An Efficient Square-Root Algorithm for BLAST

Babak Hassibi
Mathematics of Communications Research
Bell Labs, Lucent Technologies, Murray Hill, NJ 07974

January 27, 2000

Abstract

Bell Labs Layered Space-Time (BLAST) is a scheme for transmitting information over a rich-scattering wireless environment using multiple receive and transmit antennas. The main computational bottleneck in the BLAST algorithm is a "nulling and cancelation" step, where the optimal ordering for the sequential estimation and detection of the received signals is determined. To reduce the computational cost of BLAST, in this paper we develop an efficient square-root algorithm for the nulling and cancellation step. The main features of the algorithm include *efficiency*: the computational cost is reduced by $0.7M$, where M is the number of transmit antennas, and *numerical stability*: the algorithm is division-free and uses only orthogonal transformations. In a 14 antenna system designed for transmission of 1 Mbit/sec over a 30 kHz channel, the nulling and cancellation computation is reduced from 190 MFlops/sec to 19 MFlops/sec, with the overall computations being reduced from 220 MFlops/sec to 49 MFlops/sec. The numerical stability of the algorithm also make it attractive for implementation in fixed-point, rather than floating-point, architectures.

1 Introduction

Multiple transmit and receive antennas can, in theory, greatly increase the capacity, as well as significantly lower the probability of error, of a wireless communications link. One practical scheme for transmittng information over a flat-fading, rich-scattering, wireless environment is Bell Labs Layered Space-Time (BLAST), developed by Foschini, et al, 1997 [1]. BLAST has the potential to increase the capacity of the wireless link by a factor of M (where M is the minimum of the number of receive and transmit antennas), and is applicable for fixed wireless access (as in a wireless LAN).

Since multi-antenna communications allows the transmission of information at very high rates, a major issue of concern is to keep the computational complexity of the decoding algorithm at the receiver within reasonable bounds. Thus, maximum-likelihood decoding, for example, is clearly beyond question since it typically requires a search over the prohibitively large set of all possible transmitted signals. BLAST alleviates this problem by employing a "divide and conqure" decoding strategy: at each time instant, rather than jointly decoding the signals from all the transmit antennas, BLAST first decodes the "strongest" signal, then cancels the effect of this strongest transmit signal from each of the received signals, and then proceeds to decode the "strongest" of the remaining transmit signals, and so on. Such a detection scheme is often referred to as sequential *nulling and cancellation*. The optimal detection order in such a scheme is from the strongest to the weakest signal, where what we mean by strongest and weakest will be defined shortly, since this minimizes the propagation of errors from one step of detection to the next.

It turns out that the main computational bottelneck in the BLAST algorithm is the step where the optimal ordering for the sequential estimation and detection of the transmitted signals, as well as the corresponding so-called *nulling vectors*, is determined. Current implementations of BLAST (involving on the order of ten transmit and ten receive antennas) devote over 80% of the total computational cost (including channel identification and payload processing) to this step. This high computational cost limits the scope of applications that admit inexpensive real-time solutions. Moreover, when the number of transmit and receive antennas is large, the repeated pseudoinverse computations that BLAST requires can lead to numerical instability, and so a numerically robust and stable algorithm is desired.

In an attempt to reduce the computational cost of BLAST, in this paper we develop an efficient square-root algorithm for the nulling-vector-optimal-ordering step. The main features of the algorithm include *efficiency*: the computational cost is reduced by an order of magnitude, effectively from $O(M^3)$ to $O(M^4)$, where M is the number of transmit antennas, and *numerical stability*: the algorithm is division-free and uses only orthogonal transformations. The numerical stability of the algorithm also make it attractive for implementation in fixed-point, rather than floating-point, architectures.

The remainder of the paper is organized as follows. Sec. 2 begins by describing the basic model for BLAST and outlines the nulling and cancellation idea. The optimal detection order and the nulling vectors are obtained from the repeated computation of the psedoinverses of certain deflated subchannel matrices. Sec. 3 develops the main result of this paper which is a square-root algorithm for BLAST. It is first shown how it is possible to obtain the optimal ordering and all the nulling vectors by implicitly computing a *single* pseudoinverse. Thus, the computation of the subsequent pseudoinverses is unnecessary and the required computational complexity is reduced by an order of magnitude. Moreover, it is shown how all the quantities of interest can be computed using a division-free algorithm that employs only orthogonal transformations — thereby leading to a numerically stable algorithm. The paper is concluded with Sec. 4.

2 The Basic Idea of BLAST

Consider the setting of Fig. 1 where we have M signals impinging on an array of N ($N \geq M$) receivers via a *rich scattering* flat-fading environment. In BLAST the transmitted signals come from an array of transmit antennas. However, all the results and algorithms we describe are equally valid if we have M separate transmit antennas (as is the case in the uplink of a wireless LAN).

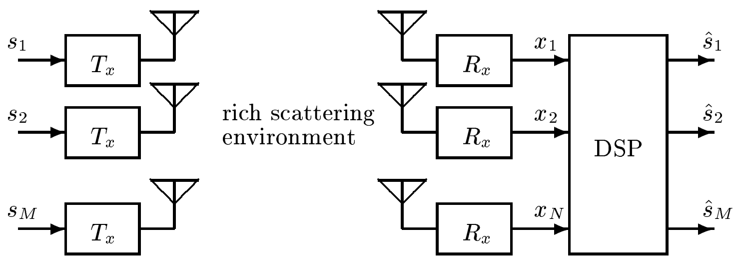

Figure 1: The basic model for BLAST.

We shall find it convenient to collect all the transmit and received signals, at each time instant, into column vectors,

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \quad \text{and} \quad s = \begin{bmatrix} s_1 \\ \vdots \\ s_M \end{bmatrix}.$$

The assumption of a flat-fading environment essentially means that the signals are narrow-band, so that we may assume a channel that is not frequency-selective. In this case, the relationship between the transmit and received signals at each time instant may be written as:

$$x = Hs + v, \tag{1}$$

where $H \in \mathcal{C}^{N \times M}$ is the channel matrix and $v \in \mathcal{C}^N$ represents spatially and temporally white additive noise. The assumption that the channel is rich-scattering essentially means that the elements of the channel matrix (when viewed as random variables) are independent of one another.

Finally, we shall also find it convenient to partition the channel matrix into its rows and columns as follows:

$$H = \begin{bmatrix} H_1 \\ \vdots \\ H_N \end{bmatrix} = \begin{bmatrix} \underline{h}_1 & \ldots & \underline{h}_M \end{bmatrix}.$$

2.1 The Procedure

In BLAST information is transmitted in bursts of length $L_T + L_P$, where L_T denotes the length of the training sequence (used to identify the channel matrix H), and L_P denotes the length of the payload (the actual data we are trying to transmit). The method for detecting the payload signals consists of three main steps:

1. **Estimate the channel matrix:** This is done using the training sequence, where L_T is of length roughly $2M$. The training sequence is typically taken to be columns of an FFT matrix.

2. **Determine the optimal detecting order and the MMSE nulling vectors**.

3. **Process the payload:** This itself is a three-step procedure referred to as nulling and cancellation:

   (a) MMSE nulling: In this step, the least-mean-squares estimate of the "strongest" transmit signal is obtained via nulling out all the "weaker" transmit signals.

   (b) slicing (decoding): In this step, the actual value of the strongest signal is detected by slicing to the nearest value in the signal constellation.

   (c) symbol cancellation: In this step, the effect of the strongest transmit signal is removed from the vector of received signals. after which we return to step (3a).

In what follows we will focus on step 2 of the above procedure.

2.2 The Optimal Ordering and MMSE Nulling Vectors

We shall assume that:

$$Ess^* = \frac{1}{\alpha} I_M, \quad Evv^* = I_N, \quad Esv^* = 0$$

where $\alpha > 0$. In other words, the signal s and additive noise v are spatially white, uncorrelated random vectors with variances $\frac{1}{\alpha} I_M$ and I_N, respectively. (The value $\frac{1}{\alpha}$ may be thought of as the SNR.) The linear least-mean-squares estimate of s, given the observations $x = Hs + v$, is

$$\hat{s} = (\alpha I + H^* H)^{-1} H^* x = (\alpha I + H^* H)^{-1} \begin{bmatrix} H^* & \sqrt{\alpha} I_M \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix},$$

or, if $\dagger$ represents the *pseudo-inverse*,

$$\hat{s} = \begin{bmatrix} H \\ \sqrt{\alpha} I_M \end{bmatrix}^{\dagger} \begin{bmatrix} x \\ 0 \end{bmatrix}. \tag{2}$$

Denoting the first N columns of the pseudo-invserse by $H_\alpha^\dagger$, we have

$$\hat{s} = H_\alpha^\dagger x.$$

Finally, if we denote the i-th row of $H_\alpha^\dagger$ by $H_{\alpha,i}^\dagger$, then we have

$$\hat{s}_i = H_{\alpha,i}^\dagger x. \tag{3}$$

$H_{\alpha,i}$ is often referred to as the i-th *MMSE nulling vector*.

2.2.1 The Estimation Error

The covariance matrix for the estimation error $s - \hat{s}$ is readily seen to be

$$E\,(s - \hat{s})\,(s - \hat{s})^* = (\alpha I + H^* H)^{-1} \triangleq P,$$

or, using the pseudo-inverse:

$$P = \begin{bmatrix} H \\ \sqrt{\alpha} I_M \end{bmatrix}^{\dagger} \left(\begin{bmatrix} H \\ \sqrt{\alpha} I_M \end{bmatrix}^{\dagger} \right)^*. \tag{4}$$

Consider now $\{\hat{s}_i, i = 1, \ldots, M\}$, *i.e.*, the entries of $\hat{s}$. Clearly, the "best" estimate, corresponding to the "strongest" signal, would be the one with the smallest error variance, *i.e.*, the one for which P_{ii} is the smallest.

Suppose that the order of the entries of s are arranged such that this best estimate is the M-th entry. Then we can independently slice $\hat{s}_M = H_{\alpha,M}^\dagger x$ to obtain a pretty good estimate of the signal s_M. If we denote the sliced value by s_M (*i.e.*, assume that we have correctly detected the signal), then we can use s_M to improve our estimate of the remaining $M-1$ signals. Thus treating s_M as a known quantity, we obtain the following reduced order problem:

$$x - \underline{h}_M s_M = H^{(M-1)} s^{(M-1)} + v, \tag{5}$$

where we have defined the deflated channel matrix, $H^{(M-1)}$, and the reduced signal vector, $s^{(M-1)}$, as

$$H^{(M-1)} = \begin{bmatrix} \underline{h}_1 & \dots & \underline{h}_{M-1} \end{bmatrix} \quad \text{and} \quad s^{(M-1)} = \begin{bmatrix} s_1 \\ \vdots \\ s_{M-1} \end{bmatrix}. \tag{6}$$

The solution is now given by

$$\hat{s}^{(M-1)} = \left(H_\alpha^{(M-1)}\right)^\dagger (x - \underline{h}_M s_M), \tag{7}$$

and the corresponding error covariance is $(\alpha I + H^{(M-1)*}H^{(M-1)})^{-1} \triangleq P^{(M-1)}$.

Note that there are two corrections in finding the new least-squares estimate $\hat{s}^{(M-1)}$. One has to do with the fact that there is one less unknown, so that the coefficient matrix has changed, and the second with the fact that the left-hand-side of the equation has changed from x to $x - \underline{h}_M s_M$.

2.2.2 Basic Algorithm

The basic idea can thus be summarized as follows:

1. Find $P = (\alpha I + H^*H)^{-1}$ and $H_\alpha^\dagger$.

2. Find the smallest diagonal entry of P and reorder the entries of s so that the smallest diagonal entry is the last (M-th) one.

3. Form the least-mean-squares estimate $\hat{s}_M = H_{\alpha,M}^\dagger x$.

4. Obtain s_M (via slicing) from $\hat{s}_M = H_{\alpha,M}^\dagger x$.

5. Cancel the effect of s_M and consider the *reduced-order* problem (5).

6. Continue to find $P^{(M-1)} = (\alpha I + H^{(M-1)*}H^{(M-1)})^{-1}$ and $H_\alpha^{(M-1)\dagger}$, and so on.

2.3 Computational Complexity

The computational complexity of the method just described can be given as follows.

- Channel estimation: $2MN\log_2 L_T$ (the $\log_2 L_T$ term is under the assumption that the training sequence is obtained from the columns of an FFT matrix.)

- Determining the nulling vectors and optimal ordering: The most numerically stable way to compute the pseudoinverse is using the singular value decomposition (svd) of $\begin{bmatrix} H^* & \frac{1}{\sqrt{\alpha}} \end{bmatrix}^*$. (An alternative way is to use the QR decomposition, which reduces the computation by a factor of roughly 1.3-1.4. However, it is less numerically stable since one is required to invert the upper triangular R of the QR factorization to obtain the pseudoinverse.)

  The complexity of performing the svd, $A = U\Sigma V^*$, of a $n \times m$ matrix A, as given in [2], is $2n^2m + 11m^3$. The complexity of finding the pseudoinverse $A^\dagger = V\Sigma^{-1}U^*$ is $2nm^2$. Now since in the BLAST algorithm described we need to repeatedly compute the pseudoinverse of a deflated channel matrix with dimensions $(N+i)\times i$, with $i = M, M-1, \ldots, 1$, the total computational complexity is:

$$\sum_{i=1}^{M}\left[2(N+i)^2 i + 11i^3 + 2(N+i)i^2\right] = N^2M^2 + 2NM^3 + \frac{15}{4}M^4.$$

- Processing the payload: $2MNL_P$.

When $M = N$ these complexities simplify to:

- $2M^2\log_2 L_T$,
- $\frac{27}{4}M^4$,
- $2M^2 L_P$,

respectively. Thus the complexity of the optimal ordering and nulling vector computation grows as the fourth power of the number of transmit antennas.

To gain some perspective on the relative computational complexities of the above three steps, let us consider an application that targets the transmission of 1Mb/s of data over a 30khz wireless channel. Such a target can be achieved using a transmission rate of 24.3 ksymbol/sec, 16-QAM modulation, $L_T = 32$, $L_P = 100$, and $M = N = 14$ antennas. The resulting relative computational complexities are given in the table below.

	Flops/burst	MegaFlops/s	%
channel estimation	7,840	1.44	0.65
nulling vectors and ordering	1,036,000	190.8	86.3
payload processing	156,800	28.9	13.1
TOTAL	1,200,000	221.2	100

The dominant portion, nearly %90, of the computation involves determining the nulling vectors and optimal ordering. The question that begs itself, therefore, is whether this computation be reduced in a numerically stable way?

2.4 Objectives of an Alternative Algorithm

In coming up with an alternative algorithm for the 6-step scheme described above, the following objectives appear to be natural:

- The algorithm must be **cost efficient**: Is it possible to find $H_\alpha^{(M-1)\dagger}$ and $P^{(M-1)}$ from $H_\alpha^\dagger$ and P, without having to "re-solve" the reduced-order problem all over again?
- The algorithm must be *numerically stable and robust*. This typically means:
  - Avoid "squaring" things (forming H^*H, for example, is undesirable). This increases the dynamic range of the quantities involved, the condition numbers, etc.
  - Avoid "inverting" things (inverting H^*H to obtain P is undesirable).
  - Make as much use as possible of *unitary* transformations.

In what follows, we will propose a square-root implementation of the algorithm that meets the above (two-plus-three-fold) goals.

3 A Square-Root Algorithm

In order to begin the use of unitary transformations, and also to avoid squaring H, consider the QR decomposition of the augmented channel matrix

$$\left[\begin{array}{c} H \\ \sqrt{\alpha} I_M \end{array}\right] = QR = \left[\begin{array}{c} Q_\alpha \\ Q_2 \end{array}\right] R, \tag{8}$$

where Q is an $(N+M)\times M$ matrix with orthonormal columns, and R is $M\times M$ and nonsingular. We need not insist that R be upper triangular at this point. Note, further, that we have partitioned Q in correspondence with the partitioning of the augmented channel matrix — thus, Q_α has dimensions $N \times M$.

Now

$$P = (\alpha I + H^* H)^{-1} = (R^* R)^{-1} = R^{-1} R^{-*}.$$

We can thus identify R^{-1} with a "square-root" of P, *i.e.*,

$$R^{-1} = P^{1/2}, \quad P^{1/2} P^{*/2} = P. \tag{9}$$

In particular, the pseudo-inverse of the augmented channel matrix is now given by,

$$\left[\begin{array}{c} H \\ \sqrt{\alpha} I_M \end{array}\right]^\dagger = R^{-1} Q^* = P^{1/2} Q^*,$$

so that

$$H_\alpha^\dagger = P^{1/2} Q_\alpha^*. \tag{10}$$

In conclusion, given $P^{1/2}$ and Q_α, we can compute both the pseudoinverse and the error covariance matrix. Thus a key question is how best to compute $P^{1/2}$ and Q_α? However, before addressing this issue, let us focus on:

1. How to find the smallest diagonal entry of P?

2. How to find the square-root factor of $P^{(M-1)}$ from $P^{1/2}$?

3. How to find the nulling vectors?

3.1 The Square-Root Factor of $P^{(M-1)}$

The answer to question 1 is given below.

Claim 1: *The minimum diagonal entry of P, corresponds to the minimum length row of $P^{1/2}$.*

Proof: Follows immediately from $P = P^{1/2} P^{*/2}$, which implies that the diagonal entries of P are simply the squared lengths of the rows of $P^{1/2}$.

■

Suppose now that we have reordered the entries of the transmitted signal s so that the M-th diagonal entry of P is the smallest. Consider any unitary transformation Σ that rotates (or reflects) the M-th row of $P^{1/2}$ to lie along the direction of the M-th unit vector. In other words,

$$P^{1/2}\Sigma = \left[\begin{array}{cc} P^{(M-1)/2} & P_M^{(M-1)/2} \\ 0 & p_M^{1/2} \end{array}\right], \tag{11}$$

where $p_M^{1/2}$ is a scalar. Note that, since Σ is unitary, $P^{1/2}\Sigma$ is also a square-root factor of P.

Claim 2: *Let $P = (\alpha I + H^*H)^{-1}$ and $P^{(M-1)} = \left(\alpha I + H^{(M-1)*}H^{(M-1)}\right)^{-1}$, where $H^{(M-1)}$ is the deflated channel matrix of (6). Then $P^{(M-1)/2}$, as given by (11), is a square-root factor of $P^{(M-1)}$. In other words, the $(1,1)$ block entry of the block upper triangular square-root factor of P is a square-root factor of $P^{(M-1)}$.*

Proof: Note that

$$
\begin{aligned}
\alpha I + HH^* &= P^{-1} = (P^{1/2}P^{*/2})^{-1} = (P^{1/2}\Sigma\Sigma^*P^{*/2})^{-1} = (\Sigma^*P^{*/2})^{-1}(P^{1/2}\Sigma)^{-1} \\
&= \left(\begin{bmatrix} P^{(M-1)*/2} & 0 \\ (P_M^{(M-1)/2})^* & p_M^{*/2} \end{bmatrix}\right)^{-1} \left(\begin{bmatrix} P^{(M-1)/2} & P_M^{(M-1)/2} \\ 0 & p_M^{1/2} \end{bmatrix}\right)^{-1} \\
&= \begin{bmatrix} \left(P^{(M-1)*/2}\right)^{-1} & 0 \\ \times & \times \end{bmatrix} \begin{bmatrix} \left(P^{(M-1)/2}\right)^{-1} & \times \\ 0 & \times \end{bmatrix} \\
&= \begin{bmatrix} \left(P^{(M-1)/2}P^{(M-1)*/2}\right)^{-1} & \times \\ \times & \times \end{bmatrix}
\end{aligned} \tag{12}
$$

where $\times$ denotes entries that are not relevant at this time. Now, on the other hand,

$$
\alpha I + H^*H = \alpha I + \begin{bmatrix} H^{(M-1)*} \\ \underline{h}_M^* \end{bmatrix} \begin{bmatrix} H^{(M-1)} & \underline{h}_M \end{bmatrix} = \begin{bmatrix} \alpha I + H^{(M-1)*}H^{(M-1)} & \times \\ \times & \times \end{bmatrix}.
$$

Comparing this last equation with (12) yields,

$$
\left(P^{(M-1)/2}P^{(M-1)*/2}\right)^{-1} = \alpha I + H^{(M-1)*}H^{(M-1)} = \left(P^{(M-1)}\right)^{-1},
$$

from which it readily follows that $P^{(M-1)/2}$ is a square-root factor of $P^{(M-1)}$.

$\blacksquare$

Thus, to find the square-root factor of $P^{(M-1)}$, we need to make P *block upper triangular*. Once this is done, we can determine the next signal to be detected by finding the minimum length row of $P^{(M-1)/2}$. Reordering the rows of $P^{(M-1)/2}$ so that this minimum length row corresponds to the last ($(M-1)$-th) row, and upper block triangularizing $P^{(M-1)/2}$ allows us to obtain the next square-root factor, $P^{(M-2)/2}$. Continuing in a similar fashion to determine the optimal ordering for the remaining signals effectively results in an *upper-triangularization*, as opposed to just a block-upper-triangularization, of the square-root matrix $P^{1/2}$.

Remarks

1. *The structure of $P^{1/2}$:* The defining property of the final upper-triangularized matrix, which with some abuse of notation we shall continue to call $P^{1/2}$, is the following: for all $i = 2, \ldots, M$:

$$
\left(P^{1/2}\right)_{ii}^2 \leq \sum_{k=j}^{i} \left|\left(P^{1/2}\right)_{jk}\right|^2, \quad j = 1, \ldots i-1 \tag{13}
$$

   where $\left(P^{1/2}\right)_{ij}$ is the (i,j)-th entry of $P^{1/2}$. In other words, each of the diagonal entries of $P^{1/2}$ is smaller than the length of the row vectors that lie above it and to its left.

2. *The diagonal entries of $P^{1/2}$:* The squared i-th diagonal entry of $P^{1/2}$, i.e., $\left(P^{1/2}\right)^2_{ii}$, has the interpretation of being the mean-square-error in estimating the i-th signal *after all the signals before it have been successfully decoded.* Recall that the mean-square-error in estimating the i-th signal before decoding any other signals is $P_{ii} = \sum_{k=i}^{M} \left|\left(P^{1/2}\right)_{ik}\right|^2$. Thus, the decrease in the mean-square-error (assuming all earlier signals have been successfully decoded) is

$$\sum_{k=i+1}^{M} \left|\left(P^{1/2}\right)_{ik}\right|^2 .$$

3.2 The Nulling Vectors

Let us now study the consequences of an upper triangular $P^{1/2}$ to the QR decomposition of the augmented channel matrix. Thus, focusing on the first $(M-i)$ columns in (8), for some $i = 1, \ldots, M-1$, and using $R = P^{-1/2}$, we may write

$$\left[\begin{array}{cc} H^{(M-i)} & \times \\ \sqrt{\alpha} I_{M-i} & \times \\ 0 & \times \end{array}\right] = Q \left[\begin{array}{cc} P^{(M-i)/2} & \times \\ 0 & \times \end{array}\right]^{-1} = Q \left[\begin{array}{cc} \left(P^{(M-i)/2}\right)^{-1} & \times \\ 0 & \times \end{array}\right],$$

from which we readily conclude

$$\left[\begin{array}{c} H^{(M-i)} \\ \sqrt{\alpha} I_{M-i} \end{array}\right] = Q \left[\begin{array}{c} \left(P^{(M-i)/2}\right)^{-1} \\ 0 \end{array}\right] = Q^{(M-i)} \left(P^{(M-i)/2}\right)^{-1}, \tag{14}$$

where $Q^{(M-i)}$ represents the first $M-i$ columns of Q. Now since the columns of Q are orthogonal the same is true of the columns of $Q^{(M-i)}$. Thus (14) is the QR decomposition for the deflated channel matrix. In particular, $\left(P^{(M-i)/2}\right)^{-1}$ is the "R" in this QR decomposition which, in light of the results of the previous section, is an expected result. Moreover, the pseudoinverse of the deflated channel matrix is

$$\left(\left[\begin{array}{c} H^{(M-i)} \\ \sqrt{\alpha} I_{M-i} \end{array}\right]\right)^{\dagger} = P^{(M-i)/2} Q^{(M-i)*} = P^{(M-i)/2} \left[\begin{array}{cc} Q_{\alpha}^{(M-i)*} & Q_{2}^{(M-i)*} \end{array}\right]. \tag{15}$$

Recall that the $(M-i)$-th nulling vector is simply the first N entries of the last ($(M-i)$-th) row of the pseudoinverse for the above deflated channel matrix. Since $P^{(M-i)/2}$ is upper triangular, the last row of the pseudoinverse is simply the last row of $Q^{(M-i)*}$ multiplied by the $(M-i, M-i)$ diagonal entry of $P^{(M-i)/2}$.

We thus have demonstrated the following result.

Claim 3: *Suppose that we have repeated the steps of the above Claims 1 and 2 until $P^{1/2}$ is transformed to an upper triangular matrix. Moreover, let $\underline{q}_{\alpha,i}$, $i = 1, \ldots, M$ denote the resulting columns of Q_α, i.e.,*

$$Q_\alpha = \left[\begin{array}{ccc} \underline{q}_{\alpha,1} & \cdots & \underline{q}_{\alpha,M} \end{array}\right].$$

Then the nulling vectors for the signals s_1 to s_M are given by:

$$H^{\dagger}_{\alpha,i} = p_i^{1/2} \underline{q}^{*}_{\alpha,i}, \tag{16}$$

where $p_i^{1/2}$ denotes the i-th diagonal entry of $P^{1/2}$.

In other words, the desired nulling vectors are simply found from scaling the columns of Q_α by the diagonals of $P^{1/2}$.

The important conclusion of Claims 1 to 3 is the following. Once $P^{1/2}$ and Q_α are computed, there is no need to recompute them for the deflated channel matrices $H^{(M-i)}$, $i = 1, \ldots, M-1$. All the information we need to determine the optimal ordering and the nulling vectors is already implicit in $P^{1/2}$ and Q_α. This leads to an order of magnitude of computational savings.

For convenience, let us summarize the algorithm obtained so far.

3.2.1 Computing the Optimal Ordering and Nulling Vectors

1. Compute $P^{1/2}$ and Q_α.

2. Find the minimum length row of $P^{1/2}$ and permute it to be the last (Mth) row. This represents the first signal to be detected. Permute s accordingly.

3. Find a unitary Σ such that $P^{1/2}\Sigma$ is block upper triangular:

$$P^{1/2}\Sigma = \begin{bmatrix} P^{(M-1)/2} & P_M^{(M-1)/2} \\ 0 & p_M^{1/2} \end{bmatrix}.$$

4. Update Q_α to $Q_\alpha \Sigma$.

5. The nulling vector for the M-th signal is given by $p_M^{1/2} \underline{q}_{\alpha,M}^*$, where $\underline{q}_{\alpha,M}$ is the$M$-th column of Q_α.

6. Go back to step 3, but now with $P^{(M-1)/2}$ and $Q_\alpha^{(M-1)}$ (the first $M-1$ columns of Q_α).

3.3 Computing $P^{1/2}$ and Q_α

The algorithm presented so far satisfies all our requirements of cost-efficiency (since we do not repeatedly compute pseudoinverses) and numerical stability (since we avoided squaring and computing inverses, and have used only unitary transformations). The only issue that remains is one of computing $P^{1/2}$ and Q_α.

An obvious method is via the QR decomposition of the augmented channel given in (8). Although this immediately gives us Q_α, it is an undesirable solution since it requires us to invert R to obtain $P^{1/2}$.[1]

In fact, we can avoid inverting R by constructing $P = (\alpha I + H^* H)^{-1}$ directly. Let's see how. Note that:

$$P = \left(\alpha I + \sum_{j=1}^{N} H_j^* H_j \right)^{-1}.$$

Therefore, if we define

$$P_{|i} \triangleq \left(\alpha I + \sum_{j=1}^{i} H_j^* H_j \right)^{-1}, \tag{17}$$

[1]Even though R is typically upper triangular, so that R^{-1} can be computed with a method akin to back-substitution, this is still an undesirable step.

using the matrix inversion lemma, we have

$$P_{|i} = P_{|i-1} - \frac{P_{|i-1}H_i^* H_i P_{|i-1}}{r_{e,i}}, \quad r_{e,i} = 1 + H_i P_{|i-1} H_i^*, \quad P_{|0} = \alpha^{-1} I. \tag{18}$$

Clearly, after N steps we obtain $P_{|N} = P$. The above recursion is readily recognized as the so-called Riccati recursion of the RLS (recursive-least-squares) algorithm, a special case of the Kalman filter (see, e.g, [3]). This suggests that there should be a way to recursively compute Q_α.

To this end, note that $\hat{s} = P^{1/2} Q_\alpha^* x = H_\alpha^\dagger x$, the least-mean-squares estimate of the signal, satisfies the RLS recursion:

$$\hat{s}_{|i} = \hat{s}_{|i-1} + \bar{K}_{p,i} r_{e,i}^{-1/2} (x_i - H_i \hat{s}_{|i-1}), \quad \bar{K}_{p,i} = P_{|i-1} H_i^* r_{e,i}^{-*/2}, \quad \hat{s}_{|0} = 0,$$

so that after N steps we obtain $\hat{s} = \hat{s}_{|N}$. The above recursion shows that if we replace the N-dimensional column vector x by the $N \times N$ identity matrix I_N, then we can obtain a recursion for the pseudoinverse $H_\alpha^\dagger = P^{1/2} Q_\alpha$ as follows:

$$H_{\alpha|i}^\dagger = H_{\alpha|i-1}^\dagger + \bar{K}_{p,i} r_{e,i}^{-1/2} \left(e_i^* - H_i H_{\alpha|i-1}^\dagger \right), \quad \bar{K}_{p,i} = P_{|i-1} H_i^* r_{e,i}^{-*/2}, \quad H_{\alpha|0}^\dagger = 0_{M \times N}, \tag{19}$$

where e_i^* is the i-th unit row vector with zeros in all entries except the i-th one which is unity. After N steps of (19) we obtain $H_\alpha^\dagger = H_{\alpha|N}^\dagger$. Note that by replacing x with I_N all we have done in (19) is replace x_i, the i-th entry of x, with e_i^*, the i-th row of I_N.

The recursion (18) constitutes an algorithm for computing P that avoids explicit inversions. However, it has the drawback of squaring quantities ($H_i^* H_i$), requiring scalar division (by $r_{e,i}$), and not using unitary transformations. Moreover, we still need to find $P^{1/2}$ from P via a Cholesky factorization, or some other means.

Fortunately, there is a way to directly propogate $P_{|i}^{1/2}$ *i.e.*, the square-root factor of $P_{|i}$ (see, e.g., [3]). For completeness, we will outline an argument for how this can be done. Consider the block matrix

$$T_i = \begin{bmatrix} r_{e,i} & H_i P_{|i-1} \\ P_{|i-1} H_i^* & P_{|i-1} \end{bmatrix}.$$

A block upper-lower triangular factorization of T_i yields:

$$T_i = \begin{bmatrix} 1 & H_i \\ 0 & I \end{bmatrix} \begin{bmatrix} \underbrace{r_{e,i} - H_i P_{|i-1} H_i^*}_{=1} & 0 \\ 0 & P_{|i-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ H_i^* & I \end{bmatrix} = \begin{bmatrix} 1 & H_i P_{|i-1}^{1/2} \\ 0 & P_{|i-1}^{1/2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ P_{|i-1}^{*/2} H_i^* & P_{|i-1}^{*/2} \end{bmatrix},$$

whereas a block lower-upper triangular factorization yields:

$$T_i = \begin{bmatrix} 1 & 0 \\ P_{|i-1} H_i^* & I \end{bmatrix} \begin{bmatrix} r_{e,i} & 0 \\ 0 & \underbrace{P_{|i-1} - \frac{P_{|i-1} H_i^* H_i P_{|i-1}}{r_{e,i}}}_{P_{|i}} \end{bmatrix} \begin{bmatrix} 1 & H_i P_{|i-1} \\ 0 & I \end{bmatrix} = \begin{bmatrix} r_{e,i}^{1/2} & 0 \\ \bar{K}_{p,i} & P_{|i}^{1/2} \end{bmatrix} \begin{bmatrix} r_{e,i}^{*/2} & \bar{K}_{p,i}^* \\ 0 & P_{|i}^{*/2} \end{bmatrix}.$$

Equating the last two equalities:

$$\begin{bmatrix} 1 & H_i P_{|i-1}^{1/2} \\ 0 & P_{|i-1}^{1/2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ P_{|i-1}^{*/2} H_i^* & P_{|i-1}^{*/2} \end{bmatrix} = \begin{bmatrix} r_{e,i}^{1/2} & 0 \\ \bar{K}_{p,i} & P_{|i}^{1/2} \end{bmatrix} \begin{bmatrix} r_{e,i}^{*/2} & \bar{K}_{p,i}^* \\ 0 & P_{|i}^{*/2} \end{bmatrix}. \tag{20}$$

Now we will need to use the following readily verified fact.

Claim 4: *Let A and B be two square and nonsingular matrices such that $AA^* = BB^*$. Then there exists a unitary transformation Θ, such that*

$$A\Theta = B.$$

Proof: Clearly, $A\underbrace{A^*B^{-*}}_{\Theta} = B$. It now follows that Θ is unitary, since:

$$\Theta^*\Theta = B^{-1}AA^*B^{-*} = B^{-1}BB^*B^{-*} = I.$$

■

Applying Claim 4 to (20) shows that there exists a unitary matrix, Θ_i, such that

$$\begin{bmatrix} 1 & H_i P_{|i-1}^{1/2} \\ 0 & P_{|i-1}^{1/2} \end{bmatrix} \Theta_i = \begin{bmatrix} r_{e,i}^{1/2} & 0 \\ \bar{K}_{p,i} & P_{|i}^{1/2} \end{bmatrix}. \tag{21}$$

The above equation shows us how to propagate the square-root factors $P_{|i}^{1/2}$. At time $i-1$ all we need to do is to form the (so-called) *pre-array*

$$\begin{bmatrix} 1 & H_i P_{|i-1}^{1/2} \\ 0 & P_{|i-1}^{1/2} \end{bmatrix},$$

and find a unitary transformation (essentially a rotation or a reflection, or a combination thereof) that makes the first row to lie along the direction of the first unit row vector. The square-root factor, $P_{|i}^{1/2}$, at the next time instant is then found from the triangularized (so-called) *post-array*:

$$\begin{bmatrix} r_{e,i}^{1/2} & 0 \\ \bar{K}_{p,i} & P_{|i}^{1/2} \end{bmatrix}.$$

The algorithm is clearly initialized with $P_{|0}^{1/2} = \frac{1}{\sqrt{\alpha}} I$.

Therefore, the only remaining issue is how to find Q_α. One method is to use the recursion (19) to find $H_\alpha^\dagger$, and then to use $Q_\alpha^* = P^{-1/2} H_\alpha^\dagger$. But this is undesirable, since it requires us to invert $P^{1/2}$. It turns out that we can extend the square-root algorithm (21) to compute Q_α directly. This appears to be a new result.

To this end, augment the pre-array in (21) by the block row vector $\begin{bmatrix} -e_i & \left(H_{\alpha|i-1}^\dagger\right)^* P_{|i-1}^{-*/2} \end{bmatrix}$, and apply the same unitary transformation to the extended pre-array. This will yield:

$$\begin{bmatrix} 1 & H_i P_{|i-1}^{1/2} \\ 0 & P_{|i-1}^{1/2} \\ -e_i & \left(H_{\alpha|i-1}^\dagger\right)^* P_{|i-1}^{-*/2} \end{bmatrix} \Theta_i = \begin{bmatrix} r_{e,i}^{1/2} & 0 \\ \bar{K}_{p,i} & P_{|i}^{1/2} \\ A_i & B_i \end{bmatrix}, \tag{22}$$

where $\begin{bmatrix} A_i & B_i \end{bmatrix}$ is the resulting transformed last block row. [Note that the first two block rows in the post-array are the same, since we are using the same Θ_i.]

To determine A_i and B_i, we can use the fact that unitary transformations do not change inner products. Thus, equating the inner products of the first and third block rows in the above pre- and post-arrays, yields

$$-e_i^* + H_i H^\dagger_{\alpha|i-1} = r_{e,i}^{1/2} A_i^*,$$

so that

$$A_i^* = -r_{e,i}^{-1/2}\left(e_i^* - H_i H^\dagger_{\alpha|i-1}\right) \tag{23}$$

Likewise, equating the innerproducts of the second and third block rows in the pre- and post-arrays, yields

$$H^\dagger_{\alpha|i-1} = \bar{K}_{p,i} A_i^* + P_{|i}^{1/2} B_i^* = -\bar{K}_{p,i} r_{e,i}^{-1/2}\left(e_i^* - H_i H^\dagger_{\alpha|i-1}\right) + P_{|i}^{1/2} B_i^*,$$

so that

$$B_i^* = P_{|i}^{-1/2}\left(H^\dagger_{\alpha|i-1} + \bar{K}_{p,i} r_{e,i}^{-1/2}(e_i^* - H_i H^\dagger_{\alpha|i-1})\right) = P_{|i}^{-1/2} H^\dagger_{\alpha|i}! \tag{24}$$

We thus have shown that

$$\begin{bmatrix} 1 & H_i P_{|i-1}^{1/2} \\ 0 & P_{|i-1}^{1/2} \\ -e_i^* & \left(H^\dagger_{\alpha|i-1}\right)^* P_{|i-1}^{-*/2} \end{bmatrix} \Theta_i = \begin{bmatrix} r_{e,i}^{1/2} & 0 \\ \bar{K}_{p,i} & P_{|i}^{1/2} \\ -\left(e_i^* - H_i H^\dagger_{\alpha|i}\right)^* r_{e,i}^{-*/2} & \left(H^\dagger_{\alpha|i}\right)^* P_{|i}^{-*/2} \end{bmatrix}. \tag{25}$$

The above arguments have demonstrated the following claim, which is our proposed method for computing $P^{1/2}$ and Q_α.

Claim 5: *$P^{1/2}$ and Q_α can be computed using the following recursion.*

$$\begin{bmatrix} 1 & H_i P_{|i-1}^{1/2} \\ 0 & P_{|i-1}^{1/2} \\ -e_i & B_{i-1} \end{bmatrix} \Theta_i = \begin{bmatrix} r_{e,i}^{1/2} & 0 \\ \bar{K}_{p,i} & P_{|i}^{1/2} \\ A_i & B_i \end{bmatrix}, \quad P_{|0}^{1/2} = \frac{1}{\sqrt{\alpha}} I,\ B_0 = 0_{N\times M} \tag{26}$$

where e_i is the i-th unit vector of dimension N (i.e., it is an $N \times 1$ vector of all zeros except for the i-th entry, which is unity), and Θ_i is any unitary transformation that transforms the first row of the pre-array to lie along the direction of the first unit row vector. After N steps the algorithm yields the desired quantities via:

$$P^{1/2} = P_{|N}^{1/2} \quad \text{and} \quad Q_\alpha = B_N. \tag{27}$$

Proof: The proof uses the arguments above. The only things left to show are the initial condition $B_0^* = P_{|0}^{-1/2} H^\dagger_{\alpha|0} = 0_{M\times N}$ (since $H^\dagger_{\alpha|0} = 0_{M\times N}$), and that $B_N^* = P_{|N}^{-1/2} H^\dagger_{\alpha|N} = P^{-1/2} H^\dagger_\alpha = Q_\alpha^*$.

■

3.4 The Algorithm

We are now in a position to summarize the full algorithm.

1. Compute $P^{1/2}$ and Q_α.

   - Propagate the following square-root algorithm:

   $$\begin{bmatrix} 1 & H_i P^{1/2}_{|i-1} \\ 0 & P^{1/2}_{|i-1} \\ -e_i & B_{i-1} \end{bmatrix} \Theta_i = \begin{bmatrix} r^{1/2}_{e,i} & 0 \\ \bar{K}_{p,i} & P^{1/2}_{|i} \\ A_i & B_i \end{bmatrix}, \quad P^{1/2}_{|0} = \frac{1}{\sqrt{\alpha}} I, \; B_0 = 0_{N\times M}$$

   where e_i is the i-th unit vector of dimension N, and Θ_i is any unitary transformation that block lower triangularizes the pre-array. After N steps, we have

   $$P^{1/2} = P^{1/2}_{|N} \quad \text{and} \quad Q_\alpha = B_N.$$

2. Find the minimum length row of $P^{1/2}$ and permute it to be the last (Mth) row. Permute s accordingly.

3. Find a unitary Σ such that $P^{1/2}\Sigma$ is block upper triangular:

   $$P^{1/2}\Sigma = \begin{bmatrix} P^{(M-1)/2} & P^{(M-1)/2}_M \\ 0 & p^{1/2}_M \end{bmatrix}.$$

4. Update Q_α to $Q_\alpha \Sigma$.

5. The nuling vector for the M-th signal is given by $p^{1/2}_M \underline{q}^*_{\alpha,M}$, where $\underline{q}_{\alpha,M}$ is the M-th column of Q_α.

6. Go back to step 3, but now with $P^{(M-1)/2}$ and $Q^{(M-1)}_\alpha$ (the first $M-1$ columns of Q_α).

Remarks

The above algorithm satisfies both our objective of cost-efficiency:

- we have avoided computing the pseudo-inverse (or QR decomposition) for each deflated subchannel matrix.

and of numerical stability and robustness:

- we have avoided squaring any of the quantities.
- we have avoided computing inverses (and even scalar divisions) altogether.
- we have used unitary transformations as much as possible.

Computational Complexity

The computational complexity of the square-root algorithm of Sec. 3.4 can be computed as follows.

- Computing $P^{1/2}$ and Q_α:
  - Forming the pre-array: $2M^2$. (This corresponds to computing $H_i P^{1/2}_{|i-1}$.)
  - Propagating the i-th step of the square-root algorithm: $4M^2 + 4Mi$. (This corresponds to right-multiplying the $(M+1)\times(M+i+1)$ pre-array with a Householder matrix Θ_i. The pre-array is taken to have dimensions $(M+1)\times(M+i+1)$, rather than $(M+1)\times(M+N+1)$, since at the i-th step the last $N-i$ rows of the pre-array are all zero.)

  Thus, retaining only the dominant terms, the total computational complexity of this stage is:

$$\sum_{i=1}^{N}\left[2M^2 + 4M^2 + 4Mi\right] = 6M^2N + 2MN^2.$$

- Finding the optimal ordering and nulling vectors:
  - The dominant computations come the from the products $P\Sigma$ and $Q_\alpha\Sigma$. When Σ is taken as a Householder matrix the complexity of these multiplications in the i-th step is $2i^2 + 2Ni$.

  Thus, retaining only the dominant terms, the total complexity here is:

$$\sum_{i=1}^{N}\left[2i^2 + 2N^i\right] = \frac{2}{3}M^3 + M^2N.$$

The total computational complexity thus amounts to

$$\frac{2}{3}M^3 + 7M^2N + 2MN^2. \tag{28}$$

In particular, when $M = N$, we obtain $\frac{29}{3}M^3$. Comparing this with $\frac{27}{4}M^4$, the complexity obtained for the original method, we see that the new method offers computational savings by a factor of roughly $0.7M$.

Returning to our target application (1 Mbit/s, 30 kHz channel, 24.3 ksymbol/sec, 16-QAM, $M = N = 14$, $L_T = 32$, $L_P = 100$), the resulting relative computational complexities are given in the table below.

	Flops/burst	MegaFlops/s	%
channel estimation	7,840	1.44	2.9
nulling and ordering	106,100	19.5	39.1
payload processing	156,800	28.9	58.0
TOTAL	270,400	49.8	100

Thus the computation for the nulling vector and optimal ordering step has been reduced from 190 MFlops/s to 19 MFlops/s, and the total computation has been reduced from 221 MFlops/s to 50 MFlops/s.

Fig.2 further compares the computational requirements of the original and new schemes as a function of $M = N$.

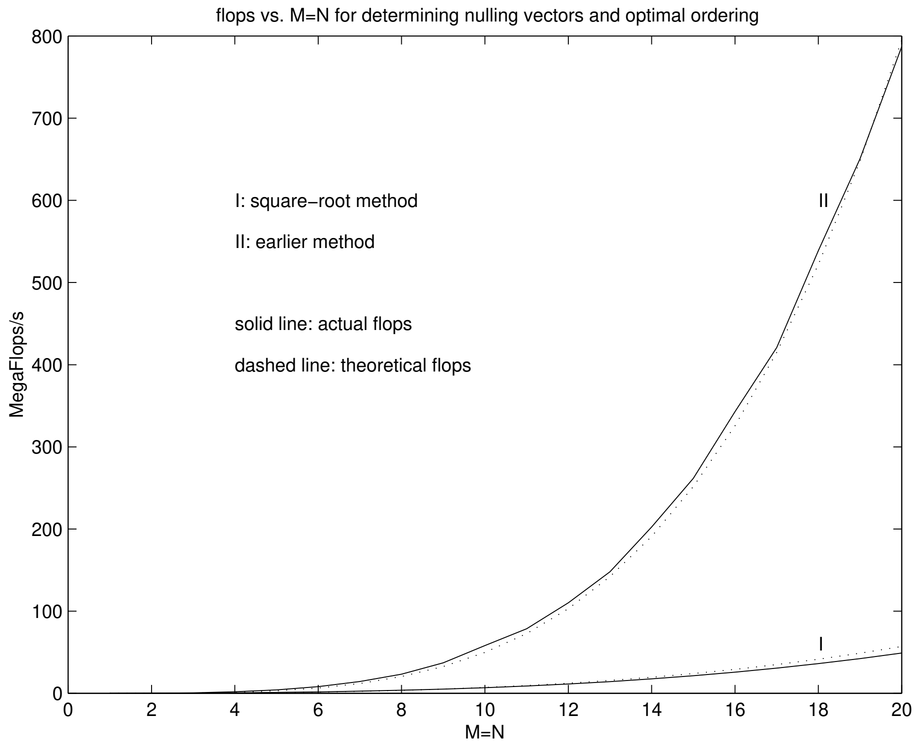

Figure 2: *Flops for determining nulling vectors and optimal ordering as a function of $M = N$. I: square-root method. II: earlier method. Solid line: actual flops, Dashed line: theoretical flops.*

4 Conclusion and Final Remarks

In this paper we developed an efficient square-root algorithm for BLAST which offers an order of magnitude of savings in the computational complexity compared to earlier methods.

The prominent component of the algorithm is the use of unitary transformations (the Θ_i and the Σ) which introduce zeros in prescribed entries of given row vectors. These can be performed either by using a single Householder reflection, or a sequence of Givens rotations (see, e.g., [2, 3]). In hardware, the sequence of Givens rotations can be implemented using division-free methods, such as the CORDIC method. They can also by parallelized by means of a systolic-array-type architecture. Moreover, the algorithm can be generalized to take account of updates to the the channel matrix. This could result in changes to the optimal ordering for estimating the signals.

Finally, the savings in computational complexity, as well as the numerical stability of the algorithm, suggest that it may be possible to implement the algorithm using a single commercial fixed-point DSP processor.

References

[1] G.J. Foschini and M.J. Gans. On limits of wireless communications in a fading environment when using multiple antennas. *Wireless Personal Communications*, 6(3):311, 1998.

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

[3] T. Kailath, A.H. Sayed, and B. Hassibi. *State Space Estimation*. Prentice-Hall, Englewood Cliffs, NJ, 1998.