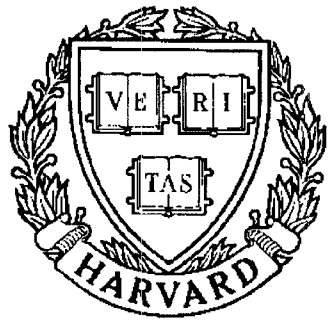
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry

Fast Orthogonalization Algorithm and Parallel Implementation for AR Spectral Estimation Based on Forward-Backward Linear Prediction

by K.J.R. Liu and S.F. Hsieh

TR 91-70

Fast Orthogonalization Algorithm and Parallel Implementation for AR Spectral Estimation Based on Forward-Backward Linear Prediction

K.J. Ray Liu
Electrical Engineering Dept.
Systems Research Center
University of Maryland
College Park, MD 20742

S.F. Hsieh
Dept. of Communication
Engineering
Nat'l Chiao Tung University
Hsinchu, Taiwan 30039

ABSTRACT

High-resolution spectral estimation is an important subject in many applications of modern signal processing. The fundamental problem in applying various high-resolution spectral estimation algorithms is the computational complexity. Recently, the truncated QR methods have been shown to be comparable to the SVD-based methods for the sinusoidal frequency estimation based on the forward-backward linear prediction (FBLP) model. However, without exploiting the special structure of the FBLP matrix, the QR decomposition (QRD) of the FBLP matrix has the computational complexity on the order of $\frac{2}{3}(6m-n)n^2+O(n^2)$ for a $2m \times n$ FBLP matrix. Here we propose a fast algorithm to perform the QRD of the FBLP matrix. It is based on exploiting the special Toeplitz-Hankel form of the FBLP matrix. The computational complexity is then reduced to $10n^2+4mn+O(n)$. The fast algorithm can also be easily implemented onto a linear systolic array. The number of time steps required is further reduced to $2m+5n-4$ by using the parallel implementation. The geometric transformation, which improves the numerical stability, for the downdating of the Cholesky factors is also considered.

1 Introduction

High-resolution spectral estimation is an important subject in many applications of modern signal processing [1]. The fundamental problem in applying various high-resolution spectral estimation algorithms is the computational complexity. In the pioneering paper of Tufts and Kumaresan [2], 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 a 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 computation required by SVD makes it unsuitable for *real time* super-resolution applications.

Recently, the truncated QR methods [5] have been shown to be comparable to the SVD-based methods in various situations. It is very effective for the sinusoidal frequency estimation based on the FBLP model. However, without considering the special structure of the FBLP matrix, the QR decomposition (QRD) of the FBLP matrix still has the computational complexity on the order of $O(n^3)$.

Seeking fast algorithms for specially structured matrices has captured lots of attention recently, especially the Toeplitz-structured matrices are used in many signal processing applications [3, 4, 9, 10, 11, 12, 13]. However, exploiting the special structure of the FBLP matrix for fast algorithm implementation has not yet been considered so far. Here we propose a fast algorithm to perform the QRD of the FBLP matrix. The computational cost of the truncated QR methods can be further reduced from $O(n^3)$ to $O(n^2)$ which makes it more attractive than the SVD-based methods. Without exploiting the special structure of

the FBLP matrix, the straight-forward QRD of the FBLP matrix has the computational complexity on the order of $\frac{2}{3}(6m-n)n^2+O(n^2)$ for a $2m \times n$ FBLP matrix. The proposed fast algorithm reduces it to the order of $10n^2+4mn+O(n)$. The geometric transformation is also considered to improve the numerical stability in downdating the Cholesky factors. We will also show that the proposed fast algorithm is amenable to parallel processing. A fully-pipelined linear systolic array based on the multi-phase operations is used to implement the fast algorithm parallelly. The required time steps is further reduced to $2m+5n-4$.

This paper is organized as follows. The basic properties and the special structure of the FBLP matrix are presented in Section 2 and exploiting the Toeplitz-Hankel structure is considered in Section 3. In Section 4, the rank-1 modifications of the Cholesky factors are briefly reviewed and the geometric transformation is proposed to improve the numerical stability in downdating the Cholesky factors. The fast algorithm is then given in Section 5. Finally the parallel implementation is considered in Section 6.

2 Forward-Backward Linear Prediction

2.1 Forward and Backward Linear Prediction

Suppose we observe a time sequence $u(i-1), u(i-2), \cdots, u(i-M)$, and would like to predict $u(i)$ based on a linear LS estimation. The forward linear prediction problem is to minimize the sum of the forward prediction-error energy,

$$\min_{\underline{w}_f} \sum_{i=M+1}^{N} |e_f(i)|^2, \tag{1}$$

where

$$e_f(i) = u(i) - \underline{w}_f^T \underline{u}(i-1),$$

$$\underline{u}^T(i-1) = [u(i-1), u(i-2), \cdots, u(i-M)],$$

and $\underline{w}_f \in \Re^{M\times 1}$ is a forward prediction weight vector. This is equivalent to solving the LS problem

$$A_f \underline{w}_f \approx \underline{b}_f, \tag{2}$$

where

$$A_f = \begin{bmatrix} u(1) & u(2) & \cdots & u(M) \\ u(2) & u(3) & \cdots & u(M+1) \\ \vdots & \vdots & & \vdots \\ u(N-M) & u(N-M+1) & \cdots & u(N-1) \end{bmatrix},$$

and

$$\underline{b}_f^T = [u(M+1), \cdots, u(N)].$$

On the other hand, for the backward linear prediction, we observe a sequence $u(i-M+1), u(i-M+2), \cdots, u(i)$, and would like to predict $u(i-M)$ based on a linear LS estimation. The backward linear prediction problem is to optimize the criterion,

$$\min_{\underline{w}_b} \sum_{i=M+1}^{N} |e_b(i)|^2, \tag{3}$$

where

$$e_b(i) = u(i-M) - \underline{w}_b^T \underline{u}^B(i),$$

$$\underline{u}^{BT}(i) = [u(i-M+1), u(i-M+2), \cdots, u(i)],$$

and $\underline{w}_b \in \Re^{M\times 1}$ is a backward prediction weight vector. Here B denotes the backward arrangement of a vector, that is, $\underline{u}^B(i)$ is a backward arrangement of the vector $\underline{u}(i)$. Similarly, this is equivalent to solving the LS problem

$$A_b \underline{w}_b \approx \underline{b}_b, \tag{4}$$

where

$$A_b = \begin{bmatrix} u(M+1) & u(M) & \cdots & u(2) \\ \vdots & \vdots & & \vdots \\ u(N-1) & u(N-2) & \cdots & u(N-M) \\ u(N) & u(N-1) & \cdots & u(N-M+1) \end{bmatrix},$$

and

$$\underline{b}_b^T = [u(1), \cdots, u(N-M)].$$

2.2 Forward-Backward Linear Prediction

To obtain a smoother result, we can combine both the forward and backward linear prediction together. We call this th forward-backward linear prediction (FBLP) method. The idea was originated by Burg [6] for the lattice predictors and the first application of the FBLP method to the design of linear predictor based on the method of least-squares was developed independently by Ulrych and Clayton (1976) and Nuttall (1976) [7]. To improve the performance, Tufts and Kumaresan [2] developed a modified FBLP method which is very effective for estimating closely spaced frequencies.

The FBLP method is to minimize the sum of the FBLP errors energy,

$$\min_{\underline{w}} \sum_{i=M+1}^{N} [|e_f(i)|^2 + |e_b(i)|^2]. \tag{5}$$

The data matrix $A \in \Re^{2(N-M)\times M}$ of the FBLP method is given by

$$A = \begin{bmatrix} A_f \\ - - - \\ A_b \end{bmatrix}, \tag{6}$$

and the desired response vector $\underline{b}$ is

$$\underline{b}^T = [\underline{b}_f^T \vdots \underline{b}_b^T]. \tag{7}$$

The FBLP method is to solve the following LS problem,

$$A\underline{w} \cong \underline{b}. \tag{8}$$

Denote $\underline{\hat{w}}_f$, $\underline{\hat{w}}_b$, and $\underline{\hat{w}}$ as the optimal weight vectors of the forward, backward, and forward-backward linear prediction, respectively. It can be easily shown from the normal equation approach that

$$\underline{\hat{w}} = [\Phi_f + \Phi_b]^{-1}[\Phi_f \underline{\hat{w}}_f + \Phi_b \underline{\hat{w}}_b], \tag{9}$$

where $\Phi_f = A_f^T A_f$ and $\Phi_b = A_b^T A_b$. It is certainly obvious that $\underline{\hat{w}}$ is a weighted average of $\underline{\hat{w}}_f$ and $\underline{\hat{w}}_b$. Accordingly, a smoother solution than that of the forward and backward linear predictions can be obtained. It is unlikely to develop fast algorithms for the FBLP method based on (9), though fast algorithms to obtain $\underline{\hat{w}}_f$ and $\underline{\hat{w}}_b$ do exist by exploiting the Toeplitz or Hankel structure.

An augmented form of the FBLP method can be obtained by putting A and $\underline{b}$ together

as

$$[A \vdots \underline{b}] = \begin{bmatrix} u(1) & u(2) & \cdots & u(M) & u(M+1) \\ u(2) & u(3) & \cdots & u(M+1) & u(M+2) \\ \vdots & \vdots & & \vdots & \vdots \\ u(N-M) & u(N-M+1) & \cdots & u(N-1) & u(N) \\ ---- & ---- & ---- & ---- & ---- \\ u(M+1) & u(M) & \cdots & u(2) & u(1) \\ \vdots & \vdots & & \vdots & \vdots \\ u(N-1) & u(N-2) & \cdots & u(N-M) & u(N-M-1) \\ u(N) & u(N-1) & \cdots & u(N-M+1) & u(N-M) \end{bmatrix} = \begin{bmatrix} H \\ --- \\ T \end{bmatrix}. \tag{10}$$

It is interesting to see that there is a special structure in this augmented matrix. The matrix can be partitioned into two parts; the upper submatrix is of Hankel structure and the lower one is of Toeplitz structure. Furthermore, both matrices can be related as follows,

$$H = TJ, \tag{11}$$

where $J \in \Re^{(M+1)\times(M+1)}$ is an exchange matrix given by

$$J = \begin{bmatrix} & & & & 1 \\ & & & \cdot & \\ & & \cdot & & \\ & \cdot & & & \\ 1 & & & & \end{bmatrix}.$$

The matrix of the form as given in (10) is called the *Toeplitz-Hankel* matrix. As we can see,

the augmented matrix of the FBLP problem is of the Toeplitz-Hankel form with a special property, *i.e.*

$$K = [A \vdots \underline{b}] = \begin{bmatrix} H \\ - - - - \\ T \end{bmatrix} = \begin{bmatrix} TJ \\ - - - - \\ T \end{bmatrix} = \begin{bmatrix} \underline{k}_1 \\ \underline{k}_2 \\ \vdots \\ \underline{k}_{2(N-M)} \end{bmatrix}. \tag{12}$$

This special property can be used for developing a fast algorithm that will be considered in following sections.

3 Exploiting the Toeplitz-Hankel Structure

By using the truncated QR method for the high-resolution AR spectral estimation, the key computational issue is to solve the FBLP LS problem based on the QR decomposition (QRD). Without considering the special structure, a conventional QRD requires $\approx 4(N - M)M^2 + O(M^2)$ multiplications to obtain the upper triangular matrix R. This is on the order of $O(M^3)$ since usually $N \gg M$. Thus, a reasonable approach is to find a fast algorithm for the FBLP LS problem by exploiting its special Toeplitz-Hankel structure. This problem has not been considered so far, though the LS problem with Toeplitz structure has been studied extensively [3, 4, 9, 10, 11, 12, 13].

The Toeplitz part of the Toeplitz-Hankel matrix can be partitioned as

$$T = \begin{bmatrix} u(M+1) & \underline{x}^T \\ \underline{y} & \tilde{T} \end{bmatrix} = \begin{bmatrix} \tilde{T} & \underline{u} \\ \underline{v}^T & u(N-M) \end{bmatrix}, \tag{13}$$

where

$$\tilde{T} = \begin{bmatrix} u(M+1) & u(M) & \cdots & u(2) \\ u(M+2) & u(M+1) & \cdots & u(3) \\ \vdots & \vdots & & \vdots \\ u(N-1) & u(N-2) & \cdots & u(N-M) \end{bmatrix},$$

$$\underline{x}^T = [u(M), \cdots, u(2), u(1)],$$

$$\underline{y}^T = [u(M+2), \cdots, u(N-1), u(N)],$$

$$\underline{u}^T = [u(1), u(2), \cdots, u(N-M-1)],$$

$$\underline{v}^T = [u(N), u(N-1), \cdots, u(N-M+1)],$$

and the Hankel part of the Toeplitz-Hankel matrix can be partitioned as

$$H = TJ = \begin{bmatrix} \underline{u} & \tilde{H} \\ u(N-M) & \underline{v}^{BT} \end{bmatrix} = \begin{bmatrix} \underline{x}^{BT} & u(M+1) \\ \tilde{H} & \underline{y} \end{bmatrix}, \tag{14}$$

where

$$\tilde{H} = \tilde{T}J = \begin{bmatrix} u(2) & u(3) & \cdots & u(M+1) \\ u(3) & u(4) & \cdots & u(M+2) \\ \vdots & \vdots & & \vdots \\ u(N-M) & u(N-M+1) & \cdots & u(N-1) \end{bmatrix},$$

$$\underline{v}^{BT} = [u(N-M+1), \cdots, u(N-1), u(N)] = \underline{v}^T J,$$

and

$$\underline{x}^{BT} = [u(1), u(2), \cdots, u(M)] = \underline{x}^T J.$$

Again, here B denotes the backward arrangement of a vector.

Now, from the above partitions, the Toeplitz-Hankel matrix K can be partitioned as follows,

$$K = \begin{bmatrix} \underline{u} & \tilde{H} \\ u(N-M) & \underline{v}^{BT} \\ ---- & ---- \\ u(M+1) & \underline{x}^T \\ \underline{y} & \tilde{T} \end{bmatrix}, \tag{15}$$

and

$$K^T K = \begin{bmatrix} \underline{u}^T\underline{u} + u^2(N-M) + u^2(M+1) + \underline{y}^T\underline{y} & \underline{u}^T\tilde{H} + u(N-M)\underline{v}^{BT} + u(M+1)\underline{x}^T + \underline{y}^T\tilde{T} \\ \tilde{H}^T\underline{u} + \underline{v}^B u(N-M) + \underline{x}u(M+1) + \tilde{T}^T\underline{y} & \tilde{H}^T\tilde{H} + \tilde{T}^T\tilde{T} + \underline{v}^B\underline{v}^{BT} + \underline{x}\,\underline{x}^T \end{bmatrix}. \tag{16}$$

Also, the matrix K can be partitioned as

$$K = \begin{bmatrix} \underline{x}^{BT} & u(M+1) \\ \tilde{H} & \underline{y} \\ ---- & ---- \\ \tilde{T} & \underline{u} \\ \underline{v}^T & u(N-M) \end{bmatrix}, \tag{17}$$

and with this partition, we have

$$K^T K = \begin{bmatrix} \underline{x}^B\underline{x}^{BT} + \tilde{H}^T\tilde{H} + \tilde{T}^T\tilde{T} + \underline{v}\,\underline{v}^T & \underline{x}^B u(M+1) + \tilde{H}^T\underline{y} + \tilde{T}^T\underline{u} + \underline{v}u(N-M) \\ u(M+1)\underline{x}^{BT} + \underline{y}^T\tilde{H} + \underline{u}^T\tilde{T} + u(N-M)\underline{v}^T & u^2(M+1) + \underline{y}^T\underline{y} + \underline{u}^T\underline{u} + u^2(N-M) \end{bmatrix}. \tag{18}$$

Let the QRD of the matrix K be $K = QR$, where $R \in \Re^{(M+1)\times(M+1)}$ is an upper

triangular matrix and it can also be partitioned as follows,

$$R = \begin{bmatrix} r_{1,1} & \underline{r}_1^T \\ \underline{0} & R_b \end{bmatrix} = \begin{bmatrix} R_t & \underline{r}_2 \\ \underline{0}^T & r_{M+1,M+1} \end{bmatrix}, \tag{19}$$

where $R_b \in \Re^{M\times M}$ is the principal bottom submatrix of R, $R_t \in \Re^{M\times M}$ is the principal top submatrix of R, and

$$\underline{r}_1^T = [r_{1,2}, r_{1,3}, \cdots, r_{1,M+1}],$$

$$\underline{r}_2^T = [r_{1,M+1}, r_{2,M+1}, \cdots, r_{M,M+1}].$$

Note that both R_b and R_t are upper triangular matrices. Since the matrix Q is orthogonal, we have

$$K^T K = R^T R, \tag{20}$$

and

$$R^T R = \begin{bmatrix} r_{11}^2 & r_{11}\underline{r}_1^T \\ \underline{r}_1 r_{11} & \underline{r}_1\underline{r}_1^T + R_b^T R_b \end{bmatrix} = \begin{bmatrix} R_t^T R_t & R_t^T \underline{r}_2 \\ \underline{r}_2^T R_t & \underline{r}_2^T \underline{r}_2 + r_{M+1,M+1}^2 \end{bmatrix}. \tag{21}$$

Define

$$\tilde{K} = \begin{bmatrix} \tilde{H} \\ - - - \\ \tilde{T} \end{bmatrix}, \tag{22}$$

then we have

$$\tilde{K}^T \tilde{K} = \tilde{H}^T \tilde{H} + \tilde{T}^T \tilde{T}. \tag{23}$$

From the lower right submatrices of (16) and (21), we obtain

$$R_b^T R_b + \underline{r}_1 \underline{r}_1^T = \tilde{K}^T \tilde{K} + \underline{x}\,\underline{x}^T + \underline{v}^B \underline{v}^{BT}. \tag{24}$$

Also, from the upper left submatrices of (18) and (21), we have

$$R_t^T R_t = \tilde{K}^T \tilde{K} + \underline{v}\,\underline{v}^T + \underline{x}^B \underline{x}^{BT}. \tag{25}$$

Substituting (25) to (24), we obtain the relation between R_b and R_t as given by

$$R_b^T R_b = R_t^T R_t + \underline{x}\,\underline{x}^T - \underline{x}^B \underline{x}^{BT} + \underline{v}^B \underline{v}^{BT} - \underline{v}\,\underline{v}^T - \underline{r}_1 \underline{r}_1^T. \tag{26}$$

4 Rank-1 Modifications of Cholesky Factors

The key issues in computing (26) are the rank-1 updatings and downdatings of the Cholesky factors. Let both R_u and R_d be $M \times M$ upper triangular matrices with positive elements on the diagonal. The rank-1 updating of the Cholesky factor R_u is given by

$$R_d^T R_d = R_u^T R_u + \underline{x}\,\underline{x}^T, \tag{27}$$

where $\underline{x}$ is an M-vector and R_d is the Cholesky factor after R_u is updated by $\underline{x}$. Consider the QRD of the augmented matrix

$$\begin{bmatrix} \underline{x}^T \\ R_u \end{bmatrix} = QR. \tag{28}$$

Obviously, since Q is an orthogonal matrix, we see that

$$R^T R = R_u^T R_u + \underline{x}\,\underline{x}^T = R_d^T R_d. \tag{29}$$

That is, the rank-1 updating can be achieved by obtaining the QRD of $[\underline{x}, R_u^T]^T$. Since R_u is an upper triangular matrix, the orthogonal matrix Q is a product of M Givens rotations matrices given by

$$Q^T = U_{M,M+1} U_{M-1,M} \cdots U_{1,2}, \tag{30}$$

where

$$U_{k,k+1} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c_k & s_k & & \\ & & -s_k & c_k & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix},$$

with

$$c_k = \frac{a}{\sqrt{a^2+b^2}}, \quad s_k = \frac{b}{\sqrt{a^2+b^2}}.$$

a and b are the elements in the k-th and $k+1$-th rotation planes, respectively. Accordingly,

$$\begin{bmatrix} R_d \\ \underline{0}^T \end{bmatrix} = U_{M,M+1} U_{M-1,M} \cdots U_{1,2} \begin{bmatrix} \underline{x}^T \\ R_u \end{bmatrix}. \tag{31}$$

On the other hand, the rank-1 downdating of the Cholesky factor R_d can be obtained from (27) as

$$R_u^T R_u = R_d^T R_d - \underline{x}\underline{x}^T. \tag{32}$$

Similarly, the downdating can be obtained by a sequence of hyperbolic rotations given by

$$\begin{bmatrix} R_u \\ \underline{0}^T \end{bmatrix} = \tilde{U}_{M,M+1} \tilde{U}_{M-1,M} \cdots \tilde{U}_{1,2} \begin{bmatrix} \underline{x}^T \\ R_d \end{bmatrix}, \tag{33}$$

where the hyperbolic rotation matrices are of the form

$$\tilde{U}_{k,k+1} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & \tilde{c}_k & -\tilde{s}_k & & \\ & & -\tilde{s}_k & \tilde{c}_k & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix},$$

with

$$\tilde{c}_k = \frac{a}{\sqrt{a^2 - b^2}}, \quad \tilde{s}_k = \frac{b}{\sqrt{a^2 - b^2}}.$$

Again, here a and b are the elements in the k-th and $k+1$-th rotation planes, respectively.

Geometric Transformation

The hyperbolic rotation is well known for its numerical instability. This can be seen easily by noting that if $a \approx b$, then $\tilde{c}_k$ and $\tilde{s}_k$ can be possibly unbounded or even imaginary-valued. Specifically, the hyperbolic rotation of two given rows is given by

$$\begin{bmatrix} \tilde{c}_k & -\tilde{s}_k \\ -\tilde{s}_k & \tilde{c}_k \end{bmatrix} \begin{bmatrix} a_{k,k} & a_{k,k+1} & \cdots, & a_{k,M+1} \\ b_{k,k} & b_{k,k+1} & \cdots, & b_{k,M+1} \end{bmatrix} = \begin{bmatrix} \hat{a}_{k,k} & \hat{a}_{k,k+1} & \cdots, & \hat{a}_{k,M+1} \\ 0 & \hat{b}_{k,k+1} & \cdots, & \hat{b}_{k,M+1} \end{bmatrix}, \quad (34)$$

where

$$\hat{a}_{k,k} = \sqrt{a_{k,k}^2 - b_{k,k}^2},$$

$$\tilde{c}_k = \frac{a_{k,k}}{\hat{a}_{k,k}}, \quad \tilde{s}_k = \frac{b_{k,k}}{\hat{a}_{k,k}},$$

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

$$\hat{a}_{k,j} = \tilde{c}_k a_{k,j} - \tilde{s}_k b_{k,j},$$

$$\hat{b}_{k,j} = -\tilde{s}_k a_{k,j} + \tilde{c} b_{k,j}. \quad (35)$$

Since both $\tilde{c}$ and $\tilde{s}$ are unbounded parameters, the computation of (35) may suffer severe roundoff errors or even overflow under finite-precision implementation. In fact, as shown in Fig.1, the hyperbolic rotation parameters can be expressed by

$$\begin{aligned} \tilde{c}_k &= \sec\theta = \frac{1}{\cos\theta}, \\ \tilde{s}_k &= \tan\theta = \frac{\sin\theta}{\cos\theta}, \end{aligned} \tag{36}$$

where

$$\cos\theta = \frac{\hat{a}_{k,k}}{a_{k,k}}, \quad \sin\theta = \frac{b_{k,k}}{a_{k,k}}. \tag{37}$$

Now that

$$\begin{aligned} -\sin\theta\hat{a}_{k,j} &= -\frac{\sin\theta}{\cos\theta}a_{k,j} + \frac{\sin^2\theta}{\cos\theta}b_{k,j} \\ &= -\tilde{s}_k a_{k,j} + \tilde{c}_k b_{k,j} - \cos\theta b_{k,j}, \end{aligned} \tag{38}$$

equation (35) can be expressed as

$$\begin{aligned} \hat{a}_{k,j} &= \frac{1}{\cos\theta}(a_{k,j} - \sin\theta b_{k,j}), \\ \hat{b}_{k,j} &= -\sin\theta\hat{a}_{k,j} + \cos\theta b_{k,j}. \end{aligned} \tag{39}$$

Obviously, this is preferable because most of the computations involved bounded parameters such as $\cos\theta$ and $\sin\theta$. Similar results have also been obtained in [3], however, the approach gives no geometric sense. The new downdating algorithm is summarized as follows:

$$\hat{a}_{k,k} = \sqrt{a_{k,k}^2 - b_{k,k}^2},$$

$$\cos\theta = \frac{\hat{a}_{k,k}}{a_{k,k}}, \quad \sin\theta = \frac{b_{k,k}}{a_{k,k}},$$

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

$$\begin{aligned} \hat{a}_{k,j} &= (a_{k,j} - \sin\theta b_{k,j})/\cos\theta, \\ \hat{b}_{k,j} &= -\sin\theta\hat{a}_{k,j} + \cos\theta b_{k,j}. \end{aligned} \tag{40}$$

Such a geometric transformation is not unique. For example, it can be easily shown that, similar to the above derivations, (35) can also be transformed to

$$\hat{b}_{k,j} = (-\sin\theta a_{k,j} + b_{k,j})/\cos\theta,$$

$$\hat{a}_{k,j} = \cos\theta a_{k,j} - \sin\theta\hat{b}_{k,j}. \tag{41}$$

5 The Fast Algorithm

It is now clear how to perform the updating and downdating of the Cholesky factors. We can then go back to the computational issue of (26). As we can see, in (26), there are two rank-1 updatings and three rank-1 downdatings. Let us split (26) into a sequence of five up/down-dating equations given by

$$\begin{aligned}
R_1^T R_1 &= R_t^T R_t + \underline{x}\underline{x}^T, \\
R_2^T R_2 &= R_1^T R_1 - \underline{x}^B \underline{x}^{BT}, \\
R_3^T R_3 &= R_2^T R_2 + \underline{v}^B \underline{v}^{BT}, \\
R_4^T R_4 &= R_3^T R_3 - \underline{v}\underline{v}^T, \\
R_b^T R_b &= R_4^T R_4 - \underline{r}_1 \underline{r}_1^T,
\end{aligned} \tag{42}$$

where R_t, R_b, and $R_i, i = 1,2,3,4$ are all $M \times M$ upper triangular matrices, and all of the vectors involved are M-dimensional. The definitions of these vectors can be found from Section 3. Suppose now we have the first row of R_t, $[r_{1,1}^t, r_{1,2}^t, \cdots, r_{1,M}^t]$, to obtain the first row of R_1, $[r_{1,1}^1, r_{1,2}^1, \cdots, r_{1,M}^1]$, we use a Givens rotation such that

$$\begin{bmatrix} r_{1,1}^1 & r_{1,2}^1 & \cdots & r_{1,M}^1 \\ 0 & & \underline{x}^{T(1)} & \end{bmatrix} = U_{(1)}^1 \begin{bmatrix} u(M) & u(M-1) & \cdots & u(1) \\ r_{1,1}^t & r_{1,2}^t & \cdots & r_{1,M}^t \end{bmatrix}, \tag{43}$$

where $U^{\cdot}_{(\cdot)}$ denotes a Givens rotation matrix and $\underline{x}^{(1)}$ is an $(M-1)$-vector. With the first row of R_1, we can proceed to find the first row of R_2 by using a hyperbolic rotation given by

$$\begin{bmatrix} r^2_{1,1} & r^2_{1,2} & \cdots & r^2_{1,M} \\ 0 & & \underline{x}^{BT(1)} & \end{bmatrix} = \tilde{U}^2_{(1)} \begin{bmatrix} u(1) & u(2) & \cdots & u(M) \\ r^1_{1,1} & r^1_{1,2} & \cdots & r^1_{1,M} \end{bmatrix}, \tag{44}$$

where $\tilde{U}^{\cdot}_{(\cdot)}$ denotes a hyperbolic rotation matrix and $\underline{x}^{B(1)}$ is an $(M-1)$-vector. Similarly, the first rows of R_3, R_4 and R_b can be obtained as follows:

$$\begin{bmatrix} r^3_{1,1} & r^3_{1,2} & \cdots & r^3_{1,M} \\ 0 & & \underline{v}^{BT(1)} & \end{bmatrix} = U^3_{(1)} \begin{bmatrix} u(N-M+1) & u(N-M+2) & \cdots & u(N) \\ r^2_{1,1} & & r^2_{1,2} & \cdots & r^2_{1,M} \end{bmatrix}, \tag{45}$$

$$\begin{bmatrix} r^4_{1,1} & r^4_{1,2} & \cdots & r^4_{1,M} \\ 0 & & \underline{v}^{T(1)} & \end{bmatrix} = \tilde{U}^4_{(1)} \begin{bmatrix} u(N) & u(N-1) & \cdots & u(N-M+1) \\ r^3_{1,1} & r^3_{1,2} & \cdots & r^3_{1,M} \end{bmatrix}, \tag{46}$$

$$\begin{bmatrix} r^b_{1,1} & r^b_{1,2} & \cdots & r^b_{1,M} \\ 0 & & \underline{r}^{BT(1)}_1 & \end{bmatrix} = \tilde{U}^5_{(1)} \begin{bmatrix} r_{1,2} & r_{1,3} & \cdots & r_{1,M+1} \\ r^4_{1,1} & r^4_{1,2} & \cdots & r^4_{1,M} \end{bmatrix}, \tag{47}$$

where $U^3_{(1)}$ is a Givens rotation matrix, $\tilde{U}^4_{(1)}$ and $\tilde{U}^5_{(1)}$ are hyperbolic rotations, and $\underline{v}^{B(1)}$, $\underline{v}^{(1)}$, and $\underline{r}^{(1)}_1$ are all $(M-1)$-vectors. From (19), it is clear that the first row of R_b is, in fact, the second row of R_t, except for $r^b_{1,M}$. That is,

$$[r^t_{2,2}, r^t_{2,3}, \cdots, r^t_{2,M}] = [r^b_{1,1}, r^b_{1,2}, \cdots, r^b_{1,M-1}]. \tag{48}$$

Note that $r^t_{i,j} = 0$ for $i > j$. With the second row of R_t and $\underline{x}^{(1)}$, $\underline{x}^{B(1)}$, $\underline{v}^{B(1)}$, $\underline{v}^{(1)}$, and $\underline{r}^{(1)}_1$, the computations of eqns. (43) - (47) can be repeated to obtain the second row of R_b by applying two updatings, $U^1_{(2)}$ and $U^3_{(2)}$, and three downdatings, $\tilde{U}^2_{(2)}$, $\tilde{U}^4_{(2)}$, and $\tilde{U}^5_{(2)}$. The dimension of each vector is $M-1$ for this second iteration. Similar iterations can

be continued to generate the subsequent rows of R_t and R_b until the matrix R in (19) is obtained.

The issue now is how to obtain the first row of R_t. This is equivalent to obtaining the first row of R. In general, there is no short cut for obtaining this row and it can be done by a sequence of Givens rotations on the matrix K to zero out the first column of K, except its leading element on the diagonal. Denote # as a "don't care" element or vector, the fast algorithm is given as follows:

The Fast Toeplitz-Hankel Orthogonalization Algorithm

(Initialization)

$\underline{x}^{T(0)} = [u(M), u(M-1), \cdots, u(1)]$,

$\underline{x}^{BT(0)} = [u(1), u(2), \cdots, u(M)]$,

$\underline{v}^{T(0)} = [u(N), u(N-1), \cdots, u(N-M+1)]$,

$\underline{v}^{BT(0)} = [u(N-M+1), u(N-M+2), \cdots, u(N)]$,

$\underline{k}_{(1)} = \underline{k}_1$,

For $i = 1$ to $2(N-M)-1$,

$$\begin{bmatrix} \underline{k}_{(i+1)} \\ \# \end{bmatrix} = U_{(i)} \begin{bmatrix} \underline{k}_{(i)} \\ \underline{k}_{i+1} \end{bmatrix},$$

End For;

$[r_{11}, \underline{r}_1^T] = [\underline{r}_1^t, \#]^T = \underline{k}_{(2(N-M))}^T$,

$\underline{r}_1^{T(0)} = \underline{r}_1^T$.

(Main Iterations)

For $i = 1$ to $M-1$,

(Phase 1)

$$\begin{bmatrix} \underline{r}_i^{1T} \\ 0, \ \underline{x}^{T(i)} \end{bmatrix} = U_{(i)}^1 \begin{bmatrix} \underline{x}^{T(i-1)} \\ \underline{r}_i^{tT} \end{bmatrix},$$

(Phase 2)

$$\begin{bmatrix} \underline{r}_i^{2T} \\ 0, \ \underline{x}^{BT(i)} \end{bmatrix} = \tilde{U}_{(i)}^2 \begin{bmatrix} \underline{x}^{BT(i-1)} \\ \underline{r}_i^{1T} \end{bmatrix},$$

(Phase 3)

$$\begin{bmatrix} \underline{r}_i^{3T} \\ 0, \ \underline{v}^{BT(i)} \end{bmatrix} = U_{(i)}^3 \begin{bmatrix} \underline{v}^{BT(i-1)} \\ \underline{r}_i^{2T} \end{bmatrix},$$

(Phase 4)

$$\begin{bmatrix} \underline{r}_i^{4T} \\ 0, \ \underline{v}^{T(i)} \end{bmatrix} = \tilde{U}_{(i)}^4 \begin{bmatrix} \underline{v}^{T(i-1)} \\ \underline{r}_i^{3T} \end{bmatrix},$$

(Phase 5)

$$\begin{bmatrix} \underline{r}_i^{bT} \\ 0, \ \underline{r}_1^{T(i)} \end{bmatrix} = \tilde{U}_{(i)}^5 \begin{bmatrix} \underline{r}_1^{T(i-1)} \\ \underline{r}_i^{4T} \end{bmatrix},$$

$\underline{r}_{i+1}^{tT}$ = the elements of $\underline{r}_i^{bT}$ excluding the last one,

End For.

As we can see, for the initialization (obtaining the first row of R), the computational cost is $[2 \cdot 2(N-M)-1] \cdot M \approx 4(N-M)M$ multiplications (since only half of the rotation needed to be done). Following this, the recursions in the main iterations are then started. As there are five rotation-like up/downdatings, the computational cost is

$$5 \times (4M + 4(M-1) + \cdots + 4 \cdot 1) = 20(\frac{M(M+1)}{2} - 1) \approx 10M^2 + O(M)$$

(for multiplication). Therefore, the total computational complexity is $\approx 10M^2+4(N-M)M$ (for multiplication) for a $2(N-M)\times M$ Toeplitz-Hankel matrix. As mentioned before, without considering the special structure, by using the conventional QRD, the computational complexity is of $\approx 4M^3 + O(M^2)$. Obviously, the proposed fast algorithm has an improvement of an order of magnitude. In general, for the QRD of a $2m \times n$ Toeplitz-Hankel matrix, the fast algorithm needs $10n^2 + 4mn + O(n)$ multiplications, while a conventional implementation needs $\frac{2}{3}(6m - n)n^2 + O(n^2)$, where $m \gg n$.

If the least-squares weight vector is of interested, a back substitution can then be used for computing the weight vector. For the truncated QR method, a truncation of the noise subspace is necessary before computing the weight vector [5].

6 Parallel Implementation

The fast algorithm obtained in the previous section not only reduces the computational complexity, but is also amenable for parallel implementation. From the fact that only the first row of the upper triangular matrix R has to be obtained first, a linear array of $M + 1$ processing cells, as shown in Fig.2, can be used to rotate the matrix K such that the first column can be zeroed out and when the initialization phase is finished, the first row of the matrix R is kept in the linear array. Fig.3 shows the initialization to obtain the first row of R. The operations of the processing cells are given in Table 1. The data matrix is arranged in a skewed manner for the systolic array implementation [15]. The idea is similar to the triangular array for the QRD proposed by Gentleman and Kung [14]. The difference is that their scheme is a general one without considering any special structure of the data matrix. Accordingly, a full triangular array is needed.

Due to the consideration of the special Toeplitz-Hankel structure, once the first row of the matrix R is available, the subsequent rows of R can be generated one by one by the main iterations given in the fast algorithm. To start the main iterations, $\underline{r}^{tT}_1$ is needed. Fortunately, it is the first M elements of the first row of R that are stored in the first M processing cells. The main iterations are now started with inputs $\underline{x}^{T(0)}$, $\underline{x}^{BT(0)}$, $\underline{x}^{BT(0)}$, $\underline{v}^{T(0)}$, and $\underline{r}^{T(0)}_1$, and the outputs are $\underline{x}^{T(1)}$, $\underline{x}^{BT(1)}$, $\underline{x}^{BT(1)}$, $\underline{v}^{T(1)}$, and $\underline{r}^{T(1)}_1$, respectively, as illustrated in Fig.4. The outputs all have one less dimensions than their inputs. The vector $\underline{r}^{bT}_1$ can now be obtained on the linear array.

As given in the main iterations, there are five different phases. The operations of the processing cells for different phases are given in Table 1. Based on the multi-phase concept proposed in [16], the outputs are fedback to the input ports for another iteration of different phases. Note that the outputs are obtained from $PE2$ to PEM. The feedback is, however, directed to the processors from $PE1$ to $PE(M-1)$. Since $\underline{r}^{tT}_2$ takes the first $M-1$ elements of $\underline{r}^{bT}_1$, it occupies the first $M-1$ processing cells. The second iteration is started once the fedback data are available. It is fully pipelined without any intermediate data arrangement and interrupt. The iterations are then continued until all the rows of R are obtained. The data movement of the third iteration is given in Fig.5 and the overall data arrangement is given in Fig.6.

The number of time steps required for this linear array implementation is now being further reduced to $2(N-M)+(5(M-1)+1) = 2N+3M-4$ (or $2m+5n-4$ for a $2m \times n$ Toeplitz-Hankel matrix) which is linearly proportional to either M or N (m or n).

If the LS weight vector is of interest, another phase for the back substitution can be started easily since all the data are now available in the linear array. The details of the

operations of the back substitution using a linear array can be found in [14, 17]

7 Conclusions

In this paper, we propose a fast algorithm for the QRD of a Toeplitz-Hankel matrix. The computational complexity for the QRD of a $2m \times n$ Toeplitz-Hankel matrix is $10n^2 + 4mn + O(n)$ multiplications, which has an order of magnitude improvement over conventional algorithms. This algorithm can also be implemented onto a fully pipelined multi-phase linear systolic array. The number of time steps required is further reduced to $2m + 5 - 4$ for the parallel implementation. An interesting point for the QRD of the specially structured matrices such as Toeplitz and Toeplitz-Hankel forms is that there is no need to store all the generated rows of the upper triangular matrix R. As long as the first row of R is known, all the subsequent row can be generated recursively, and this is also the basic principle of the proposed fast algorithm.

References

[1] S.M. Kay, Modern Spectral Estimation, Prentice Hall, 1988.

[2] 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, Sep. 1982.

[3] A.W. Bojanczyk, R.P. Brent, and F.R. de Hoog, "QR factorization of Toeplitz matrices", Numer. Math. 49, 81-94, 1986.

[4] D.R. Sweet, "Fast Toeplitz orthogonalization", Numer. Math. 43, pp.1-21, 1984.

[5] S.F. Hsieh, K.J.R. Liu, and K. Yao, "Comparisons of truncated QR and SVD methods for AR spectral estimations", in SVD and Signal Processing: Algorithms, Applications and Architectures, pp.403-418, Ed. R.J. Vaccaro, Elsevier Science Publishers, 1991.

[6] J.P. Burg, "Maximum entropy spectral analysis", 37th Ann. Intern. Meeting, Soc. Explor. Geophysics, Oklahoma City, 1967.

[7] S. Haykin, Adaptive Filter Theory, Prentice Hall, 1986.

[8] C.P. Rialan and L.L. Scharf, "Fast algorithms for computing QR and Cholesky factors of Toeplitz operators", IEEE Trans. on ASSP, Vol 36, pp.1740-1748, Nov. 1988.

[9] T. Kailath, S.-Y. Kung, and M. Morf, "Displacement ranks of matrices and linear equations", Journal of Math. Analysis and Applications 68, pp.395-407, 1979.

[10] S.-Y. Kung and Y.H. Hu, "A highly concurrent algorithm and pipelined architecture for solving Toeplitz systems", IEEE Trans. on ASSP, Vol 31, Feb. 1983.

[11] J.R. Bunch, "Stability of methods for solving Toeplitz systems of equations", SIAM J. Sci. Stat. Comput., Vol 6, pp.349-364, April 1985.

[12] A.W. Bojanczyk, R.P. Brent, and F.R. de Hoog, "Linearly connected arrays for Toeplitz least-squares problems", J. of Parallel and Distributed Computing 9, pp.261-270, 1990.

[13] J. Chun, T. Kailath and H. Lev-Ari, "Fast parallel algorithms for QR and triangular factorization", SIAM J. Sci. Stat. Comput., Vol 8, pp.899-912, Nov. 1987.

[14] W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic array," Proc. SPIE, Vol. 298, pp. 298, 1981.

[15] S.Y. Kung, VLSI Array Processors, Prentice Hall, 1988.

[16] K.J.R. Liu and K. Yao, "Multi-phase systolic algorithms for spectral decomposition", to appear, IEEE Trans. on Signal Processing, Jan. 1992.

[17] H.T. Kung, "Why systolic array?" IEEE Computer, Vol 15, pp.37, Jan. 1982.

Table and Figure Captions:

Table 1 The operations of the processing cells in different phases.

Fig.1 The geometric relationship.

Fig.2 The linear systolic array and its processing cells.

Fig.3 The initialization.

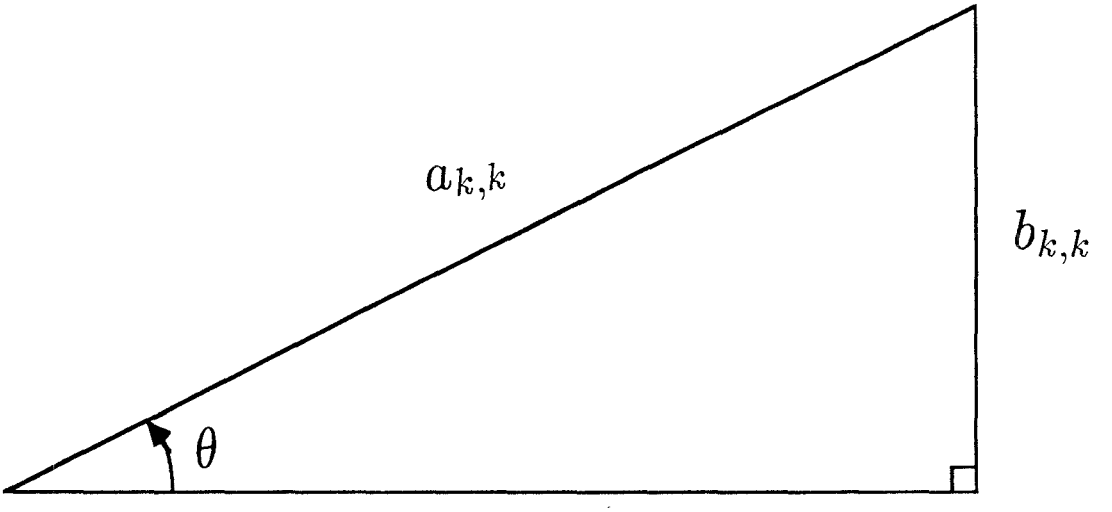
Fig.4 The first iteration with five different phases.

Fig.5 The second iteration with five different phases.

Fig.6 The overall data arrangement.

	Initialization	Phases 1 & 3	Phases 2, 4, & 5
PE1	$d = \sqrt{r^2 + x^2}$ $c = r/d, \quad s = x/d$ r=d	$d = \sqrt{r^2 + x^2}$ $c = r/d, \quad s = x/d$ r=d	$d = \sqrt{r^2 - x^2}$ $c = d/r, \quad s = x/r$ r=d
PEi, $1 \leq i \leq M+1$	$r = cr + sx$ $y = -sr + cx$	$r = cr + sx$ $y = -sr + cx$	$r = (r - sx)/c$ $y = -sr + cx$

Table 1: The operations of the processing cells in different phases



$$\hat{a}_{k,k} = \sqrt{a_{k,k}^2 - b_{k,k}^2}$$

Fig.1

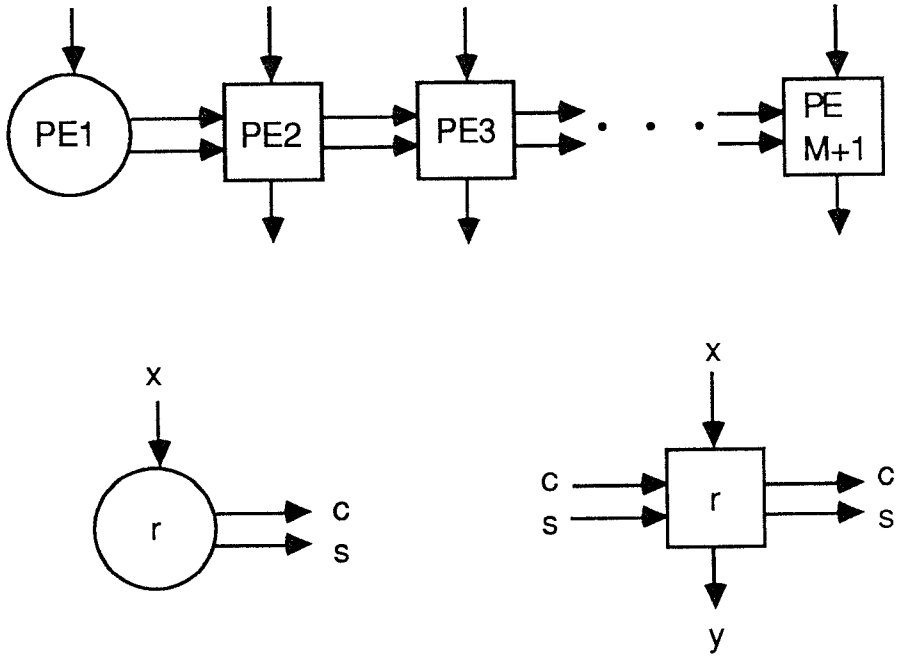


Fig.2

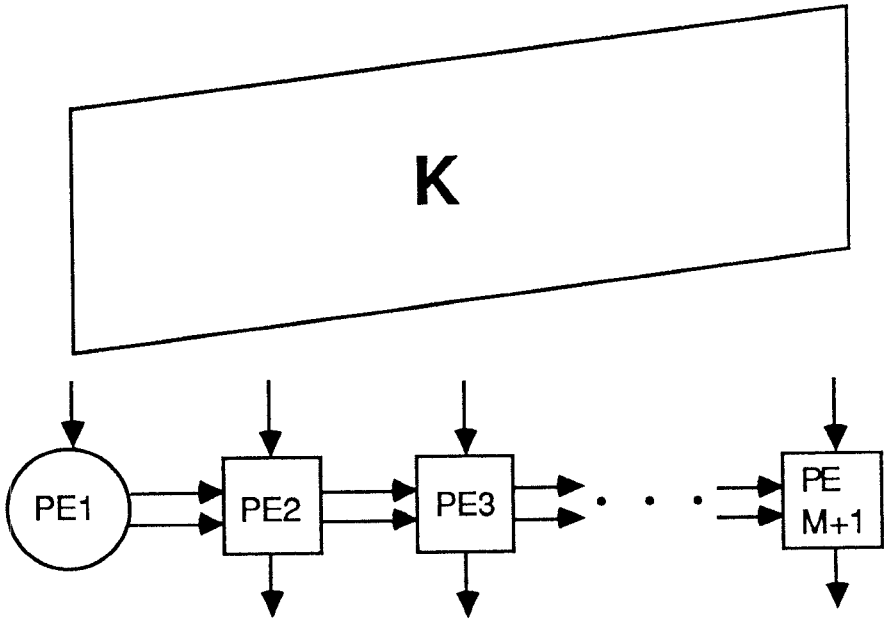


Fig.3

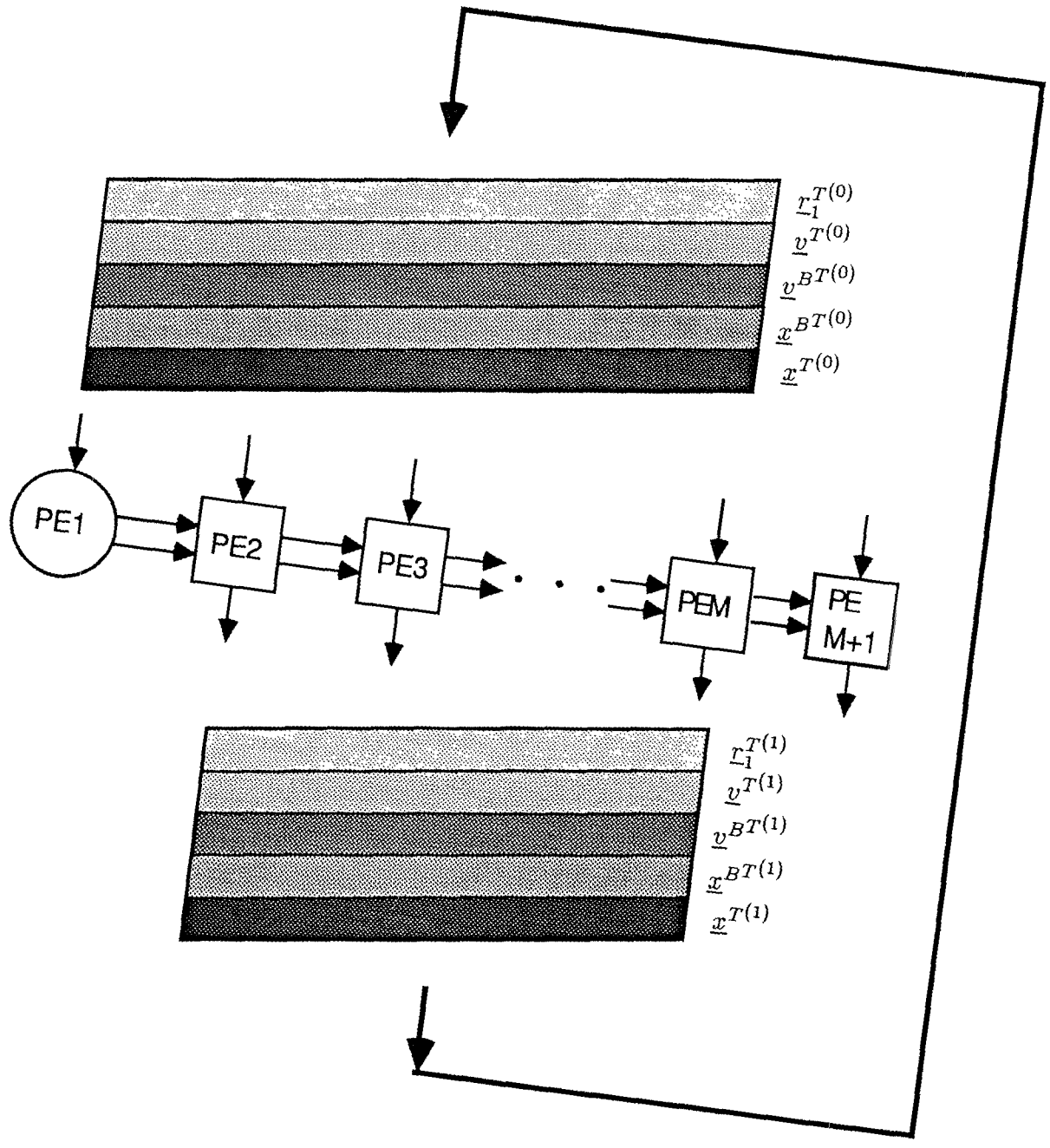

$\underline{r}_1^{T(0)}$
$\underline{v}^{T(0)}$
$\underline{v}^{BT(0)}$
$\underline{x}^{BT(0)}$
$\underline{x}^{T(0)}$
PE1
PE2
PE3
PEM
PE M+1
$\underline{r}_1^{T(1)}$
$\underline{v}^{T(1)}$
$\underline{v}^{BT(1)}$
$\underline{x}^{BT(1)}$
$\underline{x}^{T(1)}$

Fig.4

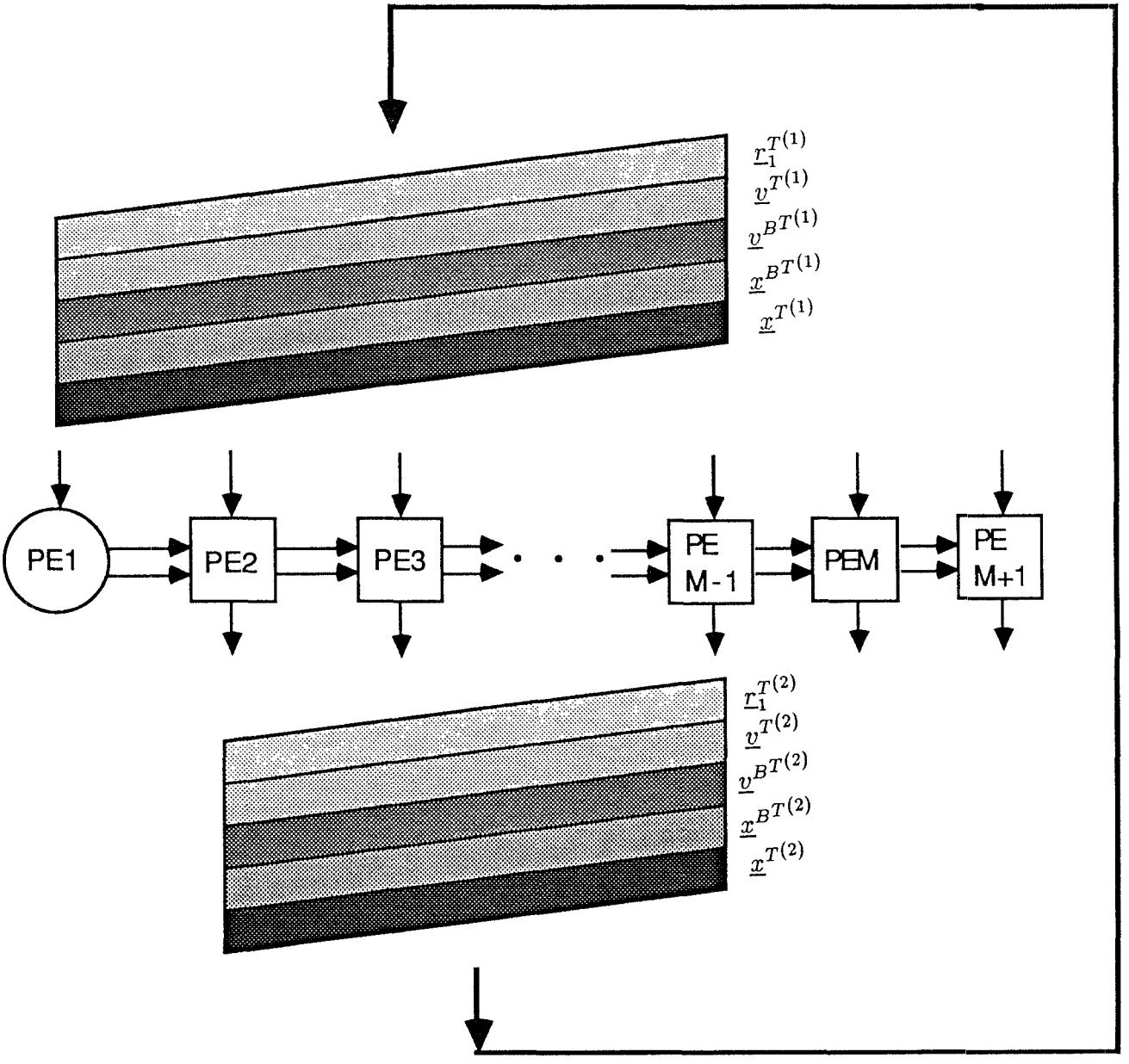

$\underline{r}_1^{T(1)}$
$\underline{v}^{T(1)}$
$\underline{v}^{BT(1)}$
$\underline{x}^{BT(1)}$
$\underline{x}^{T(1)}$
PE1
PE2
PE3
PE M-1
PEM
PE M+1
$\underline{r}_1^{T(2)}$
$\underline{v}^{T(2)}$
$\underline{v}^{BT(2)}$
$\underline{x}^{BT(2)}$
$\underline{x}^{T(2)}$

Fig.5

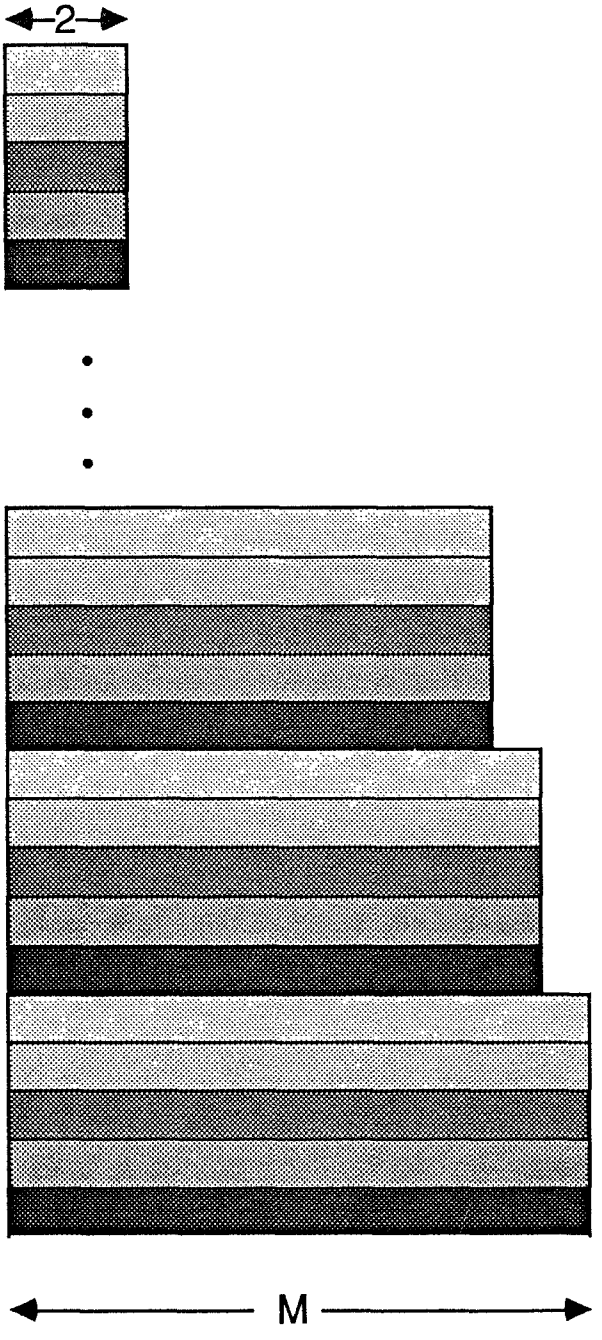


Fig.6