Reducing the Computational Complexity for BLAST by Using a Novel Fast Algorithm to Compute an Initial Square-root Matrix

Hufei Zhu[1], Wen Chen[2], Dageng Chen[1], Yinggang Du[1], and Jianmin Lu[1]
[1]Wireless Research Dept., Huawei Technologies Co., Ltd. Email: {zhuhufei;dagengchen;duyinggang;lujianmin}@huawei.com
[2]Dept. of Electronic Engineering, Shanghai Jiao Tong Univ. Email: wenchen@sjtu.edu.cn

***Abstract*—We propose a fast algorithm to compute an initial triangular square-root of the estimation error covariance matrix for BLAST, which are then applied to develop a square-root algorithm for BLAST. The speedups of our square-root BLAST algorithm over the previous square-root BLAST algorithm in the number of multiplications and additions are 3.78-5.8 and 3.95-5 respectively, and the ratios between the computational complexity of our BLAST algorithm and that of the linear MMSE detection algorithm in the number of multiplications and additions are 1.10-0.71 and 0.90-0.71 respectively, which means that for the first time, the nonlinear MMSE BLAST detector with successive interference cancellation can have even lower complexity than the linear MMSE detector. Moreover, our BLAST algorithm is also numerically stable and hardware friendly, since it uses unitary transformations to avoid the matrix inversions, and gets the initial square-root which is equivalent to a Cholesky factor of the estimation error covariance matrix.**

I. Introduction

Bell Labs Layered Space-Time architecture (BLAST), including the most practical version- vertical BLAST (V-BLAST), can achieve very high spectral efficiency in rich multipath environments through exploiting the extra spatial dimension [1]. However, the required computational complexity is quite high. Recently several fast algorithms have been proposed for efficient implementation of V-BLAST [2]–[7]. The square-root algorithm in [2] can reduce the computational load of the conventional V-BLAST by $0.7M$, where M is the number of transmit antennas. The speedups of the recursive algorithm in [3] over the algorithm in [2] in the number of multiplications and additions are 1.54 and 1.89 respectively [5]. The improved square-root algorithm in [4] speeds up the original square-root algorithm in [2] by 1.36. And the speedups of the improved recursive algorithm in [5] over the original recursive algorithm in [3] in the number of multiplications and additions are 1.47 and 1.2 respectively. In recent published [6] Shang-Xia gave the "fastest known algorithm" by incorporating the improvements in [5], [7] into the original fast recursive algorithm [3], and proposed a further improvement.

In this paper we will further improve the algorithm in [4]. We propose a fast algorithm to compute an initial upper triangular matrix $\mathbf{P}^{1/2}$, a square-root [2] of the estimation error covariance matrix $\mathbf{P}$. Notice that $\mathbf{P}^{1/2}$ is also equivalent to a Cholesky factor of $\mathbf{P}$. Then we develop a V-BLAST detection algorithm, which is faster than all existing algorithms in [2]–[7].

The V-BLAST system and its basic detection algorithm are overviewed in section II, followed by the description of the previous square-root algorithm in [4] in Section III. A fast algorithm to compute an initial triangular $\mathbf{P}^{1/2}$ is proposed in Section IV. Then it is applied to develop the novel algorithm for BLAST detection in Section V. The complexity of the presented V-BLAST detection algorithm is evaluated in Section VI. Finally, we make conclusion in Section VII.

In this paper, $(\bullet)^T$, $(\bullet)^*$, and $(\bullet)^H$ denote matrix transposition, matrix conjugate, and matrix conjugate transposition, respectively. $\mathbf{0}_M$ is the $M \times 1$ zero vector, I_M is the identity matrix with size M, and $P_{i,i}$ is the i^{th} diagonal entry of $\mathbf{P}$.

II. V-BLAST System Model and Detection

The considered V-BLAST system consists of M transmit antennas and $N (\geq M)$ receive antennas in a rich-scattering and flat-fading wireless channel. At the transmitter, the data stream is de-multiplexed into M streams, and each sub-stream is encoded and fed to its respective transmit antenna. Each receive antenna receives the signals from all M transmit antennas. Let $\mathbf{a} = [a_1, a_2, \cdots, a_M]^T$ denote the vector of transmit symbols from M antennas and assume $E(\mathbf{a}\mathbf{a}^H) = \sigma_{\mathbf{a}}^2 \mathrm{I}_M$. Then the received signal is given by

$$\mathbf{r} = \mathbf{H} \cdot \mathbf{a} + \mathbf{w}, \tag{1}$$

where $\mathbf{H} = [\mathbf{H}_1^T \cdots \mathbf{H}_N^T]^T = [\mathbf{h}_1 \cdots \mathbf{h}_M]$ is the $N \times M$ complex channel matrix with statistically independent entries, with $\mathbf{H}_n$ and $\mathbf{h}_m$ to be the n^{th} row and the m^{th} column of $\mathbf{H}$ respectively. $\mathbf{w}$ is the complex zero-mean Gaussian noise vector with covariance $\sigma_w^2 \mathrm{I}_N$. Let $\alpha = \sigma_w^2 / \sigma_a^2$. Then the linear minimum mean squared error (MMSE) detection of $\mathbf{a}$ is

$$\hat{\mathbf{a}} = \mathbf{G} \cdot \mathbf{r} = (\mathbf{H}^H \cdot \mathbf{H} + \alpha \mathbf{I}_M)^{-1} \cdot \mathbf{H}^H \cdot \mathbf{r}, \tag{2}$$

where the MMSE filtering matrix $\mathbf{G} = \mathbf{P} \cdot \mathbf{H}^H$ and $\mathbf{P} = (\mathbf{H}^H \cdot \mathbf{H} + \alpha \mathbf{I}_M)^{-1}$. It is easy to show that $\mathbf{P}$ is the covariance matrix for the detection error $\mathbf{a} - \hat{\mathbf{a}}$ [2]. Denote the i^{th} row of $\mathbf{G}$ by $\mathbf{G}_i$. Then the i^{th} element of $\hat{\mathbf{a}}$ is $\hat{a}_i = \mathbf{G}_i \cdot \mathbf{r}$, where $\mathbf{G}_i$ is usually referred to as the i^{th} MMSE nulling vector.

The conventional V-BLAST detection detects M elements of the transmit vector $\mathbf{a}$ iteratively with the optimal ordering.

According to (2), the best detected element of $\hat{\mathbf{a}}$ would be the one with the smallest error variance, i.e. the one for which the $P_{i,i}$ is the smallest [2], and it can be used to improve the detection of the remaining $M-1$ signals. Suppose that the order of the entries of the transmit vector $\mathbf{a}$ is arranged such that the best detected element is the M^{th} entry, and treat a_M as the correctly detected element. Then we obtain the following reduced order problem:

$$\mathbf{r}^{(M-1)} = \mathbf{r} - \mathbf{h}_M a_M = \mathbf{H}^{(M-1)}\mathbf{a}^{(M-1)} + \mathbf{w}, \qquad (3)$$

where $\mathbf{H}^{(M-1)} = [\mathbf{h}_1, \mathbf{h}_2 \cdots, \mathbf{h}_{M-1}]$ is the deflated channel matrix and $\mathbf{a}^{(M-1)} = [a_1, a_2, \cdots, a_{M-1}]^T$ is the reduced signal vector. Then the solution is given by

$$\hat{\mathbf{a}}^{(M-1)} = \mathbf{G}^{(M-1)}(\mathbf{r} - \mathbf{h}_M a_M) = \mathbf{G}^{(M-1)}\mathbf{r}^{(M-1)}, \qquad (4)$$

where $\mathbf{G}^{(M-1)} = \mathbf{P}^{(M-1)} \cdot \mathbf{H}^{(M-1)}$, and $\mathbf{P}^{(M-1)} \equiv \left(\left(\mathbf{H}^{(M-1)}\right)^H \cdot \mathbf{H}^{(M-1)} + \alpha \cdot \mathbf{I}_{(M-1)}\right)^{-1}$ is the corresponding error covariance matrix. The detection will proceed until all elements are detected.

III. The Previous Square-Root Algorithm

The square-root algorithms for V-BLAST [2], [4] use $\mathbf{P}^{1/2}$ instead of $\mathbf{P}$ to calculate the MMSE nulling vectors, where $\mathbf{P}^{1/2}$ can be identified as a square-root of $\mathbf{P}$, i.e.

$$\mathbf{P}^{1/2}\left(\mathbf{P}^{1/2}\right)^H = \mathbf{P}. \qquad (5)$$

Specifically, the previous square-root algorithm in [4] can be summarized as follows:

Initialization:

1) Compute the initial $\mathbf{P}^{1/2} = \mathbf{P}^{(M)/2}$.
Let $\mathbf{P}^{1/2}_{|0} = \frac{1}{\sqrt{\alpha}}\mathbf{I}_M$. Then form the $(M+1)\times(M+1)$ pre-array $\mathbf{\Omega}_{i-1} = \begin{bmatrix} 1 & \mathbf{H}_i\mathbf{P}^{1/2}_{|i-1} \\ \mathbf{0}_M & \mathbf{P}^{1/2}_{|i-1} \end{bmatrix}$ and propagate the square-root algorithm $\mathbf{\Omega}_{i-1}\cdot\mathbf{\Theta}_i = \begin{bmatrix} \times & \mathbf{0}_M^T \\ \times & \mathbf{P}^{1/2}_{|i} \end{bmatrix}$, till get $\mathbf{P}^{1/2}_{|N} = \mathbf{P}^{1/2}$, where $\mathbf{\Theta}_i$ is any unitary transformation that block lower triangularizes the pre-array $\mathbf{\Omega}_{i-1}$, and "$\times$" denotes irrelevant entries at this time. In addition, set the initial $m = M$, $\mathbf{r}^{(M)} = \mathbf{r}$, $\mathbf{a}^{(M)} = \mathbf{a}$, and $\mathbf{H}^{(M)} = \mathbf{H}$.

Recursion:

2) Find the minimum length row of $\mathbf{P}^{(m)/2}$ and permute it to be the last row. Permute $\mathbf{a}^{(m)}$ and $\mathbf{H}^{(m)}$ accordingly.
3) Find a unitary transformation $\mathbf{\Sigma}$ such that $\mathbf{P}^{(m)/2}\mathbf{\Sigma}$ is block-upper triangular, i.e.,

$$\mathbf{P}^{(m)/2}\mathbf{\Sigma} = \begin{bmatrix} \mathbf{P}^{(m-1)/2} & \mathbf{u}_{m-1} \\ \mathbf{0}_{m-1}^T & \lambda_m \end{bmatrix}, \qquad (6)$$

where $\mathbf{u}_{m-1}$ is a $(m-1)\times 1$ vector and λ_m is a scalar.
4) Form the least-mean-square estimate of a_m, i.e.,

$$\hat{a}_m = \lambda_m \left[\begin{array}{cc} (\mathbf{u}_{m-1})^H & (\lambda_m)^* \end{array}\right]\left(\mathbf{H}^{(m)}\right)^H \mathbf{r}^{(m)}. \qquad (7)$$

5) Obtain a_m from $\hat{a}_m$ via slicing.
6) Cancel the effect of a_m to the received signal $\mathbf{r}^{(m)}$ by

$$\mathbf{r}^{(m-1)} = \mathbf{r}^{(m)} - \mathbf{h}_m \cdot a_m, \qquad (8)$$

to get the above-mentioned reduced-order problem

$$\mathbf{r}^{(m-1)} = \mathbf{H}^{(m-1)} \cdot \mathbf{a}^{(m-1)} + \mathbf{w}. \qquad (9)$$

7) If $m > 1$, go back to step 2) with $\mathbf{r}^{(m-1)}$, $\mathbf{a}^{(m-1)}$, $\mathbf{H}^{(m-1)}$ and $\mathbf{P}^{(m-1)/2}$ instead of $\mathbf{r}^{(m)}$, $\mathbf{a}^{(m)}$, $\mathbf{H}^{(m)}$ and $\mathbf{P}^{(m)/2}$, respectively. Otherwise stop.

IV. An Fast Algorithm to Compute an Initial square-root Matrix For BLAST

By the square-root algorithm introduced in the previous section, the computational load is extremely high to compute the initial $\mathbf{P}^{1/2}$. So we propose a fast algorithm to compute an initial $\mathbf{P}^{1/2}$, where we get an upper triangular $\mathbf{P}^{1/2}$.

If $\mathbf{P}^{(m)/2}$ satisfies (5), any $\mathbf{P}^{(m)/2}\mathbf{\Sigma}$ also satisfies (5). Then it can be seen from (6) that there must be a square-root of $\mathbf{P}^{(m)}$ in the form of

$$\mathbf{P}^{(m)/2} = \begin{bmatrix} \mathbf{P}^{(m-1)/2} & \mathbf{u}_{m-1} \\ \mathbf{0}_{m-1}^T & \lambda_m \end{bmatrix}. \qquad (10)$$

By (6) we can compute $\mathbf{P}^{(m-1)/2}$ from $\mathbf{P}^{(m)/2}$. On the contrary, by (10) we can compute $\mathbf{P}^{(m)/2}$ from $\mathbf{P}^{(m-1)/2}$.

Let $\mathbf{R}^{(m)} = \left(\mathbf{P}^{(m)}\right)^{-1} = \left(\mathbf{H}^{(m)}\right)^H \cdot \mathbf{H}^{(m)} + \alpha\mathbf{I}_m$ $= [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_m]^H[\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_m] + \alpha\mathbf{I}_m =$

$$\begin{pmatrix} \mathbf{h}_1^H\mathbf{h}_1 + \alpha & \mathbf{h}_1^H\mathbf{h}_2 & \cdots & \mathbf{h}_1^H\mathbf{h}_m \\ \mathbf{h}_2^H\mathbf{h}_1 & \mathbf{h}_2^H\mathbf{h}_2 + \alpha & \cdots & \mathbf{h}_2^H\mathbf{h}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{h}_m^H\mathbf{h}_1 & \mathbf{h}_m^H\mathbf{h}_2 & \cdots & \mathbf{h}_m^H\mathbf{h}_m + \alpha \end{pmatrix}. \qquad (11)$$

Then

$$\begin{bmatrix} \mathbf{R}^{(m-1)} & \mathbf{v}_{m-1} \\ (\mathbf{v}_{m-1})^H & \beta_m \end{bmatrix} = \mathbf{R}^{(m)}, \qquad (12)$$

where

$$\mathbf{v}_{m-1} = \left[\begin{array}{cccc} \mathbf{h}_1^H\cdot\mathbf{h}_m & \mathbf{h}_2^H\cdot\mathbf{h}_m & \cdots & \mathbf{h}_{m-1}^H\cdot\mathbf{h}_m \end{array}\right]^T \qquad (13)$$

and

$$\beta_m = \mathbf{h}_m^H \cdot \mathbf{h}_m + \alpha. \qquad (14)$$

So we can compute $\mathbf{P}^{(m)/2}$ from $\mathbf{P}^{(m-1)/2}$ by (10), using λ_m and $\mathbf{u}_{m-1}$ as

$$\lambda_m\lambda_m^* = \frac{1}{\beta_m - \mathbf{v}_{m-1}^H\mathbf{P}^{(m-1)/2}\left(\mathbf{P}^{(m-1)/2}\right)^H\mathbf{v}_{m-1}} \qquad (15)$$

and

$$\mathbf{u}_{m-1} = -\lambda_m\mathbf{P}^{(m-1)/2}\left(\mathbf{P}^{(m-1)/2}\right)^H\mathbf{v}_{m-1}, \qquad (16)$$

where (15) and (16) is derived in Appendix A.

Moreover, one of the λ_ms satisfying (15) can be expressed as

$$\lambda_m = \sqrt{\frac{1}{\beta_m - \mathbf{v}_{m-1}^H\mathbf{P}^{(m-1)/2}\left(\mathbf{P}^{(m-1)/2}\right)^H\mathbf{v}_{m-1}}}. \qquad (17)$$

By this way, we can use (17), (16) and (10) to compute $\mathbf{P}^{(m)/2}$ from $\mathbf{P}^{(m-1)/2}$ recursively till we get $\mathbf{P}^{(M)/2}$. This recursion starts from $\mathbf{P}^{(1)/2}$ computed from (5) and (11), i.e.

$$\mathbf{P}^{(1)/2} \cdot \left(\mathbf{P}^{(1)/2}\right)^{H} = \left(\mathbf{R}^{(1)}\right)^{-1}. \tag{18}$$

One of the $\mathbf{P}^{(1)/2}$s satisfying (18) is

$$\mathbf{P}^{(1)/2} = \sqrt{\left(\mathbf{R}^{(1)}\right)^{-1}} = \sqrt{\left(\mathbf{h}_1^H \mathbf{h}_1 + \alpha\right)^{-1}}. \tag{19}$$

Since we can exchange columns and rows in $\mathbf{P}^{(M)/2}$ and $\mathbf{P}^{(M)}$ to get the new lower triangular $\mathbf{P}^{(M)/2}$ and $\mathbf{P}^{(M)}$ still satisfying (5), the upper triangular $\mathbf{P}^{(M)/2}$ is equivalent to a Cholesky factor [8] of $\mathbf{P}^{(M)}$.

To sum up, we get the new step 1* instead of the step 1 in the previous square-root algorithm, as follows:

1*) Compute the initial upper triangular $\mathbf{P}^{1/2} = \mathbf{P}^{(M)/2}$.

1*-i) Assume that the M transmit antennas are detected in the successive order of $t_M, t_{M-1}, \cdots, t_1$. Correspondently, subscript or superscript (t_m) is added to matrices $\mathbf{P}^{(m)/2}$, $\mathbf{R}^{(m)}$, vectors $\mathbf{u}_{m-1}$, $\mathbf{v}_{m-1}$, and scalars λ_m, to express that they are corresponding to the m transmit antennas $(t_m, \cdots, t_1)$. While the scalar β_m is expressed as β_{t_m} since it is corresponding to only one transmit antenna t_m.

1*-ii) Compute $\mathbf{R}^{(M)}$ defined in (11). It can be seen from (12) that all $\mathbf{R}^{(m-1)}_{(t_{m-1})}$, $\mathbf{v}^{(t_m)}_{m-1}$, $\beta_{t_m} (m = 2, 3, , M)$ are just matrices, vectors, and scalars in $\mathbf{R}^{(M)}$, respectively, and then can be derived directly [3].

1*-iii) Compute $\mathbf{P}^{(1)/2}_{(t_1)}$ by (19). Then use (17), (16) and (10) to compute $\mathbf{P}^{(m)/2}_{(t_m)}$ from $\mathbf{P}^{(m-1)/2}_{(t_{m-1})}$ iteratively, till $\mathbf{P}^{(M)/2}_{(t_M)}$, the initial $\mathbf{P}^{1/2}$. Notice that in (17) $\lambda^{(t_m)}_m$ is computed from β_{t_m} and $\mathbf{v}^{(t_m)}_{m-1} = \left[\begin{array}{cccc} \mathbf{h}_{t_1}^H \cdot \mathbf{h}_{t_m} & \mathbf{h}_{t_2}^H \cdot \mathbf{h}_{t_m} & \cdots & \mathbf{h}_{t_{m-1}}^H \cdot \mathbf{h}_{t_m} \end{array}\right]^T$.

V. The Novel BLAST Detection Algorithm

Since $\mathbf{R}^{(M)}$ is computed in step 1*-ii, there is an analogy between the steps in our current BLAST detection and the steps in [5]. Then the method in [5] to avoid computing $\left(\mathbf{H}^{(m)}\right)^H \cdot \mathbf{r}^{(m)}$ in (7) can be applied in our BLAST detection, which is briefly introduced in the following.

Let

$$\mathbf{z}^{(m)} = \left(\mathbf{H}^{(m)}\right)^H \cdot \mathbf{r}^{(m)}. \tag{20}$$

Then (7) becomes

$$\hat{a}_m = \lambda_m \cdot \left[\begin{array}{cc} (\mathbf{u}_{m-1})^H & (\lambda_m)^* \end{array}\right] \cdot \mathbf{z}^{(m)}. \tag{21}$$

As can be seen from (21), $\hat{a}_m$ can be computed via $\mathbf{z}^{(m)}$ instead of $\mathbf{r}^{(m)}$ in (7). From (8), we can update $\mathbf{z}^{(m)}$ by

$$\mathbf{z}^{(m-1)} = \bar{\mathbf{z}}^{(m)} - a_m \cdot \bar{\mathbf{f}}_m, \tag{22}$$

where $\bar{\mathbf{z}}^{(m)}$ is the first $(m-1)$ rows of the m-length column vector $\mathbf{z}^{(m)}$ after permutation, and $\bar{\mathbf{f}}_m$ is the first $(m-1)$ rows of the mth column in the matrix $\mathbf{F}^{(m)} = \left(\mathbf{H}^{(m)}\right)^H \mathbf{H}^{(m)}$ (after permutation of rows and columns) [5]. $\mathbf{F}^{(M)}$ is ready when computing $\mathbf{R}^{(M)}$ in step 1*-ii, and $\mathbf{F}^{(m-1)}$ is the first $(m-1)$ rows and columns of the matrix $\mathbf{F}^{(m)}$ after permutation of rows and columns [5].

$$\begin{bmatrix} \times\times\times\times\times \\ 0\times\times\times\times \\ 00\times\times\times \\ 000\times\times \\ 0000\times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times\times\times'\times'\times \\ 0\times\times'\times'\times \\ 000\,\times'\times \\ 00\times'\times'\times \\ 000\;0\;\times \end{bmatrix} \xrightarrow{(4,5)} \begin{bmatrix} \times\times\times'\times''\times'' \\ 0\times\times'\times''\times'' \\ 000\;0\;\times'' \\ 00\times'\times''\times'' \\ 000\,\times''\times'' \end{bmatrix} \xrightarrow{\mathbf{P}^{\frac{(4)}{2}}} \begin{bmatrix} \times\times\times'\times'' \\ 0\times\times'\times'' \\ 00\times'\times'' \\ 000\,\times'' \end{bmatrix}$$

Fig. 1. The effect of the sequence of Givens rotations

So we add the following step 1*-iv after the step 1*-iii,

1*-iv) Compute $\mathbf{z}^{(M)} = \left(\mathbf{H}^{(M)}\right)^H \cdot \mathbf{r}^{(M)}$, and get $\mathbf{F}^{(M)}$ when computing $\mathbf{R}^{(M)}$ in step 1*-ii.

Finally the steps of our novel BLAST detection algorithm are as follows:

1*) Set $m = M$, and compute $\mathbf{R}^{(M)}$, $\mathbf{F}^{(M)}$, the initial upper triangular $\mathbf{P}^{1/2} = \mathbf{P}^{(M)/2}$, and $\mathbf{z}^{(M)}$, which includes the above-described step 1*-i, 1*-ii, 1*-iii and 1*-iv.

2*) Find the minimum length row in $\mathbf{P}^{(m)/2}$ and permute it to be the last m^{th} row. Permute $\mathbf{a}^{(m)}$, $\mathbf{z}^{(m)}$ and rows and columns in $\mathbf{F}^{(m)}$ accordingly [5].

3*) Find a unitary transformation $\mathbf{\Sigma}$ such that $\mathbf{P}^{(m)/2}\mathbf{\Sigma}$ is block-upper triangular, as shown in (6).

4*) Form the least-mean-square estimate $\hat{a}_m$ by (21).

5*) Obtain a_m from $\hat{a}_m$ via slicing.

6*) Cancel the effect of a_m to $\mathbf{z}^{(m)}$ by (22) and get the reduced-order problem (9).

7*) If $m > 1$, go back to step 2* with $\mathbf{P}^{(m-1)/2}$, $\mathbf{F}^{(m-1)}$, $\mathbf{z}^{(m-1)}$ and $\mathbf{a}^{(m-1)}$ instead of $\mathbf{P}^{(m)/2}$, $\mathbf{F}^{(m)}$, $\mathbf{z}^{(m)}$ and $\mathbf{a}^{(m)}$, respectively. Otherwise stop.

Moreover, since in step 1* we get the upper triangular $\mathbf{P}^{(M)/2}$, the step 3* in our algorithm requires less computational load than the corresponding step 3 in the previous square-root algorithm, which is analyzed as follows.

When the minimum length row of $\mathbf{P}^{(M)/2}$ found in our step 2* is the i^{th} row, it must be $\left[\begin{array}{cccccc} 0 & \cdots & 0 & \varphi_i & \cdots & \varphi_M \end{array}\right]$ with the first $i-1$ elements to be zero. Then in our step 3* the transformation $\mathbf{\Sigma}$ can be performed by only $(M-i)$ Givens rotations or a $(M-i+1)\times(M-i+1)$ Householder reflection, and the former can be expressed as

$$\Sigma_g = \mathbf{\Omega}_{(i,i+1)} \mathbf{\Omega}_{(i+1,i+2)} \cdots \mathbf{\Omega}_{(M-2,M-1)} \mathbf{\Omega}_{(M-1,M)}, \tag{23}$$

where $\mathbf{\Omega}_{(k,n)}$ denotes a Givens rotation matrix which rotates the k^{th} element and the n^{th} element in each row of $\mathbf{P}^{(M)/2}$ and zeroes the k^{th} element in the minimum length row. It is easy to verify that the $\mathbf{P}^{(M-1)/2}$ got from $\mathbf{P}^{(M)/2}\mathbf{\Sigma}_g$ is always triangular. Let $M = 5$ and $i = 3$. The effect of the sequence of Givens rotations is shown in Fig. 1, where $\times$ denotes non-zero elements in $\mathbf{P}^{(M)/2}$.

In Fig. 1, it can be seen that the $\mathbf{P}^{(4)/2}$ got by deleting the 3^{rd} row and the last column in $\mathbf{P}^{(5)/2}\mathbf{\Sigma}_g$, is still triangular. So all $\mathbf{P}^{(m)/2}(m = M, M-1, \cdots, 2)$ in our step 3* are triangular.

But it is no longer satisfied and then $\mathbf{\Sigma}$ averagely needs more computational load when $\mathbf{\Sigma}$ is a Householder reflection, which has been verified by analysis and simulations.

VI. Complexity Evaluation

Table I compares the computational complexity of our BLAST detection algorithm and that of the previous square-root algorithm in [4]. In the table, we use one item to express the number of multiplications and additions, or we use (x, y) to express x multiplications and y additions when the number of multiplications and the number of additions required by a step are different.

In our step 1*-ii, $\mathbf{R}$ is Hermitian and then some computational load can be saved [3]. Our step 1*-iii uses $\mathbf{R}$ to compute an equivalent Cholesky factor of $\mathbf{P} = \mathbf{R}^{-1}$, i.e. $\mathbf{P}^{1/2}$, and the dominant computations come from (17) and (16). We can compute $\xi_{m-1} = \left(\mathbf{P}^{(m-1)/2}\right)^H \mathbf{v}_{m-1}$ firstly and then substitute ξ_{m-1} into (17) and (16), so (17) becomes $\lambda_m = \sqrt{1/\left(\beta_m - (\xi_{m-1})^H \xi_{m-1}\right)}$ and (16) becomes $\mu_{m-1} = -\lambda_m \mathbf{P}^{(m-1)/2} \xi_{m-1}$, where $\mathbf{P}^{(m-1)/2}$ is always triangular and thus much computational complexity can be saved. While it requires nearly the same complexity to compute the inverse matrix of the Cholesky factor of $\mathbf{R}$ to get $\mathbf{P}^{1/2}$ [8], but the numerical stability is expected to be worse since the matrix inversion step.

In our step 3*, $\mathbf{\Omega}_{(i+q-1,i+q)}(q = 1, \cdots, M-i)$ in (23) only rotates elements in the first $(i+q)$ rows of $\mathbf{P}^{(M)/2}$, since the $(i+q-1)^{th}$ and $(i+q)^{th}$ elements in the other rows are all zeroes. $\mathbf{\Omega}_{(i+q-1,i+q)}$ requires 6 flops, i.e. 4 multiplications and 2 additions [8], to rotate each row, and totally $6(i+q)$ flops to rotate $(i+q)$ rows. So the sequence of Givens rotations in (23) requires $\sum_{q=1}^{M-i} 6(i+q) = 3(M^2 - i^2)$ flops, and our step 3* averagely requires $\sum_{m=1}^{M}\left(\frac{1}{m}\sum_{i=1}^{m} 3(m^2 - i^2)\right) = \frac{2M^3}{3}$ flops, i.e. $\frac{4}{9}M^3$ multiplications and $\frac{2}{9}M^3$ additions, since it is reasonable to assume that the probabilities for $i = 1, 2, \cdots, m$ are equal. Moreover, the probabilities for $i = 1, 2, \cdots, m$ can be unequal and the average i can be more than $\frac{m+1}{2}$, if we set the successive detection order assumed in step 1*-i to be the optimal detection order in the last frame in the quasi-stationary channel, or the optimal detection order in the adjacent subcarrier in MIMO OFDM systems, which is quite similar to the current optimal detection order [9], [10]. Thus the computational load of our step 3* can be further reduced since averagely less Givens rotations are required, while the "best case" is that the detection order assumed in step 1*-i is just the current optimal detection order, and then the computational load of our step 3* is always zero since the probability for $i = m$ is 100%.

So the complexity of our step 3* ranges from the average $\frac{4}{9}M^3$ multiplications and $\frac{2}{9}M^3$ additions to zero in the "best case". While the step 3 in the previous square-root algorithm always requires $m-1$ Givens rotations, of which each requires $6m$ flops, and then it totally requires $2M^3$ flops, i.e. $\frac{4}{3}M^3$ multiplications and $\frac{2}{3}M^3$ additions, or the step 3 requires a Householder reflection and totally requires $\frac{2}{3}M^3$ multiplications and additions [4].

TABLE I
Complexity Comparison between the BLAST Algorithm in this Paper and the Previous Square-Root Algorithm in [4]

	This Paper	**[4]**
step 1*-ii	$\frac{1}{2}M^2N$	$3M^2N$
step 1*-iii	$\frac{1}{3}M^3$	
step 3*	$(\frac{4}{9}M^3, \frac{2}{9}M^3)$ to zero	$(\frac{4}{3}M^3, \frac{2}{3}M^3)$ or $\frac{2}{3}M^3$
step 4*	$0(M^3)$	$\frac{1}{2}M^2N$
Total	$(\frac{1}{2}M^2N + \frac{7}{9}M^3,$ $\frac{1}{2}M^2N + \frac{5}{9}M^3)$ to $\frac{1}{2}M^2N + \frac{1}{3}M^3$	$(\frac{7}{2}M^2N + \frac{4}{3}M^3,$ $\frac{7}{2}M^2N + \frac{2}{3}M^3)$ or $\frac{7}{2}M^2N + \frac{2}{3}M^3$

With the transformation $\mathbf{\Sigma}$ in [4] assumed as the more hardware-friendly Givens rotations, when $M = N$, the speedups of our algorithm over the algorithm in [4] in the number of multiplications and additions are $(\frac{7}{2}+\frac{4}{3})/(\frac{1}{2}+\frac{7}{9}) = 3.78$ and $(\frac{7}{2}+\frac{2}{3})/(\frac{1}{2}+\frac{5}{9}) = 3.95$ respectively, and become $(\frac{7}{2}+\frac{4}{3})/(\frac{1}{2}+\frac{1}{3}) = 5.8$ and $(\frac{7}{2}+\frac{2}{3})/(\frac{1}{2}+\frac{1}{3}) = 5$ respectively in the "best case". To sum up, the actual speedups in the number of multiplications and additions are 3.78-5.8 and 3.95-5 respectively.

It has been shown in [6] that the "fastest known algorithm" with the improvement totally requires $\frac{1}{2}M^2N + \frac{2}{3}M^3$ multiplications and additions. However, to get the initial estimation error covariance matrix $\mathbf{Q}$ by the equation (16) of [6], the dominant computations come from the products of $\mathbf{Tv}$ in (16) of [6], $\mathbf{g} = \mathbf{Qv}$ and $\mathbf{gg}^H = \mathbf{Qvv}^H\mathbf{Q}$ in the equation behind (16) of [6], and totally requires $\frac{5}{6}M^3$ multiplications and additions, which is more than the "$\frac{1}{2}M^3$" claimed by the equation (19) of [6]. So actually the "fastest known algorithm" with the improvement totally requires $\frac{1}{2}M^2N + M^3$ multiplications and additions, and then the speedups of our algorithm over the algorithm in [6] in the number of multiplications and additions are $(\frac{1}{2}+1)/(\frac{1}{2}+\frac{7}{9}) = 1.17$ and $(\frac{1}{2}+1)/(\frac{1}{2}+\frac{5}{9}) = 1.42$ respectively, and become $(\frac{1}{2}+1)/(\frac{1}{2}+\frac{1}{3}) = 1.8$ in the "best case".

On the other hand, we compare the computational complexity of our BLAST detection algorithm with that of the linear MMSE detection algorithm shown in (2). The linear MMSE detector needs to compute $\mathbf{R}^{-1} = \mathbf{P}$. In [3] three different algorithms to compute $\mathbf{P}$ are introduced, which all require higher computational complexity than our BLAST detection algorithm. An algorithm to compute $\mathbf{P}$ with lower complexity is got by using our fast algorithm, i.e. our step 1*, to compute an initial $\mathbf{P}^{1/2}$, and then use (5) to compute $\mathbf{P}$ from $\mathbf{P}^{1/2}$ which requires only $\frac{1}{3}M^3$ multiplications and additions. So totally it requires $\frac{1}{2}M^2N + \frac{1}{3}M^3 + \frac{1}{3}M^3 = \frac{1}{2}M^2N + \frac{2}{3}M^3$ multiplications and additions to compute $\mathbf{P}$, which are the same to the complexity of the linear MMSE detection since the computational load to get $\mathbf{z} = \mathbf{H}^H \cdot \mathbf{r}$ and $\hat{\mathbf{a}} = \mathbf{P} \cdot \mathbf{z}$ is $0(M^3)$ and can be neglected. Then if $M = N$, the ratios between the complexity of our BLAST detection algorithm

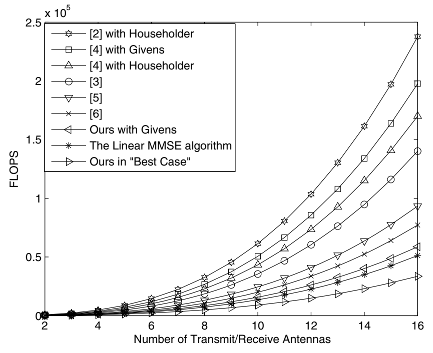

Fig. 2. Comparison of computational complexity among our algorithm, the algorithms in [2]–[6], and the linear MMSE detection algorithm. In the Fig., "Ours with Givens" means our algorithm with the unitary transformation in the step 3* to be a sequence of Givens rotations.

and that of the linear MMSE detection algorithm in the number of multiplications and additions are $(\frac{1}{2}+\frac{7}{9})/(\frac{1}{2}+\frac{2}{3}) = 1.10$ and $(\frac{1}{2}+\frac{5}{9})/(\frac{1}{2}+\frac{2}{3}) = 0.90$ respectively. The ratio becomes $(\frac{1}{2}+\frac{1}{3})/(\frac{1}{2}+\frac{2}{3}) = 0.71$ in "the best case" when the successive detection order assumed in step 1*-i is just the current optimal detection order.

Assume $M = N$, the numerical experiments to count the average flops of our algorithm and compare them with those of the algorithms in [2]–[6] and the linear MMSE detection algorithm for different number of transmit/receive antennas are shown in Fig. 2. It can be seen that they are consistent with the theoretical flops calculation.

VII. Conclusion

In this paper, we improve the square-root algorithm for BLAST in [4]. Our improved algorithm speeds up the previous algorithm [4] in the number of multiplications and additions by 3.78-5.8 and 3.95-5 respectively, and speeds up the recent published "fastest known algorithm" with the improvement [6] in the number of multiplications and additions by 1.17-1.8 and 1.42-1.8 respectively. While the ratios between the complexity of our BLAST detection algorithm and that of the linear MMSE detection algorithm in the number of multiplications and additions are 1.10-0.71 and 0.90-0.71 respectively. Meanwhile our algorithm maintains the advantages of the square-root based algorithms [2], [4] that are numerically stable and hardware-friendly, since it uses unitary transformations to avoid the matrix inversions, and gets the initial $\mathbf{P}^{1/2}$ equivalent to a Cholesky factor of $\mathbf{P}$ [8].

Appendix A
The Derivation of Equations (15) and (16)

From (5) we get $\left(\mathbf{P}^{1/2}\left(\mathbf{P}^{1/2}\right)^{H}\right)^{-1} = \mathbf{P}^{-1} = \mathbf{R}$, i.e.,

$$\left(\mathbf{P}^{(m)/2}\right)^{-H}\left(\mathbf{P}^{(m)/2}\right)^{-1} = \mathbf{R}^{(m)}. \quad (24)$$

From (10) we get

$$(\mathbf{P}^{\frac{(m)}{2}})^{-1} = \left[\begin{array}{cc} (\mathbf{P}^{\frac{(m-1)}{2}})^{-1} & -\frac{(\mathbf{P}^{\frac{(m-1)}{2}})^{-1}\mathbf{u}_{m-1}}{\lambda_m} \\ \mathbf{0}_{m-1}^T & \frac{1}{\lambda_m} \end{array}\right]. \quad (25)$$

Substituting (12) and (25) into (24), we have

$$\begin{aligned} &\left(\begin{array}{cc} \times & -\frac{(\mathbf{P}^{\frac{(m-1)}{2}})^{-H}(\mathbf{P}^{\frac{(m-1)}{2}})^{-1}\mathbf{u}_{m-1}}{\lambda_m} \\ \times & \frac{\mathbf{u}_{m-1}^H(\mathbf{P}^{\frac{(m-1)}{2}})^{-H}(\mathbf{P}^{\frac{(m-1)}{2}})^{-1}\mathbf{u}_{m-1}+1}{\lambda_m\lambda_m^*} \end{array}\right), \\ &= \left(\begin{array}{cc} \mathbf{R}^{(m-1)} & \mathbf{v}_{m-1} \\ (\mathbf{v}_{m-1})^H & \beta_m \end{array}\right) \end{aligned} \quad (26)$$

where "$\times$" denotes irrelevant entries.

From (26) we get

$$-\frac{(\mathbf{P}^{(m-1)/2})^{-H}(\mathbf{P}^{(m-1)/2})^{-1}\mathbf{u}_{m-1}}{\lambda_m} = \mathbf{v}_{m-1}, \quad (27)$$

$$\frac{\mathbf{u}_{m-1}^H\left(\mathbf{P}^{(m-1)/2}\right)^{-H}\left(\mathbf{P}^{(m-1)/2}\right)^{-1}\mathbf{u}_{m-1}+1}{\lambda_m\lambda_m^*} = \beta_m \quad (28)$$

Then from (27) we can derive (16). Substituting (16) into (28), we have

$$\mathbf{v}_{m-1}^H\mathbf{P}^{(m-1)/2}\left(\mathbf{P}^{(m-1)/2}\right)^H\mathbf{v}_{m-1} + \frac{1}{\lambda_m\lambda_m^*} = \beta_m. \quad (29)$$

From (29) we can derive λ_m satisfying (15).

Acknowledgment

The authors would like to thank the anonymous reviewers.

References

[1] P. W. Wolniansky, G. J. Foschini, G. D. Golden and R. A. Valenzuela, "V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel", *Proc. URSI International Symposium on Signals, Systems, and Electronics (ISSSE98)*, pp. 295 C300, 1998.

[2] B. Hassibi, "An efficient square-root algorithm for BLAST", *Proc. of IEEE International Conf. Acoustics, Speech, and Signal Processing, (ICASSP '00)*, pp. 737-740, June 2000.

[3] J. Benesty, Y. Huang and J. Chen, "A fast recursive algorithm for optimum sequential signal detection in a BLAST system", *IEEE Trans. on Signal Processing*, pp. 1722-1730, July 2003.

[4] H. Zhu, Z. Lei, F. P. S. Chin, "An improved square-root algorithm for BLAST", *IEEE Signal Processing Letters*, Vol. 11, No. 9, pp. 772-775, Sept. 2004.

[5] H. Zhu, Z. Lei, F.P.S. Chin, "An improved recursive algorithm for BLAST", *Signal Process.*, vol. 87, no. 6, pp. 1408-1411, Jun. 2007.

[6] Y. Shang and X. G. Xia, "An Improved Fast Recursive Algorithm for V-BLAST With Optimal Ordered Detections", *IEEE International Conference on Communications 2008 (ICC 2008)*, Session SP11.

[7] L. Szczecinski and D. Massicotte, "Low complexity adaptation of MIMO MMSE receivers, Implementation aspects", *Proc. Global Commun. Conf. (Globecom05)*, St. Louis, MO, USA, Nov. 28 - Dec. 2, 2005.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[9] W. K. Wai, C. Y. Tsui, R.S. Cheng, "A low complexity architecture of the V-BLAST system", *Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE*, 23-28 Sept. 2000 pp. 310 - 314 vol.1.

[10] W. Yan, S. Sun, Z. Lei, "A low complexity VBLAST OFDM detection algorithm", *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04)*, Vol. 4, pp. 17-21, May 2004.