q_2] \begin{bmatrix} Q_{11}^* & Q_{12}^* \\ Q_{21}^* & Q_{22}^* \end{bmatrix} \begin{bmatrix} \tilde{R}_{11} & \tilde{R}_{12} \\ 0 & \tilde{R}_{22} \end{bmatrix} \tag{7.26}$$

$$= [\tilde{Q}_1 \vdots \tilde{Q}_2] \begin{bmatrix} \tilde{R}_{11} & \tilde{R}_{12} \\ 0 & \tilde{R}_{22} \end{bmatrix}, \tag{7.27}$$

where

$$\tilde{Q}_1 = Q_1 Q_{11}^* + Q_2 Q_{21}^*, \tag{7.28}$$

$$\tilde{Q}_2 = Q_1 Q_{12}^* + Q_2 Q_{22}^*, \tag{7.29}$$

and

$$\begin{bmatrix} Q_{11}^* & Q_{12}^* \\ Q_{21}^* & Q_{22}^* \end{bmatrix} \begin{bmatrix} \tilde{R}_{11} & \tilde{R}_{12} \\ 0 & \tilde{R}_{22} \end{bmatrix} = \text{QRD} \left\{ \begin{bmatrix} R_{11} + E_{11}^* & R_{12} + E_{12}^* \\ E_{21}^* & E_{22}^* \end{bmatrix} \right\}, \tag{7.30}$$

with QRD$\{\cdot\}$ denoting the QR decomposition operator applied to the matrix in $\{\cdot\}$.

After truncation, we have

$$\tilde{A}_{QR}^{(p)} = [\tilde{Q}_1 \vdots \tilde{Q}_2] \begin{bmatrix} \tilde{R}_{11} & \tilde{R}_{12} \\ 0 & 0 \end{bmatrix} \tag{7.31}$$

$$= [\tilde{Q}_1 \tilde{R}_{11} \vdots \tilde{Q}_1 \tilde{R}_{12}]. \tag{7.32}$$

Let us define a *perfect truncation* as the case when

$$\tilde{A}_{QR}^{(p)} = A \,. \tag{7.33}$$

That is, the original noise-free data matrix A can be fully recovered after truncating the rank-weakly part of the factorization of the noise-corrupted matrix $\tilde{A}$. A sufficient condition for (7.33) (i.e., (7.32) to be equal to (7.22)), is for the noise matrix E to satisfy $E_{11}^* = E_{12}^* = E_{21}^* = 0$, where

$$E = [Q_1 \vdots Q_2] \begin{bmatrix} E_{11}^* & E_{12}^* \\ E_{21}^* & E_{22}^* \end{bmatrix} \tag{7.34}$$

$$= [Q_1 \vdots Q_2] \begin{bmatrix} 0 & 0 \\ 0 & E_{22}^* \end{bmatrix} \tag{7.35}$$

$$= [0 \vdots Q_2 E_{22}^*] \,. \tag{7.36}$$

(7.36) reveals that the first p columns of $\tilde{A}$ are noise-free and the rest of the columns all reside in the orthogonal-complement column space of the data matrix A.

Similarly for SVD, we have

$$\tilde{A} = A + E \tag{7.37}$$

$$= [U_1 \vdots U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^H \\ V_2^H \end{bmatrix} + [U_1 \vdots U_2] \begin{bmatrix} \mathcal{E}_{11}^* & \mathcal{E}_{12}^* \\ \mathcal{E}_{21}^* & \mathcal{E}_{22}^* \end{bmatrix} \begin{bmatrix} V_1^H \\ V_2^H \end{bmatrix} \tag{7.38}$$

$$= [U_1 \vdots U_2] \begin{bmatrix} \Sigma_1 + \mathcal{E}_{11}^* & \mathcal{E}_{12}^* \\ \mathcal{E}_{21}^* & \mathcal{E}_{22}^* \end{bmatrix} \begin{bmatrix} V_1^H \\ V_2^H \end{bmatrix} \tag{7.39}$$

$$= [U_1 \vdots U_2] \begin{bmatrix} U_{11}^* & U_{12}^* \\ U_{21}^* & U_{22}^* \end{bmatrix} \begin{bmatrix} \tilde{\Sigma}_1 & 0 \\ 0 & \tilde{\Sigma}_2 \end{bmatrix} \begin{bmatrix} V_{11}^{*H} & V_{21}^{*H} \\ V_{12}^{*H} & V_{22}^{*H} \end{bmatrix} \begin{bmatrix} V_1^H \\ V_2^H \end{bmatrix} \tag{7.40}$$

$$= [\tilde{U}_1 \vdots \tilde{U}_2] \begin{bmatrix} \tilde{\Sigma}_1 & 0 \\ 0 & \tilde{\Sigma}_2 \end{bmatrix} \begin{bmatrix} \tilde{V}_1^H \\ \tilde{V}_2^H \end{bmatrix}, \tag{7.41}$$

where

$$[\tilde{U}_1 \vdots \tilde{U}_2] = [U_1 \vdots U_2] \begin{bmatrix} U_{11}^* & U_{12}^* \\ U_{21}^* & U_{22}^* \end{bmatrix}, \tag{7.42}$$

$$[\tilde{V}_1 \vdots \tilde{V}_2] = [V_1 \vdots V_2] \begin{bmatrix} V_{11}^* & V_{12}^* \\ V_{21}^* & V_{22}^* \end{bmatrix}, \tag{7.43}$$

and

$$\begin{bmatrix} U_{11}^* & U_{12}^* \\ U_{21}^* & U_{22}^* \end{bmatrix} \begin{bmatrix} \tilde{\Sigma}_1 & 0 \\ 0 & \tilde{\Sigma}_2 \end{bmatrix} \begin{bmatrix} V_{11}^{*H} & V_{21}^{*H} \\ V_{12}^{*H} & V_{22}^{*H} \end{bmatrix} = \text{SVD} \left\{ \begin{bmatrix} \Sigma_1 + \mathcal{E}_{11}^* & \mathcal{E}_{12}^* \\ \mathcal{E}_{21}^* & \mathcal{E}_{22}^* \end{bmatrix} \right\}, \tag{7.44}$$

with SVD$\{\cdot\}$ denoting the SVD operator applied to the matrix in $\{\cdot\}$.

We also notice that to achieve perfect truncation for the SVD method, we need

$$\tilde{A}_{SVD}^{(p)} = \tilde{U}_1 \tilde{\Sigma}_1 \tilde{V}_1^H = A\,. \tag{7.45}$$

A sufficient condition for (7.45) is $\mathcal{E}_{11}^* = \mathcal{E}_{12}^* = \mathcal{E}_{21}^* = 0$ (i.e., $E = U_2 \mathcal{E}_{22}^* V_2^H$) and the largest singular value of E is less than the smallest singular value of Σ_1, or equivalently, the row and column spaces of E must be orthogonal to those of A, and there exists a gap between the singular values of A and E such that even the weakest signal subspace will not be corrupted by any erroneous noise space.

7.5 Minimum-norm Solutions

After truncation, the FBLP LS problem becomes rank-deficient, hence the minimum-norm LS solution is desired in order to suppress those spurious harmonics in the

pseudo-spectrum. The following lemma gives the minimum-norm solution for a rank-deficient LS problem.

Lemma 4 *(Minimum-norm solution)*
For an underdetermined LS problem,

$$B\mathbf{y} = \mathbf{c}, \qquad (B \in \mathcal{C}^{p\times\ell}, \mathbf{c} \in \mathcal{C}^{p}, p \leq \ell), \tag{7.46}$$

with rank(B)=p. The minimum-norm solution $\hat{\mathbf{y}}$ *is in the row space of* B.

Proof.Suppose B *has full row rank and* $\hat{\mathbf{y}}$ *belongs to the row space of* B, *i.e.,* $\hat{\mathbf{y}} = B^H\mathbf{z}$, *then there exists a unique solution* $\hat{\mathbf{y}} = B^H\mathbf{z}$ *since* $\mathbf{z} = (BB^H)^{-1}\mathbf{c}$ *is unique. Suppose there are other solutions of the form* $\mathbf{y} = \hat{\mathbf{y}} + \mathbf{y}^{\perp}$, *where* $\mathbf{y}^{\perp}$ *lies in the space perpendicular to the row space of* B, *i.e.,* $B\mathbf{y}^{\perp} = 0$. *Then it is obvious that*

$$\|\mathbf{y}\|^2 = \|\hat{\mathbf{y}}\|^2 + \|\mathbf{y}^{\perp}\|^2 \geq \|\hat{\mathbf{y}}\|^2. \tag{7.47}$$

So

$$\hat{\mathbf{y}} = arg\ \min\{\|\mathbf{y}\| \mid B\mathbf{y} = \mathbf{c}\}.$$

■

For simplicity, let Π be an identity in the QRD case. To avoid the cumbersome normal-equation-like computation of the minimum norm solution, $B^H(BB^H)^{-1}\mathbf{c}$, with $B = [\tilde{R}_{11}\ \tilde{R}_{12}]$ and $\mathbf{c} = \tilde{Q}_1^H\tilde{b}$, we can firstly perform a *backward* QR decomposition on the conjugate transpose of $[\tilde{R}_{11}\ \tilde{R}_{12}]$ to obtain the same solution as given in the above lemma without fill-in's(the newly introduced nonzero entities while performing QRD). Ill-conditioning will not occur because the diagonal elements are sufficiently large in the trapezoidal truncated matrix. By doing backward

modified Gram-Schmidt orthogonalization[21] procedure, we can have

$$B^H = \begin{bmatrix} \tilde{R}_{11}^H \\ \tilde{R}_{12}^H \end{bmatrix} = TL, \qquad (T \in C^{\ell \times p}; L \in C^{p \times p}), \tag{7.48}$$

with L being lower triangular. Here the columns of $T = [\mathbf{t}_1, \ldots, \mathbf{t}_p]$ (satisfying $T^H T = I$) are computed in a backward manner, i.e, from $\mathbf{t}_p$ to $\mathbf{t}_1$, and the diagonal elements of L are computed from the lower right toward the upper left. We note that the minimum solution for the truncated QR method is given by

$$\begin{aligned} g_{TQR}^p &= B^H (BB^H)^{-1} \mathbf{c} \\ &= TL(L^H T^H TL)^{-1} \mathbf{c} \\ &= T(L^{-H} \mathbf{c}), \end{aligned} \tag{7.49}$$

where it is also noted that a backward substitution is required in the computation of $L^{-H}\mathbf{c}$.

Therefore the minimum-norm LS solution can be obtained via the following procedure:

1. Do QRD on the augmented matrix $[\tilde{A} \vdots \tilde{b}]$ (with possibly column pivoting);
2. Take the transpose of the trapezoidal upper triangular matrix. Do backward QRD (save the orthogonal matrix T) ;
3. Apply backward substitution on the transpose of lower-triangular matrix obtained in step (2) and the updated right-hand-side in step (1), followed by (7.49).

According to this lemma, a minimum-norm solution vector $g^{(p)}$ must lie in the row space of the rank-reduced matrix $\tilde{A}^{(p)}$, namely, the row space of $\tilde{V}_1^H$ or

$[\tilde{R}_{11}\ \tilde{R}_{12}]$. For TSVD it is given by

$$g_{TSVD}^{(p)} = \tilde{V}_1 \tilde{\Sigma}_1^{-1} \tilde{U}_1^H \tilde{b}, \tag{7.50}$$

or

$$g_{TSVD}^{(p)} = \sum_{j=1}^{p} \frac{\tilde{u}_j^H \tilde{b}}{\tilde{\sigma}_j} \tilde{v}_j \ . \tag{7.51}$$

To obtain $g_{TQR}^{(p)}$, we can perform QRD on the right of the trapezoidal upper-triangular matrix in (7.14) to zero out $\tilde{R}_{12}$ and also obtain the orthonormal row space, $\tilde{T}^H$, of $[\tilde{R}_{11}\ \tilde{R}_{12}]$. That is

$$\tilde{A}_{TQR}^{(p)}\,\Pi = \tilde{Q}_1[\tilde{R}_{11}\ \tilde{R}_{12}] \ = \tilde{Q}_1 \tilde{L}^H \tilde{T}^H \ , \tag{7.52}$$

where $\tilde{T} = [\tilde{t}_1, \ldots, \tilde{t}_p] \in C^{\ell \times p}$ has orthonormal columns and $\tilde{L}^H \in C^{p \times p}$ is an upper-triangular matrix. This is sometimes called a *complete orthogonal factorization* [21, p. 236], and we can consider it as a two-sided direct unitary transformations on a rank-deficient matrix to compress all the energy of a matrix into a *square upper-triangular* matrix. This resembles the SVD method where two-sided iterative unitary transformations are applied to reduce a matrix into a *diagonal* matrix. Then from (7.49) and (7.52) the minimum-norm solution follows by

$$g_{TQR}^{(p)} \ = \ \Pi \begin{bmatrix} \tilde{R}_{11}^H \\ \tilde{R}_{12}^H \end{bmatrix} (\tilde{R}_{11}\tilde{R}_{11}^H + \tilde{R}_{12}\tilde{R}_{12}^H)^{-1} \tilde{Q}_1^H \tilde{b} \tag{7.53}$$

$$= \ \Pi \tilde{T} \tilde{L}^{-H} \tilde{Q}_1^H \tilde{b} \ . \tag{7.54}$$

It is noted that if no truncation is performed at all and $\tilde{A}$ has full column rank ℓ, then the LS solution from the FBLP model is either obtained from SVD as

$$\tilde{g} \ = \ \tilde{V}_1 \tilde{\Sigma}_1^{-1} \tilde{U}_1^H \tilde{b} \ + \ \tilde{V}_2 \tilde{\Sigma}_2^{-1} \tilde{U}_2^H \tilde{b}$$

$$= \sum_{j=1}^{p} \frac{\tilde{u}_j^H \tilde{b}}{\tilde{\sigma}_j} \tilde{v}_j + \sum_{j=p+1}^{\ell} \frac{\tilde{u}_j^H \tilde{b}}{\tilde{\sigma}_j} \tilde{v}_j$$

$$= g_{TSVD}^{(p)} + \sum_{j=p+1}^{\ell} \frac{\tilde{u}_j^H \tilde{b}}{\tilde{\sigma}_j} \tilde{v}_j \,, \tag{7.55}$$

or from QRD as

$$\begin{aligned} \tilde{g} &= \Pi \begin{bmatrix} \tilde{R}_{11} & \tilde{R}_{12} \\ 0 & \tilde{R}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{Q}_1^H \tilde{b} \\ \tilde{Q}_2^H \tilde{b} \end{bmatrix} \\ &= \Pi \begin{bmatrix} \tilde{R}_{11}^{-1} & -\tilde{R}_{11}^{-1} \tilde{R}_{12} \tilde{R}_{22}^{-1} \\ 0 & \tilde{R}_{22}^{-1} \end{bmatrix} \begin{bmatrix} \tilde{Q}_1^H \tilde{b} \\ \tilde{Q}_2^H \tilde{b} \end{bmatrix} \\ &= \Pi \begin{bmatrix} \tilde{R}_{11}^{-1} \tilde{Q}_1^H \tilde{b} - \tilde{R}_{11}^{-1} \tilde{R}_{12} \tilde{R}_{22}^{-1} \tilde{Q}_2^H \tilde{b} \\ \tilde{R}_{22}^{-1} \tilde{Q}_2^H \tilde{b} \end{bmatrix} \end{aligned} \tag{7.56}$$

Because $\tilde{\Sigma}_2$ and $\tilde{R}_{22}$ are both nearly zero under a high SNR condition, slight variations on them will cause significant perturbations in the solution vector $\tilde{g}$ and hence leads to many spurious frequencies in the pseudo-spectrum. Now it becomes clear why one truncates these rank-weakly quantities to remedy these ill-conditions from the view point of numerical stability and also prefilters some stray noise in an attempt to guard against possible contaminations in the pseudo-spectrum.

For many problems, the conservative approach of over-modeling (i.e., $\ell >> p$) is preferred [59, 49] to taking $\ell \gtrsim p$, since we can later *truncate* some noises that reside in the null space which is orthogonal to the signal space. The advantage of over-modeling is to provide some extra dimensions to *trap* the stray noises and then remove them by truncation. This is an effective way of enhancing the SNR. However, there is always the danger that some signal has been mistakably truncated in low SNR cases where ambiguous changes in the truncated F-norm

is possible. On the other hand, spurious frequencies are still very likely to occur when there is insufficient truncation of the rank of the data matrix.

7.6 Truncated Normal Equation Approach

Up to now, the matrix decompositions are performed with the direct data, hence the condition number of the problem is not increased. It is interesting to see that there also exists a truncated normal equation (TNE) solution which computes the minimum norm solution for the covariance data. We show that essentially this TNE method is mathematically equivalent to the TQR method except for increased roundoff errors under finite precision computations.

An *un*truncated least-squares solution for (7.6) using the normal equation approach is to solve

$$\tilde{A}^H \tilde{A} g = \tilde{A}^H \tilde{b}. \tag{7.57}$$

If we rewrite $\tilde{A}$ as $[\tilde{A}_1 \vdots \tilde{A}_2] = [\tilde{A}_1 \vdots \tilde{A}_1 \tilde{F} + N]$, where $\tilde{F} \in \mathcal{C}^{p \times (\ell - p)}$ represents the projection of $\tilde{A}_2$ onto $\tilde{A}_1$ and $N \in \mathcal{C}^{2(n-\ell) \times (\ell - p)}$ represents the remaining residual. Therefore, columns of $\tilde{A}_1 \in \mathcal{C}^{2(n-\ell) \times p}$ and $N \in \mathcal{C}^{2(n-\ell) \times (\ell - p)}$ are orthogonal to each other. Under high SNR cases, N is close to a zero matrix, and in the extreme case when the noise is absent, N is equal to zero. Then (7.57) can be rewritten as

$$\begin{bmatrix} \tilde{A}_1^H \tilde{A}_1 & \tilde{A}_1^H \tilde{A}_2 \\ \tilde{A}_2^H \tilde{A}_1 & \tilde{A}_2^H \tilde{A}_2 \end{bmatrix} g = \begin{bmatrix} \tilde{A}_1^H \tilde{b} \\ \tilde{A}_2^H \tilde{b} \end{bmatrix} \tag{7.58}$$

or

$$\begin{bmatrix} \tilde{A}_1^H \tilde{A}_1 & \tilde{A}_1^H \tilde{A}_2 \\ \tilde{F}^H \tilde{A}_1^H \tilde{A}_1 + N^H \tilde{A}_1^H & \tilde{F}^H \tilde{A}_1^H \tilde{A}_2 + N^H \tilde{A}_2^H \end{bmatrix} g = \begin{bmatrix} \tilde{A}_1^H \tilde{b} \\ \tilde{A}_2^H \tilde{b} \end{bmatrix}. \tag{7.59}$$

We can see that as N goes to zero, the bottom $(\ell - p)$ equations in (7.59) become redundant, because they are merely equal to $\tilde{F}^H$ times the top p equations. Therefore, a truncated normal equation (TNE) can be defined by discarding the almost-redundant bottom $(\ell - p)$ equations in (7.59).

Similarly, a minimum norm solution (from the above Lemma) follows by

$$g_{TNE}^{(p)} = \begin{bmatrix} \tilde{A}_1^H \tilde{A}_1 \\ \tilde{A}_2^H \tilde{A}_1 \end{bmatrix} [\tilde{A}_1^H \tilde{A}_1 \tilde{A}_1^H \tilde{A}_1 + \tilde{A}_1^H \tilde{A}_2 \tilde{A}_2^H \tilde{A}_1]^{-1} \cdot (\tilde{A}_1^H \tilde{b}). \tag{7.60}$$

It remains to show that TQR and TNE methods in (7.53) and (7.60) are mathematically equivalent. To this end, we can replace $\tilde{A}_1$ by $\tilde{Q}_1 \tilde{R}_{11}$ and $\tilde{A}_2$ by $\tilde{Q}_1 \tilde{R}_{12} + \tilde{Q}_2 R_{22}$, hence we have $\tilde{A}_1^H \tilde{A}_1 = \tilde{R}_{11}^H \tilde{R}_{11}$ and $\tilde{A}_2^H \tilde{A}_1 = \tilde{R}_{12}^H \tilde{R}_{11}$. After some manipulations, (7.60) can be written as

$$g_{TNE}^{(p)} = \begin{bmatrix} \tilde{R}_{11}^H \\ \tilde{R}_{12}^H \end{bmatrix} \tilde{R}_{11}^H [(\tilde{R}_{11} \tilde{R}_{11}^H + \tilde{R}_{12} \tilde{R}_{12}^H) \tilde{R}_{11}]^{-1} \cdot (\tilde{Q}_1^H \tilde{b}) \,, \tag{7.61}$$

which is equal to $g_{TQR}^{(p)}$ in (7.53) where Π is an identity.

We must note that although TQR and TNE methods are mathematically equivalent, the former is much favorable under finite precision computation. The rationale is that the TQR method always deals with the direct data, while the TNE method works on the covariance data where the dynamic range of data is inevitably squared. Therefore the TNE method is more susceptible to roundoff errors.

7.7 Simulation Results

Finally, we present various computer simulations based on the following model. Let $\tilde{x}_i = cos(2\pi f_1 i) + cos(2\pi f_2 i) + w_i, i = 1, 2, \cdots, 48$, with $f_1 = .125, f_2 =$

.135, $\ell = 36$ and $\{w_i\}$ is a white Gaussian random sequence. The frequencies are determined by the phases (from 0 to π) of complex roots closest to the unit circle. For TQRR, we pre-permute the columns of the FBLP matrix in the order of: {1 36 18 9 27 5 ... } as suggested by [51]. We will consider three quantities on the evaluation of the performances. The first one is the frequency bias, which is defined as the difference of the true and estimated frequencies. The standard deviation of the estimated frequency is our second performance measure. The last one is the distance of the third principal root to the unit circle. Here we represent the principal root as those roots that are close to the unit circle. Ideally we should have only 2 principal roots (due to the two sinusoids if we only consider those roots with phases within $[0, \pi]$), falling exactly on the unit circle, while all the others are inside the unit circle. Under moderate SNR conditions, a third principal root may be mistaken as a third candidate harmonic, if its distance to the unit circle is approximately equal to the first 2 principal roots. An even worse condition may occur when the true harmonic falls behind (i.e., further away from the unit circle) an spurious harmonic due to the random noise. This is similar to the case that a noise subspace enters the signal subspace. Therefore, a good separation of the 3rd harmonic from the unit circle not only decreases the chance of mistaking false harmonics but also increases our confidence on estimating the true number of harmonics.

Two classes of comparisons will be considered in the following curves. The first one is to compare these truncation methods under various SNR from 0 to 50 dB. The second is to observe the asymptotic performance by fixing the order $\ell = 36$, and increasing the number of observed data samples. 100 independent simulations are used to obtain the statistical means and standard deviations.

Fig. 7.2 gives the average fractional truncated Frobenius norms of (7.16) versus SNR when we preserve only the four most significant ranks of the FBLP matrix for the five different methods. This confirms their relationships in (7.19) and (7.20) and also shows that the truncated energy decreases monotonically as SNR increases. Fig. 7.3 and 7.4 show the averages of the frequency biases for the two harmonic frequencies. We define the average frequency bias as $E(\tilde{f}_k) - f_k$, $k = 1, 2$, where $E(\tilde{f}_k)$ is the ensemble average of $\tilde{f}_k$, which is the estimated frequency for f_k. Fig. 7.5 and 7.6 show the standard deviations of $\tilde{f}_1$ and $\tilde{f}_2$. We can see that TQRP competes quite well with TSVD, while TQRR performs slightly worse than TQRP but better than TQR without pivoting. Fig. 7.7 and 7.8 show the distances to the unit circle of the first 2 dominant roots that are closest to the unit circle. Fig. 7.9 gives the distance to the unit circle of the third closest root. Since this third root is a false one, it should be far away from the unit circle to allow for easy determination of number of harmonics.

If we fix the SNR= 10 dB and the order of the FBLP model to be $\ell = 36$, as more data are collected, the ill effect due to noise should be asymptotically smoothed out. Fig. 7.10 shows the combined average frequency bias (defined as the sum of the absolute values of the biases for f_1 and f_2) verses the number of data samples. Fig. 7.11 shows the curves of various combined standard deviations of the estimated frequencies which is defined as the square root of the sum of squares of the standard deviations of each frequency estimate. Fig. 7.12 depicts the mean distances to the unit circle of the false harmonics. From Fig. 7.10 to 7.12, it is clear that under moderate SNR conditions, the performances of TQRP closely follow that of TSVD.

7.8 Conclusions

While a myriad of researches have been focused on SVD and eigen-decomposition analysis of narrowly spaced harmonic frequency estimations [59, 49, 29], very few have been directed towards the QRD approaches. Owing to the iterative massive computations and the difficulty encountered in updating the decompositions [11] when new data are acquired under time-varying conditions, these SVD and eigen-based approaches are ill suited for real time applications. It is well known [21] that QRD is numerically as stable as SVD, requires much less computational cost, easy to update(and/or downdate), and amenable to VLSI implementations. The slightly degraded performance for these truncated QR methods is greatly compensated by all the benefits mentioned above. As well known, the performance of the LS method is usually much worse than those of the QR and SVD methods.

Table 1 summarizes the comparisons among different truncation methods. We conclude that TQR is the simplest and can be performed easily in a real time updating, but may suffer significant degradation. TQRP provides almost the same performance as SVD, but is not easy to implement in real time processing in that the difficult column reshuffling is required while performing QRD with pivoting. TQRR provides a good compromise between the above two and can also be implemented for systolic array processing. The LS method is simple to implement and update but has a poor frequency estimation capability.

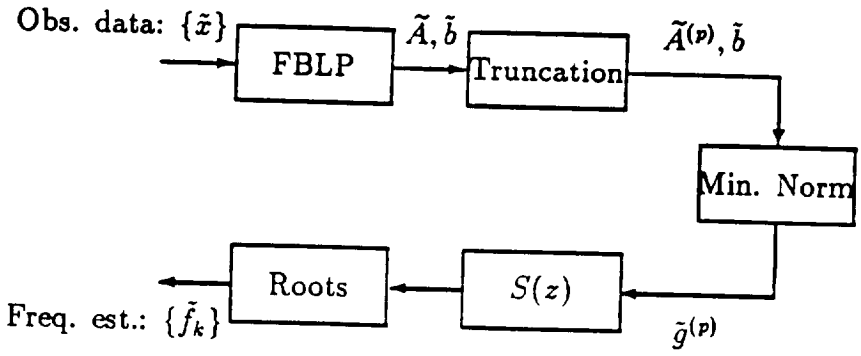


Figure 7.1: Block diagram for sinusoidal frequency estimation based on the FBLP model.

Table 7.1: Comparisons of truncated least-squares methods.

	Freq. est.	Comput. cost	VLSI	updating
TSVD	excellent	very high	complex	difficult
TQRP	very good	medium	medium	medium
TQRR	good	fair	fair	easy
TQR	fair	fair	fair	easy
LS	poor	low	low	easy

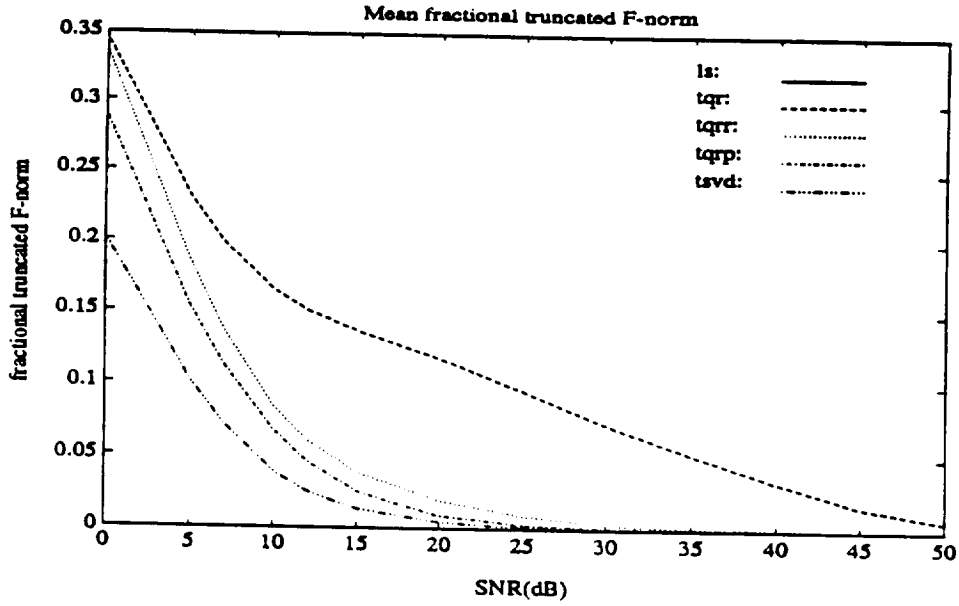


Figure 7.2: Average fractional truncated Frobenius norms.

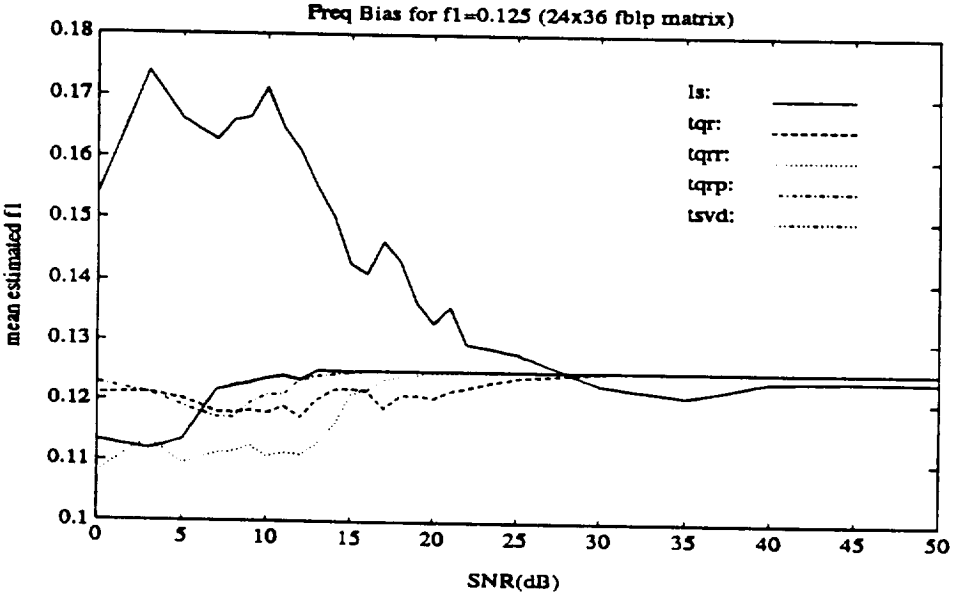


Figure 7.3: Mean frequency estimates for $f_1 = .125$ using a 24×36 FBLP matrix.

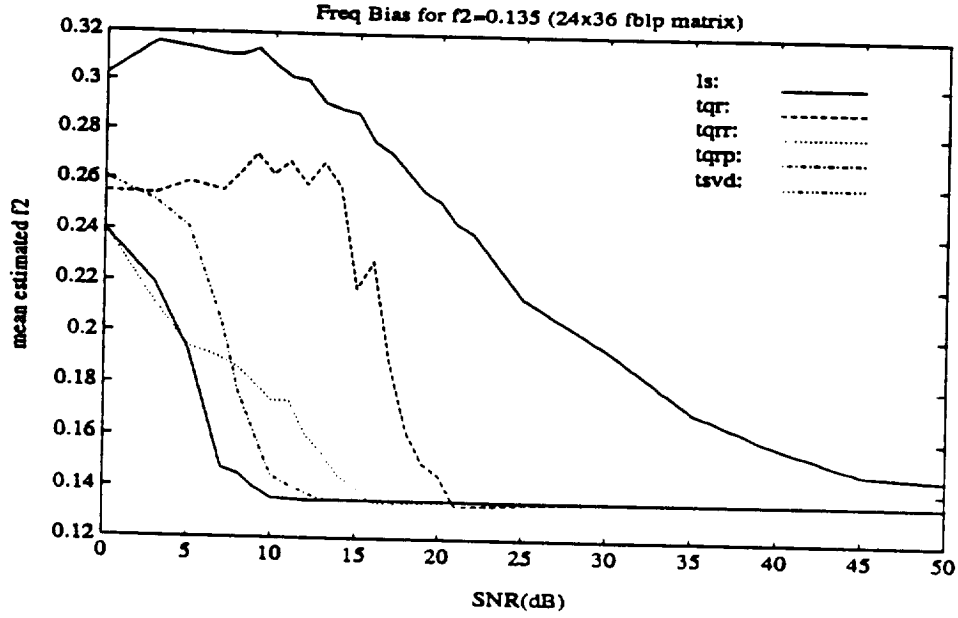


Figure 7.4: Mean frequency estimates for $f_2 = .135$ using a 24×36 FBLP matrix.

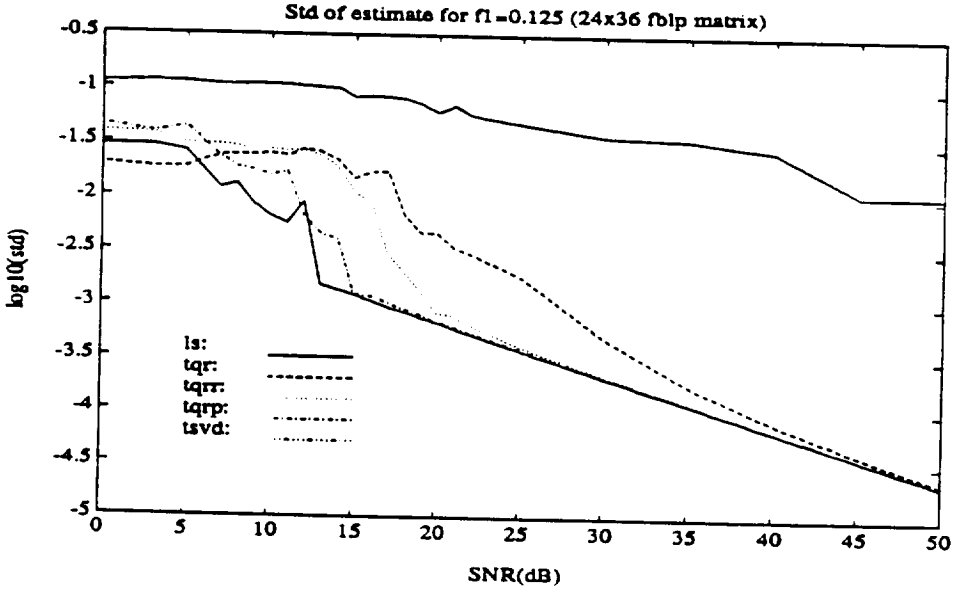


Figure 7.5: Standard deviations for estimating $f_1 = .125$ using a 24×36 FBLP matrix.

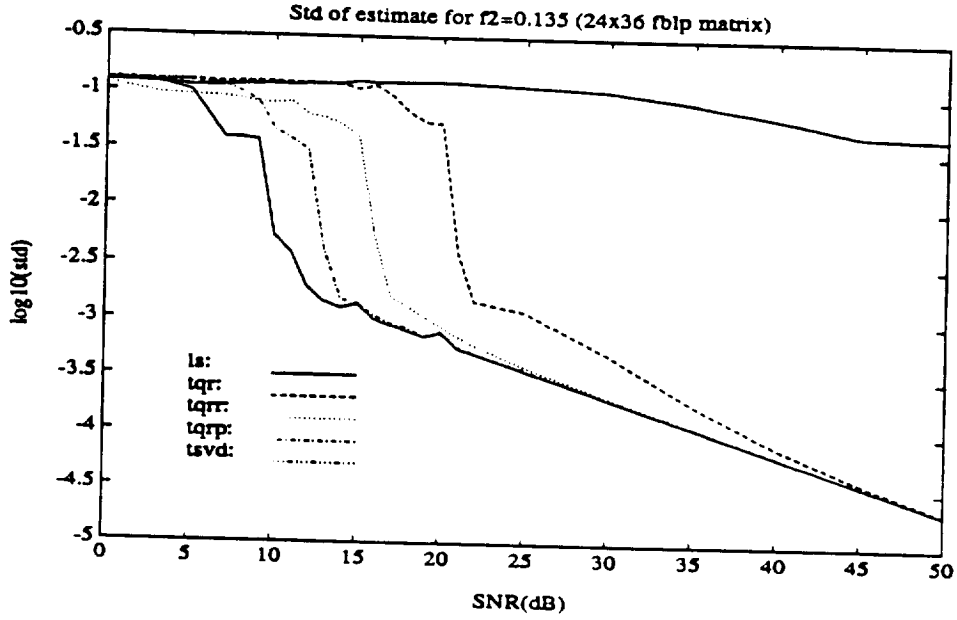


Figure 7.6: Standard deviations for estimating $f_2 = .135$ using a 24×36 FBLP matrix.

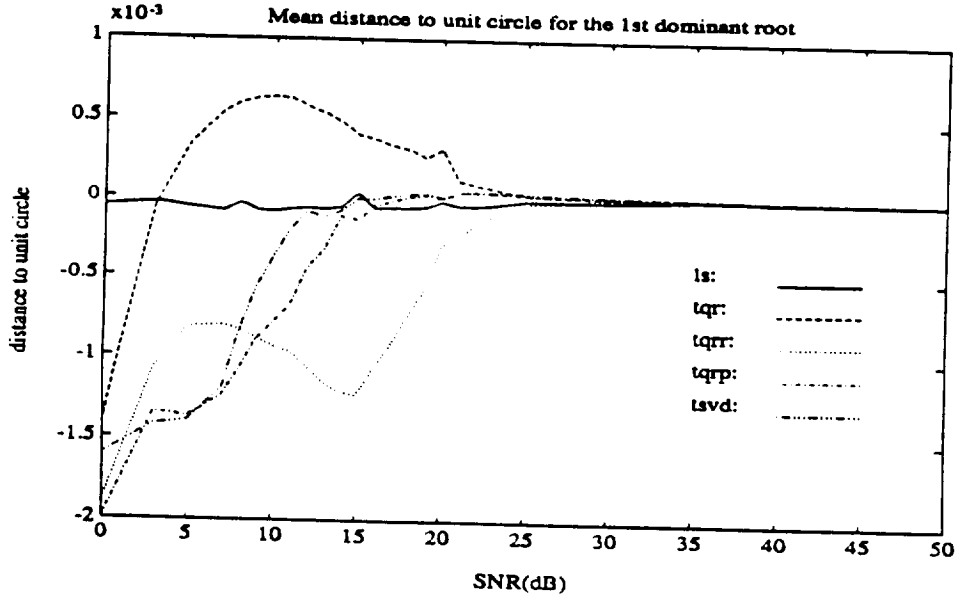


Figure 7.7: Mean distances to unit circle of the roots of the 1st harmonic freq. estimator vs. SNR.

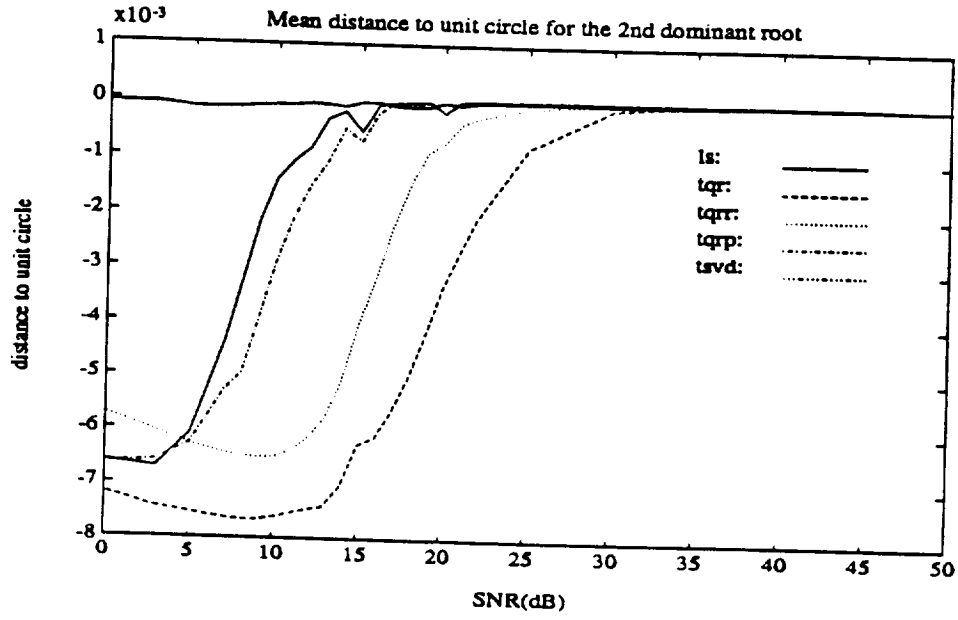


Figure 7.8: Mean distances to unit circle of the roots of the 2nd harmonic freq. estimator vs. SNR.

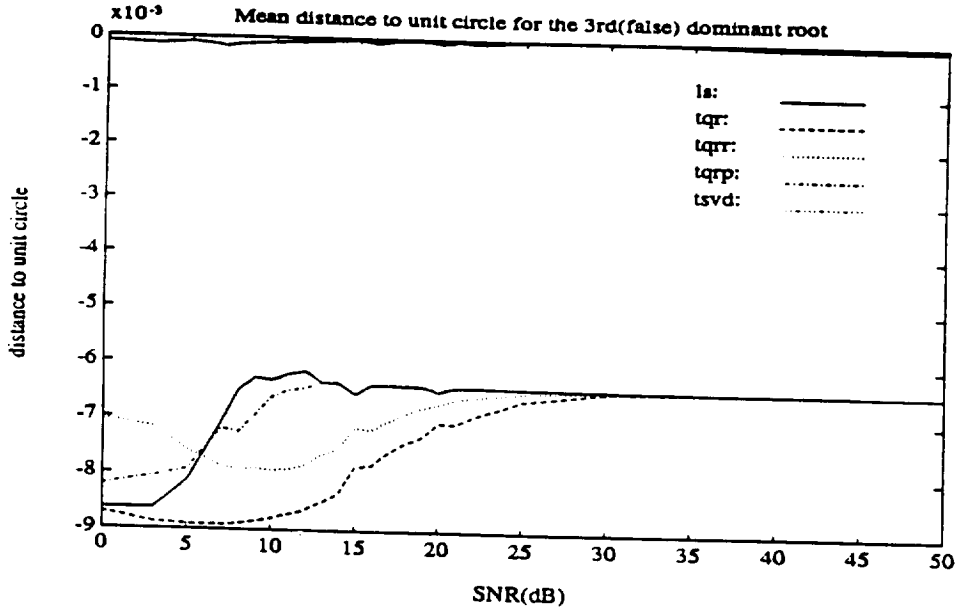


Figure 7.9: Mean distances to unit circle of the roots of the 3rd (false) harmonic freq. estimator vs. SNR

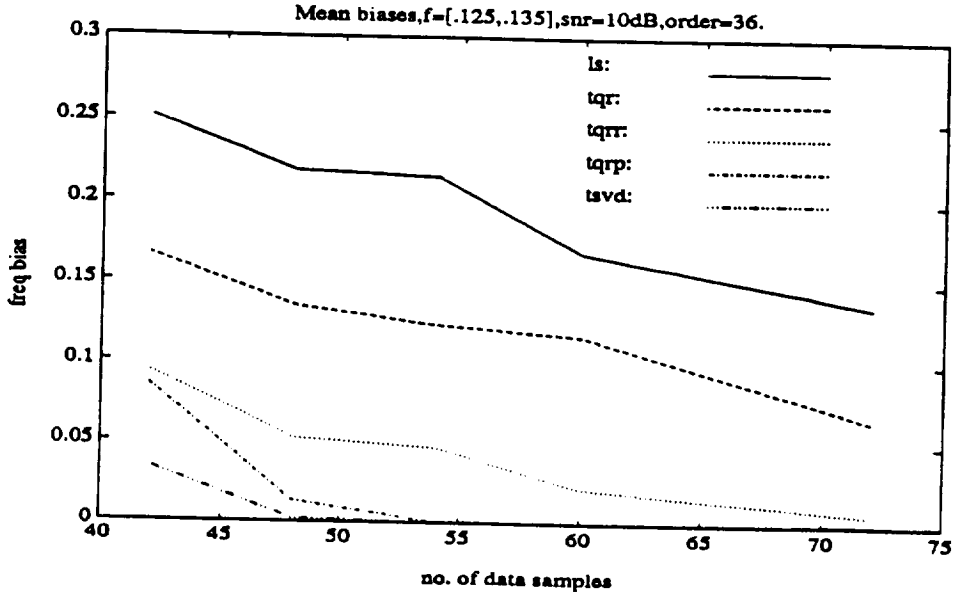


Figure 7.10: Mean freq bias vs. no. of data samples for $f = \{.125, .135\}$, SNR=10dB, and order=36.

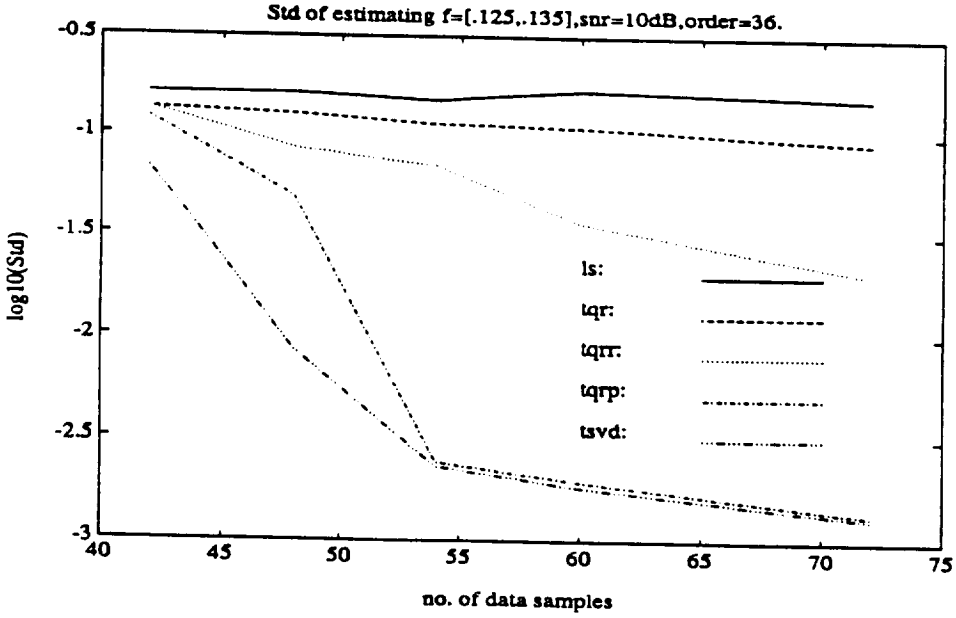


Figure 7.11: Standard deviations of estimates vs. no. of data samples for $f = \{.125, .135\}$, SNR=10dB, and order=36.

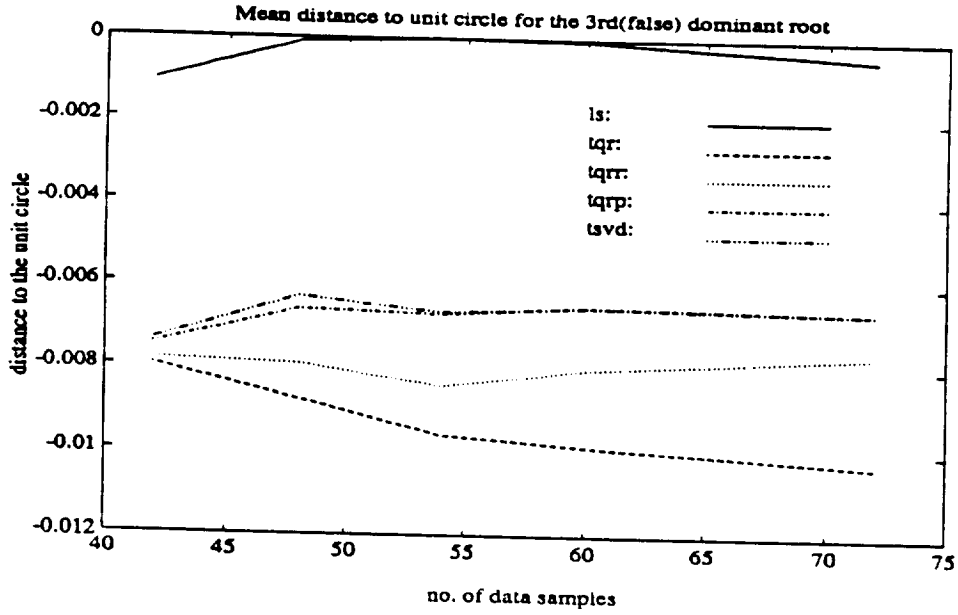


Figure 7.12: Mean distances to unit circle of the roots of the 3rd (false) harmonic freq. estimator vs. no. of data samples.

Chapter 8

Conclusions

QRD-based methods for RLS estimations are well known to be useful and effective in adaptive signal processing. A time-recursive formulation of RLS filtering using block QRD is given for efficient up/down-dating operations. To monitoring the whole residual vector a recursive formula is derived. Based on this formula, a back substitution for obtaining the optimum weight vector can be bypassed.

Existing methods of performing RLS estimation either employ exponentially weighted or fixed-window schemes under non-stationary conditions. Both are less capable in rejecting the ill effects due to temporary noise spikes. A residual-based selective window for robust RLS estimation is proposed. Computer simulations show that its abilities in tracking and reducing the bias of parameter estimation due to spurious noise spikes are superior to those of the existing methods. This new method only requires one back substitution to estimate the weight vector, while a conventional method requires two such operations.

Various up/downdating algorithms and systolic architectures are proposed

for block-type RLS estimations, which include Gram-Schmidt pseudo orthogonalization methods, and hyperbolic Householder transformations. Conventional Gram-Schmidt methods (modified or not) are reformulated such that systolic processing is possible and the computational flops are also greatly reduced.

When the block-size is reduced to be one, all vector processing becomes scalar processing. A dual-state systolic structure using planar (Givens) and hyperbolic rotations is proposed to perform joint up/down-dating operations. To further reduce the complexity and increase the throughput rate, square-root-free Cordic cells are proposed to mimic the broadcasting along the rows in the systolic triarray.

Most of the rotation-based methods require explicit square-root computations, which are undesirable from the practical VLSI circuit design point of view. A unified square-root-free rank-1 up/downdating approach is proposed. This is the first effort to establish the basic understanding toward more than ten known square-root-free QRD algorithms, from which the basic criterion is seen to be simple. This unified approach also provides a fundamental framework for the square-root-free RLS algorithms which are essential for practical VLSI implementations.

Three truncated QR methods are proposed to estimate closed spaced frequencies from limited amount of data samples. They include truncated QR methods without column pivoting, with column pivoting, and with re-ordered columns. Detailed comparisons of these methods to truncated SVD method have been given.

Table 8.1 summaries the author's contributions and previous authors' works. Matlab programs have been written for all of the algorithms addressed in this

Table 8.1: List of previously known and new results

Item	*Previous known results*	*New Results*
Res. monit.	Rader('86):two passes	one pass
Syst. RLS Filt. w/ Forget. Fac.	Gentleman/Kung('81): Scalar-valued Givens	Block-type, Householder, MGS
Syst. RLS Res. Upd. w/ Forg. Fac.	McWhirter('83): Scalar-type Givens	Block-type, Householder, MGS
Block RLS Filt. w/ Fixed-Win.	Rader/Steinhardt('86): Hyper. Householder	Hyperbolic MGS w/out sqrts
Scalar RLS Filt. w/ Fixed-Win.	Hansen/Lawson('74), Alexander, etc.('87), Bojanczryk,etc.('87): Hyper. Rot.	Dual-State Syst. Implement. & CORDIC cells
Modified GS	Rice('66): Load Imbal.	Distr. Load Bal.
Sqrt-Free	Many	Generalized Appr.
Spec. Est.	Tufts/Kumaresan('82): Truncated SVD, Comput. Extensive, Diff. to Update	Truncated QR Less comput. Easy to Update

thesis. Future work of interest includes : numerical comparisons of various square-root-free fast Givens algorithms, applications of rank-revealing QR (RRQR)[12] to FBLP-based spectral estimations.

Bibliography

[1] H. M. Ahmed, J.-M. Delosme, and M. Morf. "Highly concurrent computing structures for matrix arithmetic and signal processing". *IEEE Computer*, 15(1):65–82, Jan. 1982.

[2] S. T. Alexander, C. T. Pan, and R. J. Plemmons. "Numerical properties of a hyperbolic rotation method for windowed RLS filtering". In *IEEE ICASSP*, pages 423–426, 1987.

[3] S. T. Alexander, C.-T. Pan, and R. J. Plemmons. "Analysis of a recursive least squares hyperbolic rotation algorithm for signal processing". *Linear Algebra and its Applications*, 98:3–40, 1988.

[4] Erwin H. Bareiss. "Numerical solution of the weighted least squares problems by G-transformations". *Tech. Report 82-03-NAM-03, Dept. of Elec. Engr. and Comp. Sci., Northwestern University, Evanston, IL*, Apr. 1982.

[5] Jesse L. Barlow and Ilse C. F. Ipsen. "Scaled Givens rotations for the solution of linear least squares problems on systolic arrays". *SIAM J. Sci. Stat. Comput.*, 8(5):716–733, Sept. 1987.

[6] Richard Bartels and Linda Kaufman. "Cholesky factor updating techniques for rank 2 matrix modifications". *SIAM J. Martix Anal. Appl.*, 10(4):557–592, Oct. 1989.

[7] M. G. Bellanger. *"Adaptive digital filters and signal analysis"*. Marcel Dekker, Inc., New York and Basel, 1987.

[8] A. W. Bojanczyk, R. P. Brent, P. van Dooren, and F. R. de Hoog. "A note on downdating the Cholesky factorization". *SIAM J. Sci. Stat. Comput.*, 8(3):210–221, May 1987.

[9] A. W. Bojanczyk and F. T. Luk. "A novel MVDR beamforming algorithm". In *Proc. SPIE, Advanced algorithms and architectures for signal processing II, Vol. 826*, pages 12–16, 1987.

[10] Thomas P. Bronez and James A. Cadzow. "An algebraic approach to superresolution array processing". *IEEE Trans. on Aerospace and Electronic Systems*, AES-19(1):123–133, Jan. 1983.

[11] J. R. Bunch and C. P. Nielsen. "Updating the singular value decomposition". *Numerische Mathematik*, 31:111–129, 1978.

[12] Tony F. Chan. "Rank revealing QR-factorizations". *Lin. Alg. & Its Appl.*, pages 67–82, 1988/89.

[13] M.J. Chen and K. Yao. "Comparison of QR least-squares algorithms for systolic array processing". In *Proc. Conf. on Inform. Sciences and Systems*, pages 683–688, Mar. 1988.

[14] S.I. Chou and C.M. Rader. "Algorithm-based error detection of a cholesky factor updating systolic array using CORDIC processors". In *Proc. SPIE, Real-time Signal Processing XI*, pages 104–111, Aug. 1988.

[15] John M. Cioffi. "The fast adaptive ROTOR's RLS algorithm". *IEEE Trans. Accoust., Speech, Signal Processing*, 38(4):631–653, Apr. 1990.

[16] R. T. Compton Jr. *"Adaptive antennas: Concepts and performance"*. Prentice Hall, 1988.

[17] J. J. Dongarra, A. H. Sameh, and D. C. Sorensen. "Implementation of some concurrent algorithms for matrix factorization". *Parallel Computing*, 3:25–34, 1986.

[18] R. W. Farenbrother. *"Linear least squares computations"*. Marcel Dekker, Inc., New York and Basel, 1988.

[19] W. M. Gentleman and H. T. Kung. "Matrix triangularization by systolic array". In *Proc. SPIE, vol. 298: Real-time signal processing IV*, pages 19–26, Bellingham, Washington, 1981. Society of Photo-optical Instrumentaion Engineers.

[20] W. Morven Gentleman. "Least squares computations by Givens transformations without square roots". *J. Inst. Maths Applics*, 12:329–336, 1973.

[21] G. H. Golub and C. F. Van Loan. *"Matrix computations"*. Johns Hopkins University Press, Baltimore, MD, 2nd edition, 1989.

[22] J. Götze, B. Bruckmeier, and U. Schwiegelshohn. "VLSI-suited solution of linear systems". In *IEEE ISCAS*, pages 187–190, 1989.

[23] J. Götze and U. Schwiegelshohn. "An orthogonal method for solving systems of linear equations without square roots and with few divisions". In *Proc. ICASSP*, pages 1298–1301. IEEE, 1989.

[24] Sven Hammarling. "A note on modifications to the Givens plane rotation". *J. Inst. Maths Applics*, 13:215–218, 1974.

[25] S. Haykin. *"Adaptive filter theory"*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[26] W. Hoffmann. "Iterative algorithms for Gram-Schmidt orthogonalization". *Computing*, 41:335–348, 1989.

[27] M. L. Honig and D. G. Messerschmitt. *"Adaptive filters"*. Kluwer Academic Publishers, 1984.

[28] L. Johnsson. "A computational array for the QR-method". In *Conference on advanced research in VLSI.*, pages 123–129. M.I.T., 1982.

[29] S. L. Marple Jr. "A tutorial overview of modern spectral estimation". In *IEEE ICASSP*, pages 2152–2157, 1989.

[30] S. Kalson. *"Recursive least squares filtering algorithms with systolic array architectures:A geometrical approach"*. PhD thesis, University of California, Los Angeles, 1985.

[31] S. Kalson and K. Yao. "Systolic array processing for order and time recursive generalized least-squares estimation". In *Proc. SPIE 564, Real-Time Signal Processing VIII*, pages 28–38, 1985.

[32] M. Kaveh and A. J. Barabell. "The statistical performance of MUSIC and the minimum norm algorithms in resolving plane waves in noise". *IEEE Trans. on Acous., Speech, and Signal Processing*, ASSP-34(2):331–340, Apr. 1986.

[33] H. T. Kung. "Why systolic architectures?". *IEEE Computer*, 15:37, Jan. 1982.

[34] S. Y. Kung. *"VLSI Array Processors"*. Prentice-Hall, 1988.

[35] C. L. Lawson and R. J. Hanson. *"Solving least squares problems"*. Prentice-Hall, Englewood Cliffs, N.J., 1974.

[36] F. Ling. ***"Rapidly convergent adaptive filtering algorithms with applications to adaptive equalization and channel estimation"***. PhD thesis, Northeastern University, Sept 1984.

[37] F. Ling, D. Manolakis, and J. G. Proakis. "A recursive modified gram-schmidt algorithm for least-squares estimation". *IEEE Trans. on Acous., Speech, and Signal Processing*, ASSP-34(4):829–836, Aug. 1986.

[38] K.J.R. Liu, S.F. Hsieh, and K. Yao. "Two-level pipelined implementation of systolic block Householder transformations with application to RLS algorithm ". In *Int'l Conf. on Application-Specific Array Processors*, Sept. 1990.

[39] F. T. Luk and S. Qiao. "Analysis of a recursive least-squares signal-processing algorithm". *SIAM J. Sci. Stat. Comput.*, 10(3):407–418, May 1989.

[40] D. Manolakis, F. Ling, and J. G. Proakis. "Efficient time-recursive least-squares algorithms for finite-memory adaptive filtering". *IEEE Trans. on Circuits and Systems*, CAS-34(4):400–407, Apr. 1987.

[41] J. G. McWhirter. "Recursive least-squares minimisation using a systolic array". In *Proc. SPIE 431, Real time signal processing VI*, pages 105–112, 1983.

[42] J. G. McWhirter and T. J. Shepherd. "Systolic array processor for mvdr beamforming". In *IEE Proceedings, Pt. F, 136*, pages 75–80, 1989.

[43] Bernard Mulgrew and Colin F. N. Cowan. *"Adaptive filters and equalisers"*. Kluwer Academic Publishers, 1988.

[44] J. G. Nash, K. W. Przytula, and S. Hansen. "Systolic/cellular processor for linear algebraic operations". In *VLSI Signal Processing II*, pages 306–315, 1986.

[45] W. Murray P. E. Gill, G. H. Golub and M. A. Saunders. "Methods for modifying matrix factorizations". *Mathematics of Computation*, 28(126):505–535, Apr. 1974.

[46] B. N. Parlett. "Analysis of algorithms for reflections in bisectors". *SIAM Review*, 13(2):197–208, Apr 1971.

[47] C. M. Rader. "Wafer-scale systolic array for adaptive antenna processing". In *IEEE ICASSP*, pages 2069–2071, 1988.

[48] C. M. Rader and A. O. Steinhardt. "Hyperbolic Householder transformations". *IEEE Trans. Acoust., Speech, Signal Processing*, 34(6):1589–1602, Dec. 1986.

[49] B. D. Rao. "Perturbation analysis of an SVD-based linear prediction method for estimating the frequencies of multiple sinusoids". *IEEE Trans. Acoust., Speech, Signal Processing*, 36(7):1026–1035, July 1988.

[50] B. D. Rao and K. V. S. Hari. "Statistical performance analysis of the minimum-norm method". In *IEEE ICASSP*, pages 2760–2763, 1989.

[51] J. P. Reilly, W. G. Chen, and K. M. Wong. "A fast QR-based array-processing algorithm". In *SPIE Vol. 975 Advanced Algo. and Arch. for Signal Proc III*, pages 36–47, 1988.

[52] Ralph Schmidt. "Multiple emitter location and signal parameter estimation". In *Proc. RADC Spectrum Estimation Workshop*, pages 243–258, 1979.

[53] Robert Schreiber. "Implementation of adaptive array algorithms". *IEEE Trans. on ASSP*, ASSP-34:338–346, March 1986.

[54] Robert Schreiber and Wei-Pai Tang. "On systolic arrays for undating the Cholesky factorization". *BIT*, 26:451–466, 1986.

[55] A. O. Steinhardt. "Householder transforms in signal processing". *IEEE ASSP Magazine*, 5(3):4–12, July 1988.

[56] Peter Strobach. *"Linear prediction theory: A mathematical basis for adaptive systems"*. Springer-Verlag, 1990.

[57] J. R. Treichler, Jr. C. R. Johnson, and M. G. Larimore. *"Theory and design of adaptive filters"*. John Wiley & Sons, New York, 1987.

[58] Nai-Kuan Tsao. "A note on implementing the Householder transformation". *SIAM J. Numer. Anal.*, 12(1):53–58, Mar. 1975.

[59] D. W. Tufts and R. Kumaresan. "Estimation of frequencies of multiple sinusoids: making linear prediction perform like maximum likelihood". *Proc. IEEE*, 70(9):975–989, Sept 1982.

[60] D. W. Tufts, R. J. Vaccaro, and A. C. Kot. "Analysis of estimation of signal parameters by linear-prediction at high SNR using matrix approximation". In *IEEE ICASSP*, pages 2194–2197, 1989.

[61] Barry D. Van Veen and Kevin M. Buckley. "Beamforming: a versatile approach to spatial filtering". *IEEE ASSP Magazine*, 5(2):4–24, Apr. 1988.

[62] J. S. Walther. "A unified algorithm for elementary functions". In *AFIPS Conf. Proc., Vol. 38*, pages 379–385, 1971.

[63] J. H. Wilkinson. *"The algebraic eigenvalue problem"*. Oxford University Press, England, 1965.

[64] B. Yang and J. F. Bohme. "Systolic implementation of a general adaptive array processing algorithm". In *IEEE ICASSP*, pages 2785–2788, 1988.

[65] B. Yang and J. F. Bohme. "On a systolic implementation and the numerical properties of a multiple constrained adaptive bemformer". In *IEEE ICASSP*, pages 2819–2822, 1989.