TECHNICAL RESEARCH REPORT

S Y S T E M S
R E S E A R C H
C E N T E R

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

Research support for this report has been provided partially by UC MICRO grant and the NSF grant NCR-8814407. It is also partially supported by NSF grant ECD-8803012-06

Systolic Implementations of Up/Down-dating Cholesky Factorization Using Vectorized Gram-Schmidt Pseudo Orthogonalization

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

TR 91-23

Systolic Implementations of Up/Down-dating Cholesky Factorization Using Vectorized Gram-Schmidt Pseudo Orthogonalization *

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

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

K. Yao
Electrical Engineering Dept.
UCLA
Los Angeles, CA 90024-1594

ABSTRACT

We propose a new class of *hyperbolic* Gram-Schmidt methods to simultaneously update and downdate the Cholesky factor of a sample covariance matrix efficiently with applications to sliding window recursive least squares (RLS) filtering problems. Several vectorized versions of this Gram-Schmidt approach are introduced, which include conventional column-updating, modified row/column-updating, and square-root-free methods. Comparisons to the existing known methods, such as Householder transformation and Givens rotation, are also given. Upon further reformulating these algorithms, a systolic triarray structure is proposed to facilitate VLSI implementations.

*This work is partially supported by a UC MICRO grant and the NSF grant NCR-8814407. It is also partially supported by NSF grant ECD-8803012-06.

1 Introduction

The QR decomposition (QRD) has been proved to be numerically stable and suitable for parallel VLSI implementations in adaptive signal processing applications such as adaptive antenna arrays, system identification, etc. [2,5,6,9,11,14,19,18,20,22,24,25]. Until now, incorporating a forgetting factor to partially discard the effects of those old observed data is among the most frequently used techniques under nonstationary conditions. This method is fairly simple and only requires updating the QRD, hence is widely used for VLSI implementations [14,18,19,20]. Another attractive alternative for time-varying adaptive signal processing is to employ a fixed-window so that this window can slide *in* (include) new data and slide *out* (discard) old data as time progresses [2, pp. 216], [13, pp. 189]. The sliding window scheme entails both updating (accepting new data) and downdating (rejecting obsolete data) at each iteration as we recursively modify the least squares (LS) solution. Previously, researches have been focused on using the hyperbolic Givens rotation (HGR)[1,12] and hyperbolic Householder transformation (HHT)[22] to effectively downdate the undesired data. However, the HGR involves higher computational cost and the HHT requires square-root operations. With these reasons, here we propose another class of simple and efficient hyperbolic methods to perform downdating, namely, hyperbolic Gram-Schmidt methods. In addition to the sliding window RLS filtering, downdating some undesired spurious (badly contaminated by noise or interference) data is also useful and of great interest for some applications [22].

Our attention will be focused on recursive vectorized block data processing, which generalizes current scalar-valued (e.g., Givens rotation) processing and can possibly reduces the I/O cost in hardware implementations. For some computing machines, computational speed is much slower compared to the data movement or the overhead to establish data transfer degrades the throughput rate; this is the reason why many algorithms are being reformulated into vector-pipelined computing [8, pp. 35] with an aim to speeding up computation and increasing the degree of parallelism in computing. Vectorized processing also offers the advantages of reducing arithmetical operations and possibly minimizing the round-off errors occurring in the intermediate stages in that the wordlengths in the internal registers can be enlarged.

A key computation for a sliding-window RLS filtering problem is to modify the Cholesky factor of the sample covariance matrix X^TX recursively as new data arrives and old data leaves. If we denote X_{new} and X_{old} as the new and old data matrices to be updated and downdated respectively from the previous data matrix X, then our goal is to obtain the recent Cholesky factor $\tilde{R}$ of $\tilde{X}^T\tilde{X}$ from the knowledge of the previous one R, where $\tilde{X}$ is the recent data matrix which is obtained by detaching X_{old} from X and attaching X_{new} to it. It can be shown that $\tilde{R}^T\tilde{R} = R^TR + X_{new}^TX_{new} - X_{old}^TX_{old}$ [21], which amounts to performing a pseudo orthogonalization on the matrix $[R^T\ X_{new}^T\ X_{old}^T]^T$ to zero out X_{new} and X_{old} via updating and downdating respectively. To generalize the QRD from orthogonality to pseudo orthogonality, we reformulate several concepts in computational algebra, such as inner-product, norms, etc. Following the introduction of the pseudo Cholesky factorization, a hyperbolic Gram-Schmidt (HGS) procedure and its simile, the hyperbolic modified Gram-Schmidt (HMGS) procedure are derived to perform pseudo orthogonalization (MGSPO) decomposition.

Similar to the HHT, these new methods can also perform rank-k updating and rank-ℓ downdating simultaneously, and are indeed generalizations of the conventional GS and MGS methods where only updating can be performed. To further avoid the cumbersome square-root operations when implementing a parallel processing VLSI structure, square-root-free versions of the HGS and HMGS methods are also provided. If $n = k + \ell$ is much smaller than the rank of R, say p, then we can tacitly reformulate these MGS algorithms such that the computational cost can be greatly reduced from $\mathcal{O}(np^2 + p^3/3)$ in conventional approaches down to $\mathcal{O}(np^2 + 2n^2p)$ flops in our new proposed schemes. We also note that the complexity of the previously known HHT [22,26,19] is of $\mathcal{O}(np^2 + 1.5p^2)$, while the HGR is of $\mathcal{O}(np^2 + 1.5np)$ [1]. In addition, this modification also makes the MGS algorithms suitable for systolic processing where massive parallelism in computation is feasible.

In Section 2, the pseudo Cholesky decomposition is proposed. The downdating of the Cholesky factor is considered in Section 3 and the up/down-dating of multiples rows is presented in Section 4. Finally, a systolic triarray for vectorized up/down-dating employing the MGSPO method is proposed Section 5. A conclusion is then given in Section 6.

2 Pseudo Cholesky Decomposition

Let Z be an $(m+n)\times p$ matrix concatenated from two real-valued data matrices $A \in \mathcal{R}^{m\times p}$ and $B \in \mathcal{R}^{n\times p}$, i .e.,

$$Z = \begin{bmatrix} A \\ B \end{bmatrix}. \tag{1}$$

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

$$R^T R = Z^T H_{m/n} Z = A^T A - B^T B, \tag{2}$$

where

$$H_{m/n} = \begin{bmatrix} I_m & \\ & -I_n \end{bmatrix} \in \mathcal{R}^{(m+n)\times(m+n)} \tag{3}$$

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 $H_{m/n}$*-pseudo Cholesky factor.* For convenience, from now on we will suppress $(\cdot)_{m/n}$ in H unless it is necessary.

2.1 Notations

An *H-pseudo inner product* is defined as

$$< \mathbf{u}, \mathbf{v} >_H = \mathbf{u}^T H \mathbf{v} = < \mathbf{v}, \mathbf{u} >_H, \quad \forall\, \mathbf{u},\ \mathbf{v} \in \mathcal{R}^{m+n}, \tag{4}$$

and an *H-pseudo vector norm* is defined as

$$||\mathbf{u}||_H = \sqrt{\mathbf{u}^T H \mathbf{u}}, \quad \forall\, \mathbf{u} \in \mathcal{R}^{m+n} \text{ such that } \quad \mathbf{u}^T H \mathbf{u} \geq 0. \tag{5}$$

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

$$\begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_p^T \end{bmatrix} H [\, \mathbf{q}_1\ \mathbf{q}_2\ \cdots\ \mathbf{q}_p \,] = \tag{6}$$

$$\begin{bmatrix} ||\mathbf{q}_1||_H^2 & < \mathbf{q}_1, \mathbf{q}_2 >_H & \cdots & < \mathbf{q}_1, \mathbf{q}_p >_H \\ < \mathbf{q}_1, \mathbf{q}_2 >_H & ||\mathbf{q}_2||_H^2 & \cdots & < \mathbf{q}_2, \mathbf{q}_p >_H \\ \vdots & \vdots & \ddots & \vdots \\ < \mathbf{q}_1, \mathbf{q}_p >_H & < \mathbf{q}_2, \mathbf{q}_p >_H & \cdots & ||\mathbf{q}_p||_H^2 \end{bmatrix} = I.$$

Equivalently, we can say that Q has H-pseudo orthogonal columns; which means that for any two columns of Q, their H-inner product satisfies

$$< \mathbf{q}_i, \mathbf{q}_j >_H = \begin{cases} 1 & \text{, if } i = j. \\ 0 & \text{, otherwise.} \end{cases} \tag{7}$$

2.2 Pseudo Orthogonalization Algorithms

We denote an *H-pseudo orthogonal decomposition* of Z as

$$Z = QR, \tag{8}$$

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

$$Z^T H Z = R^T Q^T H Q R = R^T R. \tag{9}$$

Rewriting eqn. (8) as

$$[\, \mathbf{z}_1 \, \mathbf{z}_2 \, \cdots \, \mathbf{z}_p \,] = [\, \mathbf{q}_1 \, \mathbf{q}_2 \, \cdots \, \mathbf{q}_p \,] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} \\ & r_{22} & \cdots & r_{2p} \\ & & \ddots & \vdots \\ & & & r_{pp} \end{bmatrix}, \tag{10}$$

leads to

$$\mathbf{z}_i = r_{1i}\mathbf{q}_1 + r_{2i}\mathbf{q}_2 + \cdots + r_{ii}\mathbf{q}_i = \sum_{j=1}^{i} r_{ji}\mathbf{q}_j, \qquad i = 1, \ldots, p \tag{11}$$

Premultiplying $\mathbf{z}_1^T H$ on the left-hand-side and $r_{11}\mathbf{q}_1^T H$ on the right-hand-side to the first equation of eqn. (11), and also noticing that $||\mathbf{q}_1||_H^2 = 1$, we have

$$r_{11} = ||\mathbf{z}_1||_H \tag{12}$$

and

$$\mathbf{q}_1 = \mathbf{z}_1 / r_{11}. \tag{13}$$

By continuing this procedure, at the $i-th$ step, $i = 2, \ldots, p$, all of the i nonzero elements in the $i-th$ *subcolumn* of R can be computed by pseudo correlating the $i-th$ equation in eqn. (11) with $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{i-1}$ respectively, i.e.,

$$r_{ji} =< \mathbf{q}_j, \mathbf{z}_i >_H, \qquad j = 1, \ldots, i-1 \tag{14}$$

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

$$r_{ii} = ||\mathbf{z}_i - r_{1i}\mathbf{q}_1 - \cdots - r_{i-1,i}\mathbf{q}_{i-1}||_H \tag{15}$$

and

$$\mathbf{q}_i = (\mathbf{z}_i - r_{1i}\mathbf{q}_1 - \cdots - r_{i-1,i}\mathbf{q}_{i-1})/r_{ii}. \tag{16}$$

Therefore, a Gram-Schmidt pseudo orthogonalization (GSPO) or hyperbolic Gram-Schmidt (HGS) 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
        $r_{ij} =< \mathbf{q}_i, \mathbf{z}_j >_H$ ;
        $\mathbf{t} = \mathbf{t} - r_{ij}\mathbf{q}_i$ ;
    end;
    $r_{jj} = \|\mathbf{t}\|_H$ ;
    $\mathbf{q}_j = \mathbf{t}/r_{jj}$ ;
end.
```

To derive the modified Gram-Schmidt pseudo orthogonalization (MGSPO), which is numerically stable, we first take the pseudo inner products of every equations in eqn. (11) with $\mathbf{z}_1$ and also notice that $< \mathbf{q}_i, \mathbf{q}_j >_H = \delta_{ij}$, so we have $\|\mathbf{z}_1\|_H^2 = r_{11}^2$, $\mathbf{q}_1 = \mathbf{z}_1/r_{11}$ and

$$< \mathbf{z}_1, \mathbf{z}_i >_H = r_{11}r_{1i}, \quad i = 2, \ldots, p. \tag{17}$$

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

$$\hat{\mathbf{z}}_i = \mathbf{z}_i - r_{1i}\mathbf{q}_1 = \sum_{j=2}^{i} r_{ji}\mathbf{q}_j. \tag{18}$$

Similarly, the second row of $R, r_{22}, \ldots, r_{2p}$, and the second column of $Q, \mathbf{q}_2$, are ready to compute. Continuing this procedure, R and Q can be fully known. This process is essentially a *modified Gram-Schmidt* [3,4,23] 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 Q is determined column by column. A modified Gram-Schmidt pseudo orthogonalization (MGSPO) or hyperbolic modified Gram-Schmidt (HMGS) algorithm is thus given as follows:

Algorithm 2 (MGSPO (I)) *[row-wise]*

```
for $i = 1, \cdots, p$, do
    $r_{ii} = \|\mathbf{z}_i\|_H$ ;
    $\mathbf{q}_i = \mathbf{z}_i / r_{ii}$ ;
    for $j = i+1, \cdots, p$, do
        $r_{ij} = <\mathbf{q}_i, \mathbf{z}_j>_H$ ;
        $\mathbf{z}_j = \mathbf{z}_j - r_{ij}\mathbf{q}_i$ ;
    end;
end.
```

This algorithm requires about $(m+n)p^2/2$ multiply-and-add, p 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 [10], can also be derived in a similar way, and is given as follows:

Algorithm 3 (MGSPO (II)) *[column-wise]*

```
for $j = 1, \cdots, p$, do
    for $i = 1, \cdots, j-1$, do
        $r_{ij} = < \mathbf{q}_i, \mathbf{z}_j >_H$ ;
        $\mathbf{z}_j = \mathbf{z}_j - r_{ij}\mathbf{q}_i$ ;
    end;
    $r_{jj} = \|\mathbf{z}_j\|_H$ ;
    $\mathbf{q}_j = \mathbf{z}_j / r_{jj}$ ;
end.
```

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

Theorem 1 (Pseudo Orthogonal Decomposition) *For any $Z \in \mathcal{R}^{(m+n)\times p}$, $H = \begin{bmatrix} I_m & \\ & -I_n \end{bmatrix}$, an H-pseudo orthogonal decomposition of Z, $Z = QR$, 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 Q, if and only if $Z^T H Z$ is positive definite.*

The proof is omitted. A similar theorem for an orthogonal QR decomposition can be found in [8, pp. 217].

2.3 Square-Root-Free Triangularization Algorithms

For many computations, especially in VLSI implementation, it is advantageous to reduce the number of square roots operations or even eliminate them altogether. To this end, we can pull out the inverse of the diagonal elements of R and decompose R into

$$R = D_R^{1/2} R_D \tag{19}$$

$$= \begin{bmatrix} 1/r_{11} & & & \\ & 1/r_{22} & & \\ & & \ddots & \\ & & & 1/r_{pp} \end{bmatrix} \begin{bmatrix} r_{11}^2 & r_{11}r_{12} & \cdots & r_{11}r_{1p} \\ & r_{22}^2 & \cdots & r_{22}r_{2p} \\ & & \ddots & \vdots \\ & & & r_{pp}^2 \end{bmatrix}.$$

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 $1 \leq i \leq p; i \leq j \leq p$. It is well known [17,4] that the optimum coefficient vector of a least squares problem can be computed by firstly upper triangularizing the augmented data matrix via orthogonalization and then followed by performing back substitution on the upper triangular matrix. The optimum coefficient vector will not be affected if the upper triangular matrix (Cholesky factor) is multiplied by any nonzero diagonal matrix. In our case, if only the least squares solution is of interest, then it makes no difference in obtaining R or R_D, while the latter can be computed with no square roots operations. A square-root-free MGSPO procedure [4] to obtain R_D, is given below.

Algorithm 4 (Sqrt-Free MGSPO)

```
for $i = 1, \ldots, p$, do
    for $j = i, \ldots, p$, do
        $r_{ii}r_{ij} = < \mathbf{z}_i, \mathbf{z}_j >_H$ ;
        $\mathbf{z}_j = \mathbf{z}_j - (r_{ii}r_{ij}/r_{ii}^2)\mathbf{z}_i$ ;
    end;
end.
```

For some applications, a complete knowledge of R is desired. In these cases, R can be computed from R_D by dividing each row of R_D by the square root of its leading diagonal element.

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 [8,19,20,22]. Here we only consider the Gram-Schmidt methods. Eqn. (1) now becomes

$$Z = \begin{bmatrix} R \\ D \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} \\ & r_{22} & \cdots & r_{2p} \\ & & \ddots & \vdots \\ & & & r_{pp} \\ \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_p \end{bmatrix} \in \mathcal{R}^{(p+n)\times p}, \tag{20}$$

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 $\tilde{R}^T\tilde{R} = Z^T H Z = R^T R - D^T D$. The signature matrix becomes $H_{p/n}$. If we denote $Z^{(0)} = Z$, then the sqrt-free MGSPO can be rewritten as follows:

```
for $i = 1, \cdots, p$, do
    $\tilde{r}_{ii}^2 = \|\mathbf{z}_i^{(i-1)}\|_H^2$ ;
    for $j = i+1, \cdots, p$, do
        $\tilde{r}_{ii}\,\tilde{r}_{ij} = < \mathbf{z}_i^{(i-1)}, \mathbf{z}_j^{(i-1)} >_H$ ;
        $\mathbf{z}_j^{(i)} = \mathbf{z}_j^{(i-1)} - \frac{\tilde{r}_{ij}\tilde{r}_{ii}}{\tilde{r}_{ii}^2}\,\mathbf{z}_i^{(i-1)}$ ;
    end;
end.
```

Notice that

$$Z^{(i)} \equiv [\ \mathbf{z}_{i+1}^{(i)}\ \cdots ; \mathbf{z}_p^{(i)}\ ] \in \mathcal{R}^{(p+n)\times(p-i)}\ ,\quad i = 1,\ \cdots,\ p-1\ . \tag{21}$$

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

$$\sum_{i=1}^{p}[(n+i)(p-i+1)+(n+i)(p-i)] = \frac{1}{3}[p^3+3np^2+\frac{3p^2}{2}+\frac{p}{2}]. \tag{22}$$

This arises from the computational load in obtaining the lower right $\widetilde{R}$. 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 $\widetilde{R}$. 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 Appendix one step by step.

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

$$\sum_{i=1}^{p} \left\{ n^2 + n + (n^2 + n) + \left[\sum_{j=i+1}^{p} (n + n) \right] \right\} = np(p + 2n + 1) \qquad (23)$$

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 a conventional HGR [1](excluding fast sqrt-free Givens methods), it can be shown that it needs

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

multiply-and-add, np square-root and $2np$ division operations, or, $\mathcal{O}(np^2 + 1.5np)$ flops. HHT [22] requires

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

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.

4 Simultaneously Up/Down-dating Multiple Rows

Similar to the hyperbolic Householder transformations(HHT) proposed by Rader and Steinhardt [22], our MGS pseudo orthogonalization can also perform rank-k updating and rank-ℓ downdating *simultaneously* as needed in sliding-window and/or robust least squares problems.

Let

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

and

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

then an algorithm for simultaneously up/down-dating the Cholesky factor R can be derived in the similar way. All we need to do is to slightly modify the initialization in Algorithm 5 as follows: replacing $\mathbf{d}_i$ and I_n with the $i-$th column of $\begin{bmatrix} X_{new} \\ X_{old} \end{bmatrix}$ and $\begin{bmatrix} I_k & 0 \\ 0 & -I_\ell \end{bmatrix}$ respectively. After X_{new} and X_{old} are zeroed out in an $H-$pseudo orthogonal manner, the new Cholesky factor $\widetilde{R}$ will satisfy

$$\widetilde{R}^T\widetilde{R} = R^TR + X_{new}^TX_{new} - X_{old}^TX_{old}$$

as indicated previously.

5 Block Systolic Triarray Using MGSPO

Suppose under sliding window scheme, we want to up/downdate the Cholesky factor as the vector-valued new and old data of fixed size $k = \ell$ are available. We can either perform updating and downdating alternatively or simultaneously. Let us take the simultaneous case as an example and show this can be done in a block systolic triarray.

It is well known that systolic arrays possess many desirable properties in VLSI implementations [15,16]. Gentleman and Kung [7] first proposed a systolic triarray to perform QRD updating using Givens rotations. Later Liu, Hsieh and Yao [19] proposed a similar stucture using Householder transformations. Based on these, we propose another systolic structure using the reformulated modified Gram-Schmidt methods to up/down-date the Cholesky factor.

For simplicity, we will focus on recursively up/down-dating $R_{\ D}$, the sqrt-free version, instead of the Cholesky factor R itself. Suppose $\bar{r}_{ii} = r_{ii}^2$ and $\bar{r}_{ij} = r_{ii}r_{ij}$ in eqn. (19). Fig.1 is a block MGSPO systolic array without square roots. In this array, we assume all the local memory cells in the processors store $\bar{r}_{ii}$ and $\bar{r}_{ij}$, the corresponding elements of $R_{\ D}$ in eqn. (19) Before the new and old data blocks get in the triarray, they need to be skewed and interleaved, i.e., each processor will fetch from its above 2 data blocks of the same size k, one new and the other old. From Algorithm 5 and Section 4, we can see that the initialization corresponds to fetching data blocks from the above; the first nested loop (loop i) in the recursion corresponds to the job of a boundary processor; and the second nested loop (loop j) to that of a regular processor. By further noting that this triarray store $R_{\ D}$, we can thus formulate the required operations in the processors. The boundary and regular processor elements (PEs) of a block MGSPO systolic array without square-roots work as follows:

Boundary PE:

$$\begin{aligned}
\mathbf{g}_i &= G_{i-1}\mathbf{d}_i; \\
G_i &= G_{i-1} - \frac{\mathbf{g}_i\mathbf{g}_i^T}{\bar{r}_{ii}}; \\
\bar{r}'_{ii} &= \bar{r}_{ii} - \mathbf{d}_i^T\mathbf{g}_i.
\end{aligned}$$

Regular PE:

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

6 Conclusions

We have shown the derivation of a new class of hyperbolic Gram-Schmidt methods to pseudoly orthogonalize a data matrix with applications in sliding window RLS filtering problems and rejecting spurious data which is very important in time-varying signal processing. To facilitate vector-pipelined processing, all data is considered as vector-valued. However, the inherent computational load imbalance for the Gram-Schmidt method in up/down-dating the Cholesky factor impedes its application under massive data computation. This is the first successful effort in tacitly uniformly redistributing the computational load and greatly reducing the computational cost. By taking the advantages of parallel computation, a Gram-Schmidt systolic array structure is proposed and proved to be possible in up/down-dating Cholesky factorization. We also compare our methods with the existing methods, such as hyperbolic Householder transformation and hyperbolic Givens rotation. Because

Gram-Schmidt methods can be sqrt-free, it is very promising in adaptive signal processing and also in VLSI implementation.

Appendix

We will derive the algorithm **5** in details here.

Step 1.

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

$$\Downarrow \{\tilde{r}_{11}, \cdots, \tilde{r}_{1p}\} \tag{24}$$

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

$\{\tilde{r}_{11}, \cdots, \tilde{r}_{1p}\}$ 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:

$$\tilde{r}_{11}^2 = r_{11}^2 - \mathbf{d}_1^{(0)^T}\mathbf{d}_1^{(0)}, \tag{25}$$

$$\tilde{r}_{11}\tilde{r}_{1j} = r_{11}r_{1j} - \mathbf{d}_1^{(0)^T}\mathbf{d}_j^{(0)}, \quad j = 2, \cdots, p, \tag{26}$$

$$\mathbf{d}_j^{(1)} = \mathbf{d}_j^{(0)} - \frac{\tilde{r}_{11}\tilde{r}_{1j}}{\tilde{r}_{11}^2}\mathbf{d}_1^{(0)}, \quad j = 2, \cdots, p, \tag{27}$$

$$r_{1j}^{(1)} = r_{1j} - \frac{\tilde{r}_{11}\tilde{r}_{1j}}{\tilde{r}_{11}^2} r_{11}, \quad j = 2, \cdots, p. \tag{28}$$

What we need now is to express $r_{1j}^{(1)}$ in eqn. (28) in terms of $\mathbf{d}_j^{(1)}$ in eqn. (27). With eqn. (25) and (26) substituting in eqn. (28), it can be shown that

$$r_{1j}^{(1)} = \frac{1}{r_{11}}\mathbf{d}_1^{(0)T}\mathbf{d}_j^{(1)} = \mathbf{f}_1^T\mathbf{d}_j^{(1)}, \quad j = 2, \cdots, p, \tag{29}$$

with

$$\mathbf{f}_1 \equiv \frac{1}{r_{11}}\mathbf{d}_1^{(0)}. \tag{30}$$

Step 2.

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

$$\Downarrow \{\tilde{r}_{22}, \cdots, \tilde{r}_{2p}\} \tag{31}$$

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

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

$$\begin{aligned}
\tilde{r}_{22}^2 &= r_{22}^2 - \mathbf{d}_2^{(1)^T}\mathbf{d}_2^{(1)} + \mathbf{d}_2^{(1)^T}\mathbf{f}_1\mathbf{f}_1^T\mathbf{d}_2^{(1)} \\
&= r_{22}^2 - \mathbf{d}_2^{(1)^T}(I - \mathbf{f}_1\mathbf{f}_1^T)\mathbf{d}_2^{(1)}, \qquad (32)
\end{aligned}$$

$$\begin{aligned}
\tilde{r}_{22}\tilde{r}_{2j} &= r_{22}r_{2j} - \mathbf{d}_2^{(1)^T}\mathbf{d}_j^{(1)} + \mathbf{d}_2^{(1)^T}\mathbf{f}_1\mathbf{f}_1^T\mathbf{d}_j^{(1)} \\
&= r_{22}r_{2j} - \mathbf{d}_2^{(1)^T}(I - \mathbf{f}_1\mathbf{f}_1^T)\mathbf{d}_j^{(1)}, j=3,\cdots,p, \qquad (33)
\end{aligned}$$

$$\begin{aligned}
\mathbf{d}_j^{(2)} &= \mathbf{d}_j^{(1)} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}\mathbf{d}_2^{(1)}\ ,\ \ j = 3,\cdots,p, \qquad (34) \\
r_{1j}^{(2)} &= r_{1j}^{(1)} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}r_{12}^{(1)} = \mathbf{f}_1^T(\mathbf{d}_j^{(1)} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}\mathbf{d}_2^{(1)}) \\
&= \mathbf{f}_1^T\mathbf{d}_j^{(2)}\ ,\ j = 3,\cdots,p, \qquad (35) \\
r_{2j}^{(2)} &= r_{2j} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}r_{22}\ ,\ \ j = 3,\cdots,p. \qquad (36)
\end{aligned}$$

Again, by using eqn. (32, 33, 34), $r_{2j}^{(2)}$ in eqn. (36) can be expressed in terms of $\mathbf{d}_j^{(2)}$ in eqn. (34), namely,

$$\begin{aligned}
r_{2j}^{(2)} &= \frac{1}{r_{22}}[r_{22}r_{2j} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}r_{22}^2] \\
&= \frac{1}{r_{22}}\{\tilde{r}_{22}\tilde{r}_{2j} + \mathbf{d}_2^{(1)^T}(I - \mathbf{f}_1\mathbf{f}_1^T)\mathbf{d}_j^{(1)} \\
&\quad - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}[\tilde{r}_{22}^2 + \mathbf{d}_2^{(1)^T}(I - \mathbf{f}_1\mathbf{f}_1^T)\mathbf{d}_2^{(1)}]\} \\
&= \frac{1}{r_{22}}[\mathbf{d}_2^{(1)^T}(I - \mathbf{f}_1\mathbf{f}_1^T)][\mathbf{d}_j^{(1)} - \frac{\tilde{r}_{22}\tilde{r}_{2j}}{\tilde{r}_{22}^2}\mathbf{d}_2^{(1)}] \\
&= \frac{1}{r_{22}}\mathbf{d}_2^{(1)^T}(I - \mathbf{f}_1\mathbf{f}_1^T)\mathbf{d}_j^{(2)} \\
&= \mathbf{f}_2^T\mathbf{d}_j^{(2)},\ j = 3,\cdots,p, \qquad (37)
\end{aligned}$$

with

$$\mathbf{f}_2 \equiv \frac{1}{r_{22}}(I - \mathbf{f}_1\mathbf{f}_1^T)\mathbf{d}_2^{(1)}. \qquad (38)$$

Step 3.

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

$$\Downarrow \{\tilde{r}_{33}, \cdots, \tilde{r}_{3p}\} \tag{39}$$

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

$\{\tilde{r}_{33}, \cdots, \tilde{r}_{3p}\}$ 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 $\{\mathbf{d}_4^{(3)}, \cdots, \mathbf{d}_p^{(3)}\}$ are updated as follows:

$$\begin{aligned} \tilde{r}_{33}^2 &= r_{33}^2 - \mathbf{d}_3^{(2)^T}\mathbf{d}_3^{(2)} + \mathbf{d}_3^{(2)^T}\mathbf{f}_1\mathbf{f}_1^T\mathbf{d}_3^{(2)} + \mathbf{d}_3^{(2)^T}\mathbf{f}_2\mathbf{f}_2^T\mathbf{d}_3^{(2)} \\ &= r_{33}^2 - \mathbf{d}_3^{(2)^T}(I - \mathbf{f}_1\mathbf{f}_1^T - \mathbf{f}_2\mathbf{f}_2^T)\mathbf{d}_3^{(2)}, \end{aligned} \tag{40}$$

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

$$
\begin{aligned}
&+\mathbf{d}_3^{(2)^T}\mathbf{f}_2\mathbf{f}_2^T\mathbf{d}_j^{(2)} \\
&= r_{33}r_{3j} - \mathbf{d}_3^{(2)^T}(I - \mathbf{f}_1\mathbf{f}_1^T - \mathbf{f}_2\mathbf{f}_2^T)\mathbf{d}_j^{(2)} \qquad (41)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{d}_j^{(3)} &= \mathbf{d}_j^{(2)} - \frac{\tilde{r}_{3j}}{\tilde{r}_{33}}\mathbf{d}_3^{(2)}, \quad j = 4, \cdots, p, \qquad (42) \\
r_{1j}^{(3)} &= \mathbf{f}_1^T(\mathbf{d}_j^{(2)} - \frac{\tilde{r}_{3j}}{\tilde{r}_{33}}\mathbf{d}_3^{(2)}) \\
&= \mathbf{f}_1^T\mathbf{d}_j^{(3)}, \ j = 4, \cdots, p, \qquad (43) \\
r_{2j}^{(3)} &= \mathbf{f}_2^T(\mathbf{d}_j^{(2)} - \frac{\tilde{r}_{3j}}{\tilde{r}_{33}}\mathbf{d}_3^{(2)}) \\
&= \mathbf{f}_2^T\mathbf{d}_j^{(3)}, \ j = 4, \cdots, p, \qquad (44) \\
r_{3j}^{(3)} &= r_{3j} - \frac{\tilde{r}_{3j}}{\tilde{r}_{33}}r_{33}, \quad j = 4, \cdots, p. \qquad (45)
\end{aligned}
$$

Now, $r_{3j}^{(3)}$ in eqn. (45) can be written as

$$
\begin{aligned}
r_{3j}^{(3)} &= \frac{1}{r_{33}}\mathbf{d}_3^{(2)^T}(I - \mathbf{f}_1\mathbf{f}_1^T - \mathbf{f}_2\mathbf{f}_2^T)\,\mathbf{d}_j^{(3)} \qquad (46) \\
&= \mathbf{f}_3^T\mathbf{d}_j^{(3)}, \ j = 4, \cdots, p, \qquad (47)
\end{aligned}
$$

where

$$
\mathbf{f}_3 \equiv \frac{1}{r_{33}}(I - \mathbf{f}_1\mathbf{f}_1^T - \mathbf{f}_2\mathbf{f}_2^T)\mathbf{d}_3^{(2)}. \qquad (48)
$$

Proceeding in this way, it can be shown that

Step i

$$
\tilde{r}_{ii}^2 = r_{ii}^2 - \mathbf{d}_i^{(i-1)^T}(I - \mathbf{f}_1\mathbf{f}_1^T - \cdots - \mathbf{f}_{i-1}\mathbf{f}_{i-1}^T)\mathbf{d}_i^{(i-1)}, \qquad (49)
$$

$$
\tilde{r}_{ii}\tilde{r}_{ij} = r_{ii}r_{ij} - \mathbf{d}_i^{(i-1)^T}(I - \mathbf{f}_1\mathbf{f}_1^T - \cdots - \mathbf{f}_{i-1}\mathbf{f}_{i-1}^T)\mathbf{d}_j^{(i-1)}, \qquad (50)
$$

$$
\mathbf{d}_j^{(i)} = \mathbf{d}_j^{(i-1)} - \frac{\tilde{r}_{ij}}{\tilde{r}_{ii}}\mathbf{d}_i^{(i-1)}, \qquad (51)
$$

$$
r_{kj}^{(i)} = \mathbf{f}_k^T\mathbf{d}_j^{(i)} \quad k = 1, \cdots, i; \quad j = i+1, \cdots, p, \qquad (52)
$$

$$
\mathbf{f}_i \equiv \frac{1}{r_{ii}}(I - \mathbf{f}_1\mathbf{f}_1^T - \cdots - \mathbf{f}_{i-1}\mathbf{f}_{i-1}^T)\mathbf{d}_i^{(i-1)}. \qquad (53)
$$

By further defining two new quantities,

$$\begin{aligned} \mathbf{g}_i &= r_{ii}\mathbf{f}_i \in \mathcal{R}^n, \;\; i = 1, \cdots, p \qquad (54) \\ G_i &= I - \mathbf{f}_1\mathbf{f}_1^T - \cdots - \mathbf{f}_i\mathbf{f}_i^T \;\; i = 1, \cdots, p, \qquad (55) \end{aligned}$$

with $G_0 \equiv I_n$, a recursion of formula can be easily derived from eqn. (54) and (53),

$$\begin{aligned} \mathbf{g}_i &= G_{i-1}\mathbf{d}_i^{(i-1)}, \qquad (56) \\ G_i &= G_{i-1} - \mathbf{f}_i\mathbf{f}_i^T = G_{i-1} - \frac{\mathbf{g}_i\mathbf{g}_i^T}{r_{ii}^2} \in \mathcal{R}^{n \times n}, \qquad (57) \end{aligned}$$

hence eqn. (49) and (50) become

$$\begin{aligned} \tilde{r}_{ii}^2 &= r_{ii}^2 - \mathbf{d}_i^{(i-1)^T}\mathbf{g}_i \, , \qquad (58) \\ \tilde{r}_{ii}\tilde{r}_{ij} &= r_{ii}r_{ij} - \mathbf{d}_j^{(i-1)^T}\mathbf{g}_i \, . \qquad (59) \end{aligned}$$

References

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

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

[3] Å. Björck, "Solving least squares problems by Gram-Schmidt orthogonalization," *BIT 7*, pp. 1–21, 1967.

[4] Å. Björck, "Least Squares Methods" in **Handbook of Numerical Analysis Vol. II:** Finite difference methods - Solution of equations in R^n. Elsevier North Holland, 1989.

[5] J. M. Cioffi, "The fast adaptive ROTOR's RLS algorithm," *IEEE Trans. Accoust., Speech, Signal Processing*, Vol. 38, No. 4, pp. 631–653, Apr. 1990.

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

[7] W. M. Gentleman and H. T. Kung, "Matrix triangularization by systolic array," *Proc. SPIE,* Vol. 298: Real-time signal processing IV, pp. 19–26, 1981.

[8] G. H. Golub and C. F. Van Loan, **Matrix computations,** Johns Hopkins Press, 2nd ed., Baltimore, MD, 1989.

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

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

[11] S. F. Hsieh and 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.

[12] S. F. Hsieh and K. Yao, "Systolic implementation of windowed recursive LS estimation," *Proc. of IEEE Int'l Symp. on CAS,* pp. 1931–1934, May 1990.

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

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

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

[16] S. Y. Kung, **VLSI array processros,** Prentice-Hall, Englewood Cliffs, NJ, 1988.

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

[18] 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*, Vol. ASSP-34, No. 4, pp. 829–836, Aug. 1986,

[19] K. J. R. Liu, S. F. Hsieh, and K. Yao, "Two-level pipelined implementation of systolic block Householder transformations with application to RLS algorithm," *Proc. Int'l Conf. on Application-Specific Array Processors,* pp.758–769, Princeton, Sep. 1990.

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

[21] W. Murray, P. E. Gill, G. H. Golub, and M. A. Saunders, "Methods for modifying matrix factorizations," *Mathematics of Computation,* Vol. 28, No. 126, pp. 505–535, Apr. 1974.

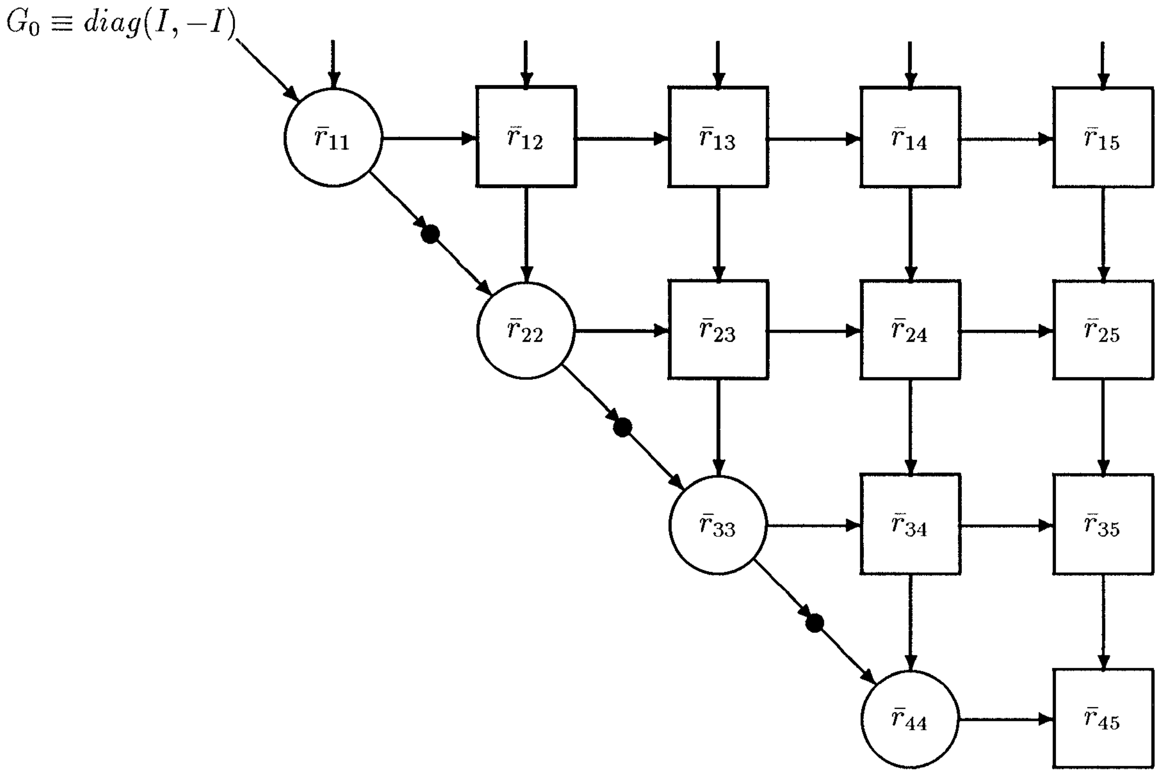
[22] C. M. Rader and A. O. Steinhardt, "Hyperbolic Householder transformations," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-34, No. 6, pp. 1589–1602, Dec. 1986.

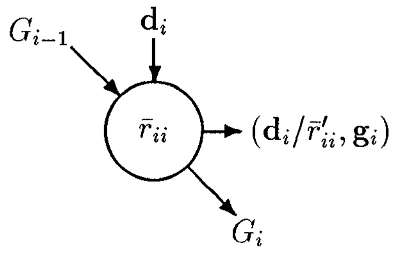
[23] J. R. Rice, "Experiments on Gram-Schmidt orthogonalization," *Math. Comp. 20*, pp. 325–328, 1966.

[24] R. Schreiber, "Implementation of adaptive array algorithms," *IEEE Trans. on Acoust., Speech, Signal Processing*, Vol. ASSP-34, pp. 338–346, Mar. 1986.

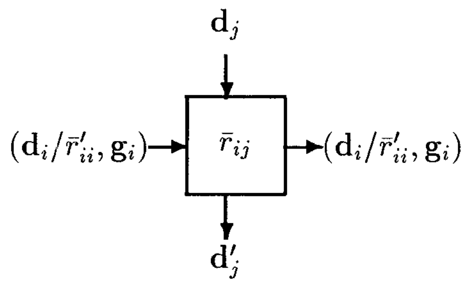
[25] P. Strobach, **Linear prediction theory: A mathematical basis for adaptive systems**, Springer-Verlag, 1990.

[26] N.-K. Tsao, "A note on implementing the Householder transformation," *SIAM J. Numer. Anal.*, Vol. 12, No. 1, pp. 53–58, Mar. 1975.

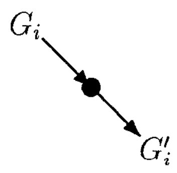




$$\mathbf{g}_i = G_{i-1}\mathbf{d}_i \; ;$$
$$G_i = G_{i-1} - \mathbf{g}_i\mathbf{g}_i^T/\bar{r}_{ii} \; ;$$
$$\bar{r}'_{ii} = \bar{r}_{ii} - \mathbf{d}_i^T\mathbf{g}_i \; .$$


$\mathbf{d}_j$
$(\mathbf{d}_i/\bar{r}'_{ii}, \mathbf{g}_i)$
$\bar{r}_{ij}$
$(\mathbf{d}_i/\bar{r}'_{ii}, \mathbf{g}_i)$
$\mathbf{d}'_j$

$$\bar{r}'_{ij} = \bar{r}_{ij} - \mathbf{d}_j^T\mathbf{g}_i \; ;$$
$$\mathbf{d}'_j = \mathbf{d}_j - \bar{r}'_{ij}\mathbf{d}_i/\bar{r}'_{ii} \; .$$



$$G'_i = G_i \; .$$

Figure 1: Block MGSPO systolic array without square roots