Erik G. Larsson

MIMO Detection Methods: How They Work

In communications, the receiver often observes a linear superposition of separately transmitted information symbols. From the receiver's perspective, the problem is then to separate the transmitted symbols. This is basically an inverse problem with a finite-alphabet constraint. This lecture will present an accessible overview of state-of-the-art solutions to this problem.

RELEVANCE
The most important motivating application for the discussion here is receivers for multiple-antenna systems such as multiple-input, multiple-output (MIMO), where several transmit antennas simultaneously send different data streams. However, essentially the same problem occurs in systems where the channel itself introduces time- or frequency-dispersion, in multiuser detection, and in cancellation of crosstalk.

PREREQUISITES
General mathematical maturity is required along with knowledge of basic linear algebra and probability.

PROBLEM STATEMENT
Concisely, the problem is to recover the vector $s \in \mathbb{R}^n$ from an observation of the form

$$y = Hs + e, \quad y \in \mathbb{R}^m, \qquad (1)$$

where $H \in \mathbb{R}^{m \times n}$ is a known (typically, estimated beforehand) channel matrix and $e \in \mathbb{R}^m$ represents noise. We assume that $e \sim N(0, \sigma I)$. The elements of s, say s_k, belong to a finite alphabet $\mathcal{S}$ of size $|\mathcal{S}|$. Hence there are $|\mathcal{S}|^n$ possible vectors s. For simplicity of

our discussion, we assume that all quantities are real-valued. This is mostly a matter of notation, since $\mathbb{C}^n$ is isomorphic to $\mathbb{R}^{2n}$. We also assume that $m \geq n$, that is, (1) is not underdetermined, and that H has full column rank. This is so with probability one in most applications. We also assume that H has no specific structure. If H has structure, for example, if it is a Toeplitz matrix, then one should use algorithms that can exploit this structure.

We want to detect s in the maximum-likelihood (ML) sense. This is equivalent to

$$\text{The problem:} \quad \min_{s \in \mathcal{S}^n} \|y - Hs\|. \qquad (2)$$

Problem (2) is a finite-alphabet-constrained least-squares (LS) problem, which is known to be nondeterministic polynomial-time (NP)-hard. The complicating factor is of course the constraint $s \in \mathcal{S}^n$, otherwise (2) would be just classical LS regression.

SOLUTIONS
As a preparation, we introduce the QL-decomposition of H: $H = QL$, where $Q \in \mathbb{R}^{m \times n}$ is orthonormal ($Q^T Q = I$), and $L \in \mathbb{R}^{n \times n}$ is lower triangular. Then

$$
\begin{aligned}
\|y - Hs\|^2 &= \|QQ^T(y - Hs)\|^2 \\
&\quad + \|(I - QQ^T)(y - Hs)\|^2 \\
&= \|Q^T y - Ls\|^2 \\
&\quad + \|(I - QQ^T)y\|^2,
\end{aligned}
$$

where the last term does not depend on s. It follows that we can reformulate (2) as

$$\text{Equivalent problem:} \quad \min_{s \in \mathcal{S}^n} \|\tilde{y} - Ls\|,$$

$$\text{where} \qquad \tilde{y} \triangleq Q^T y \qquad (3)$$

or, in yet another equivalent form, as

$$\min_{\substack{\{s_1, \ldots, s_n\} \\ s_k \in \mathcal{S}}} \{f_1(s_1) + f_2(s_1, s_2) \\ + \cdots + f_n(s_1, \ldots, s_n)\},$$

where

$$f_k(s_1, \ldots, s_k) \triangleq \left(\tilde{y}_k - \sum_{l=1}^{k} L_{k,l} s_l \right)^2. \quad (4)$$

Problem (4) can be visualized as a decision tree with $n + 1$ layers, $|\mathcal{S}|$ branches emanating from each nonleaf node, and $|\mathcal{S}|^n$ leaf nodes. See Figure 1. To any branch, we associate a hypothetical decision on s_k, and the branch metric $f_k(s_1, \ldots, s_k)$. Also, to any node (except the root), we associate the cumulative metric $f_1(s_1) + \cdots + f_k(s_1, \ldots, s_k)$, which is just the sum of all branch metrics accumulated when traveling to that node from the root. Finally, to each node, we associate the symbols $\{s_1, \ldots, s_k\}$ it takes to reach there from the root.

Clearly, a naive but valid way of solving (4) would be to traverse the entire tree to find the leaf node with the smallest cumulative metric. However, such a brute-force search is extremely inefficient, since there are $|\mathcal{S}|^n$ leaf nodes to examine. We will now review some efficient, popular, but approximate solutions to (4).

ZERO-FORCING (ZF) DETECTOR
The ZF detector first solves (2), neglecting the constraint $s \in \mathcal{S}^n$

$$
\begin{aligned}
\tilde{s} &\triangleq \arg \min_{s \in \mathbb{R}^n} \|y - Hs\| \\
&= \arg \min_{s \in \mathbb{R}^n} \|\bar{y} - Ls\| = L^{-1}\tilde{y}. \quad (5)
\end{aligned}
$$

Of course, L^{-1} does not need to be explicitly computed. For example, one can do Gaussian elimination: take $\tilde{s}_1 = \tilde{y}_1/L_{1,1}$, then $\tilde{s}_2 = (\tilde{y}_2 - \tilde{s}_1 L_{2,1})/L_{2,2}$, and so forth. ZF then approximates (2) by projecting each $\tilde{s}_k$ onto the constellation $\mathcal{S}$
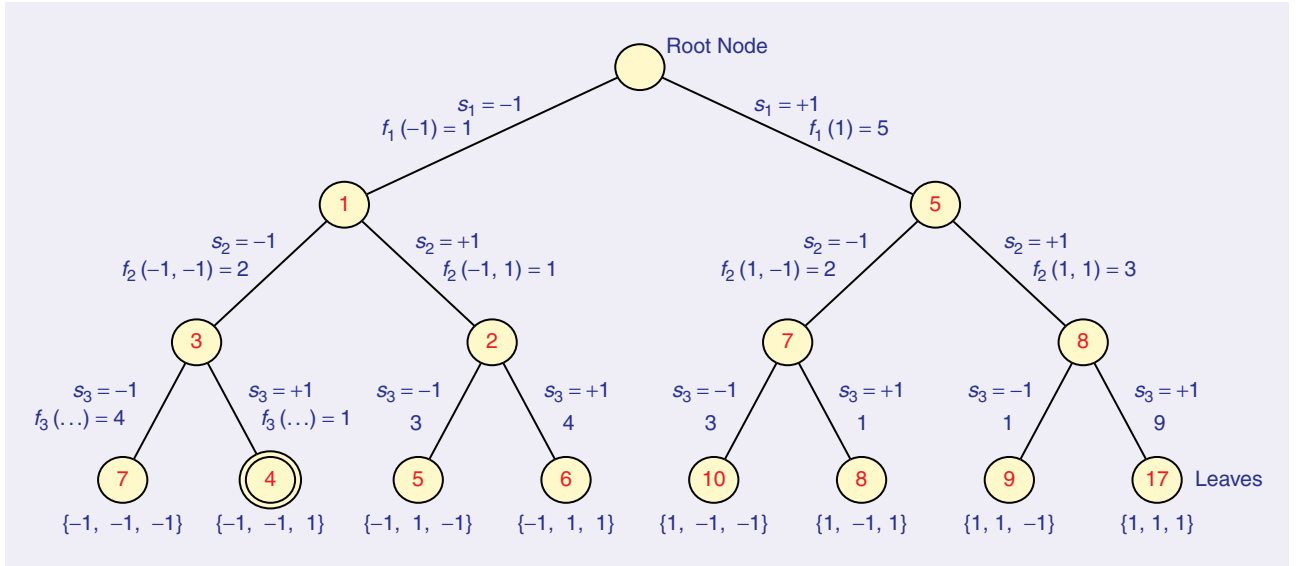
[FIG1] Problem (4) as a decision tree, exemplified for binary modulation (S = {–1, +1}, $|S|$ = 2) and n = 3. The branch metrics $f_k(s_1, \ldots, s_k)$ are in blue written next to each branch. The cumulative metrics $f_1(s_1)+ \ldots + f_k(s_1, \ldots, s_k)$ are written in red in the circles representing each node. The double circle represents the optimal (ML) decision.

$$\hat{s}_k = [\widetilde{s}_k] \triangleq \arg\min_{s_k \in S} |s_k - \widetilde{s}_k|. \quad (6)$$

We see that $\widetilde{s} = s + L^{-1}Q^T e$, so $\widetilde{s}$ in (5) is free of intersymbol interference. This is how ZF got its name. However, unfortunately ZF works poorly unless H is well conditioned. The reason is that the correlation between the noises in $\widetilde{s}_k$ is neglected in the projection operation (6). This correlation can be very strong, especially if H is ill conditioned (the covariance matrix of $\widetilde{s}$ is $\Sigma \triangleq \sigma \cdot (L^T L)^{-1}$). There are some variants of the ZF approach. For example, instead of computing $\widetilde{s}$ as in (5), one can use the MMSE estimate (take $\widetilde{s} = E[s|y]$). This can improve performance somewhat, but it does not overcome the fundamental problem of the approach.

ZF DETECTOR WITH DECISION FEEDBACK (ZF-DF)

Consider again ZF, and suppose we use Gaussian elimination to compute $\widetilde{s}$ in (5). ZF-DF [1] does exactly this, with the modification that it projects the symbols onto the constellation S in each step of the Gaussian elimination, rather than afterwards. More precisely,

1) Detect s_1

via $\hat{s}_1 = \arg\min_{s_1 \in S} f_1(s_1)$

$$= \left[\frac{\widetilde{y}_1}{L_{1,1}} \right].$$

2) Consider s_1 known ($s_1 = \hat{s}_1$) and detect s_2

via $\hat{s}_2 = \arg\min_{s_2 \in S} f_2(\hat{s}_1, s_2)$

$$= \left[\frac{\widetilde{y}_2 - \hat{s}_1 L_{2,1}}{L_{2,2}} \right].$$

3) Continue for $k = 3, \ldots, n$

$\hat{s}_k = \arg\min_{s_k \in S} f_k(\hat{s}_1, \ldots, \hat{s}_{k-1}, s_k)$

$$= \left[\frac{\widetilde{y}_k - \Sigma_{l=1}^{k-1} L_{k,l} \hat{s}_l}{L_{k,k}} \right].$$

In the decision-tree perspective, ZF-DF can be understood as just examining one single path down from the root. When deciding on s_k, it considers $s_1, \ldots, s_{k-1}$ known and takes the s_k that corresponds to the smallest branch metric. Clearly, after n steps we end up at one of the leaf nodes, but not necessarily in the one with the smallest cumulative metric.

In Figure 2(a), ZF-DF first chooses the left branch originating from the root (since $1 < 5$), then the right branch (since $2 > 1$) and at last the left branch (because $3 < 4$), reaching the leaf node with cumulative metric $1 + 1 + 3 = 5$.

The problem with ZF-DF is error propagation. If, due to noise, an incorrect symbol decision is taken in any of the n steps, then this error will propagate and many of the subsequent decisions are likely to be wrong as well. In

its simplest form (as explained above), ZF-DF detects s_k in the natural order, but this is not optimal. The detection order can be optimized to minimize the effects of error propagation. Not surprisingly, it is best to start with the symbol for which ZF produces the most reliable result: that is, the symbol s_k for which $\Sigma_{k,k}$ is the smallest, and then proceed to less and less reliable symbols. However, even with the optimal ordering, error propagation severely limits the performance.

SPHERE DECODING (SD)

The SD [2], [9] first selects a user parameter R, called the sphere radius. It then traverses the entire tree (from left to right, say). However, once it encounters a node with cumulative metric larger than R, then it does not follow down any branch from this node. Hence, in effect, SD enumerates all leaf nodes which lie inside the sphere $\|\widetilde{y} - Ls\|^2 \leq R$. This also explains the algorithm's name.

In Figure 2(b), we set the sphere radius to $R = 6$. The SD algorithm then traverses the tree from left to right. When it encounters the node "7" in the right subtree, for which $7 > 6 = R$, SD does not follow any branches emanating from it. Similarly, since $8 > 6$, SD does not visit
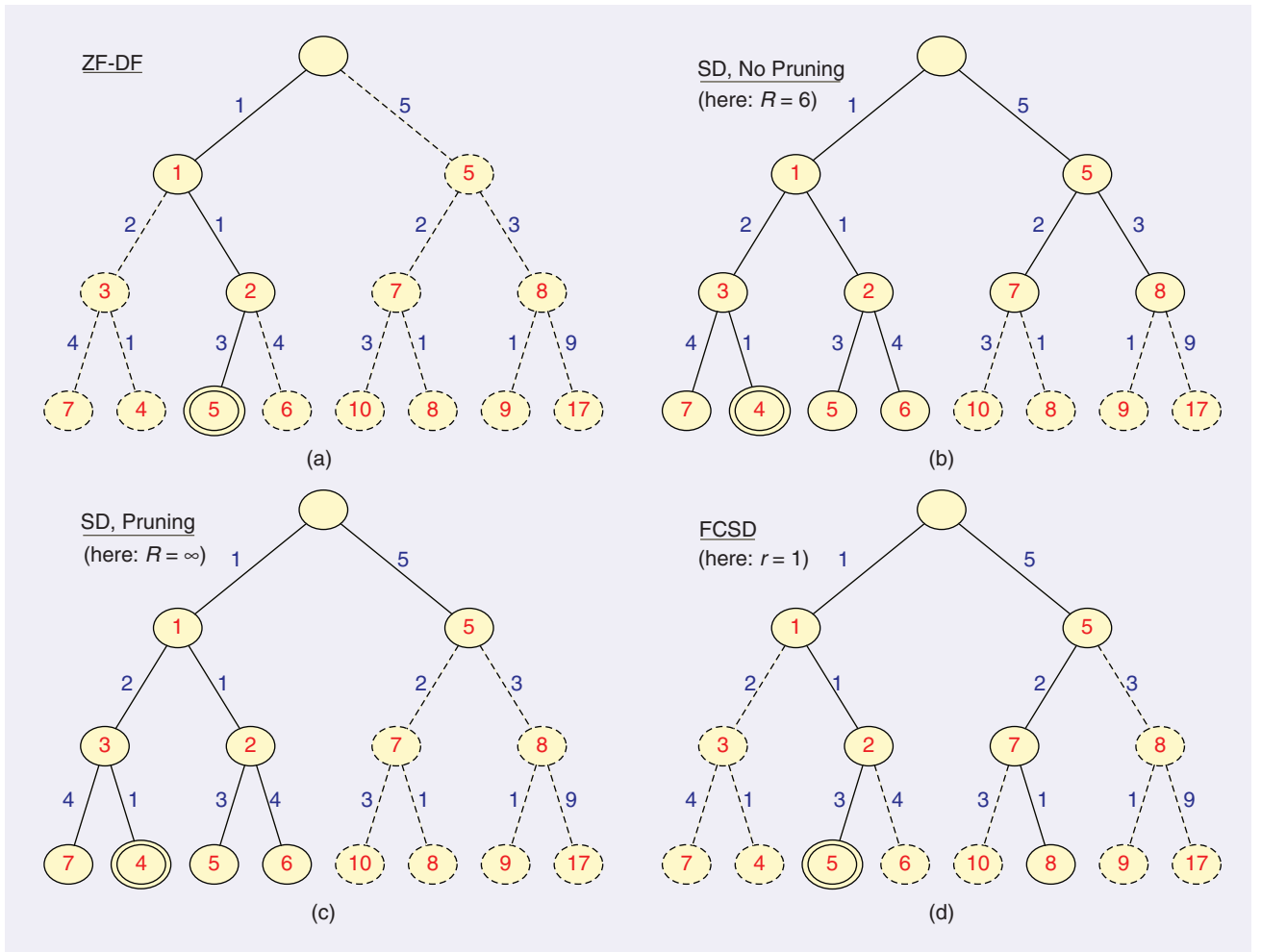
[FIG2] Illustration of detection algorithms as a tree search. Solid-line nodes and branches are visited. Dashed nodes and branches are not visited. The double circles represent the ultimate decisions on s. **(a) ZF-DF:** At each node, the symbol decision is based on choosing the branch with the smallest branch metric. **(b) SD, no pruning:** Only nodes with $\sum_{k=1}^{n} f_k(s_1, \ldots, s_k) \leq R$ are visited. **(c) SD, pruning:** Like SD, but after encountering a leaf node with cumulative metric M, the algorithm will set $R := M$. **(d) FCSD:** Visits all nodes on the rth layer, and proceeds with ZF-DF from these.

any branches below the node "8" in the rightmost subtree.

SD in this basic form can be much improved by a mechanism called pruning. The idea is this: Every time we reach a leaf node with cumulative metric M, we know that the solution to (4) must be contained in the sphere $\|\widetilde{y} - Ls\|^2 \leq M$. So if $M < R$, we can set $R := M$, and continue the algorithm with a smaller sphere radius. Effectively, we will adaptively prune the decision tree, and visit much fewer nodes than those in the original sphere. Figure 2(c) exemplifies the pruning. Here the radius is initialized to $R = \infty$, and then updated any time a leaf node is visited. For instance, when visiting the leaf node "4," R will be set to $R = 4$. This means that the algorithm

will not follow branches from nodes that have a branch metric larger than four. In particular, the algorithm does not examine any branches stemming from the node "5" in the right subtree.

The SD algorithm can be improved in many other ways, too. The symbols can be sorted in an arbitrary order, and this order can be optimized. Also, when traveling down along the branches from a given node, one may enumerate the branches either in the natural order or in a zigzag fashion (e.g., $s_k = \{-5, -3, -1, -1, 3, 5\}$ versus $s_k = \{-1, 1, -3, 3, -5, 5\}$). The SD algorithm is easy to implement, although the procedure cannot be directly parallelized. Given large enough initial radius R, SD will solve (2). However, depending on H, the time the algorithm takes to finish

will fluctuate, and may occasionally be very long.

FIXED-COMPLEXITY SPHERE DECODER (FCSD)

FCSD [3] is, strictly speaking, not really sphere decoding, but rather a clever combination of brute-force enumeration and a low-complexity, approximate detector. In view of the decision tree, FCSD visits all $|\mathcal{S}|^r$ nodes on layer r, where r, $0 \leq r \leq n$ is a user parameter. For each node on layer r, the algorithm considers $\{s_1, \ldots, s_r\}$ fixed and formulates and solves the subproblem

$$\min_{\substack{\{s_{r+1}, \ldots, s_n\} \\ s_k \in \mathcal{S}}} \{ f_{r+1}(s_1, \ldots, s_{r+1})$$

$$+ \cdots + f_n(s_1, \ldots, s_n) \}. \quad (7)$$

In effect, by doing so, FCSD will reach down to $|\mathcal{S}|^r$ of the $|\mathcal{S}|^n$ leaves. To form its symbol decisions, FCSD selects the leaf, among the leaves it has visited, which has the smallest cumulative metric $f_1(s_1) + \cdots + f_n(s_1, \ldots, s_n)$.

The subproblem (7) must be solved once for each combination $\{s_1, \ldots, s_r\}$, that is $|\mathcal{S}|^r$ times. FCSD does this approximately, using a low-complexity method (ZF or ZF-DF are good choices). This works well because (7) is overdetermined: there are n observations $(\tilde{y}_1, \ldots, \tilde{y}_n)$, but only $n - r$ unknowns $(s_{r+1}, \ldots, s_n)$. More precisely, the equivalent channel matrix when solving (7) will be a tall submatrix of H, which is generally much better conditioned than H.

Figure 2(d) illustrates the algorithm. Here $r = 1$. Thus, both nodes "1" and "5" in the layer closest to the root node are visited. Starting from each of these two nodes, a ZF-DF search is performed.

Naturally, the symbol ordering can be optimized. The optimal ordering is the one which renders the problem (7) most well-conditioned. This is achieved by sorting the symbols so that the most "difficult" symbols end up near the tree root. Note that "difficult symbol" is non-trivial to define precisely here, but intuitively think of it as a symbol s_k for which $\Sigma_{k,k}$ is large.

The choice of r offers a tradeoff between complexity and performance. FCSD solves (2) with high probability even for small r, it runs in constant time, and it has a natural parallel structure. Relatives of FCSD that produce soft output also exist [4].

SEMIDEFINITE-RELAXATION (SDR) DETECTOR

The idea behind SDR [5], [6] is to relax the finite-alphabet constraint on s into a matrix inequality and then use semidefinite programming to solve the resulting problem. We explain how it works, for binary phase-shift keying (BPSK) symbols ($s_k \in \{\pm 1\}$). Define

$$\overline{s} \triangleq \begin{bmatrix} s \\ 1 \end{bmatrix}, \quad S \triangleq \overline{s}\,\overline{s}^{\,T} = \begin{bmatrix} s \\ 1 \end{bmatrix}\begin{bmatrix} s^T & 1 \end{bmatrix},$$

$$\Psi \triangleq \begin{bmatrix} L^T L & -L^T \tilde{y} \\ -\tilde{y}^T L & 0 \end{bmatrix}.$$

Then

$$\|\tilde{y} - Ls\|^2 = \overline{s}^T \psi \overline{s} + \|\tilde{y}\|^2$$
$$= \mathrm{Trace}\{\Psi S\} + \|\tilde{y}\|^2$$

so solving (3) is the same as finding the vector $s \in \mathcal{S}^n$ that minimizes Trace $\{\Psi S\}$.

SDR exploits that the constraint $s \in \mathcal{S}^n$ is equivalent to requiring that rank $\{S\} = 1$, $\overline{s}_{n+1} = 1$ and diag $\{S\} = \{1, \ldots, 1\}$. It then proceeds by minimizing Trace $\{\Psi S\}$ with respect to S, but relaxes the rank constraint and instead requires that S be positive semidefinite. This relaxed problem is convex, and can be efficiently solved using so-called interior point methods. Once the matrix S is found, there are a variety of ways to determine s, for example to take the dominant eigenvector of S (forcing the last element to unity) and then project each element onto $\mathcal{S}$ like in (6). The error incurred by the relaxation is generally small.

LATTICE REDUCTION (LR) AIDED DETECTION

The idea behind LR [8], [9] is to transform the problem into a domain where the effective channel matrix is better conditioned than the original one. How does it work? If the constellation $\mathcal{S}$ is uniform, then $\mathcal{S}$ may be extended to a scaled enumeration of all integers, and $\mathcal{S}^n$ may be extended to a lattice $\overline{\mathcal{S}}^n$. For illustration, if $\mathcal{S} = \{-3, -1, 1, 3\}$, then $\overline{\mathcal{S}}^n = \{\ldots, -3, -1, 1, 3, \ldots\} \times \cdots \times \{\ldots, -3, -1, 1, 3, \ldots\}$. LR decides first on an $n \times n$ matrix T that has integer elements ($T_{k,l} \in \mathbb{Z}$) and which maps the lattice $\overline{\mathcal{S}}^n$ onto itself: $Ts \in \overline{\mathcal{S}}^n \ \forall s \in \overline{\mathcal{S}}^n$. That is, T should be invertible, and its inverse should have integer elements. This happens precisely if its determinant has unit magnitude: $|T| = \pm 1$. Naturally, there are many such matrices ($T = \pm I$ is one trivial example). LR chooses such a matrix T for which, additionally, HT is well conditioned. It then computes

$$\hat{s}' \triangleq \arg\min_{s' \in \overline{\mathcal{S}}^n} \|y - (HT)s'\|. \quad (8)$$

Problem (8) is comparatively easy, since HT is well conditioned, and simple methods like ZF or ZF-DF generally

work well. Once $\hat{s}'$ is found, it is transformed back to the original coordinate system by taking $\hat{s} = T^{-1}\hat{s}'$.

LR contains two critical steps. First, a suitable matrix T must be found. There are good methods for this (e.g., see references in [8], [9]). This is computationally expensive, but if the channel H stays constant for a long time then the cost of finding T may be shared between many instances of (2) and complexity is less of an issue. The other problem is that while the solution $\hat{s}$ always belongs to $\overline{\mathcal{S}}^n$, it may not belong to $\mathcal{S}^n$. Namely, some of its elements may be beyond the borders of the original constellation. Hence a clipping-type operation is necessary and this will introduce some loss.

SOFT DECISIONS

In practice, each symbol s_k typically is composed of information-carrying bits, say $\{b_{k,1}, \ldots, b_{k,p}\}$. It is then of interest to take decisions on the individual bits $b_{k,i}$, and often, also to quantify how reliable these decisions are. Such reliability information about a bit is called a "soft decision," and is typically expressed via the probability ratio

$$\frac{P(b_{k,i} = 1|y)}{P(b_{k,i} = 0|y)} = \frac{\sum_{s:b_{k,i}(s)=1} P(s|y)}{\sum_{s:b_{k,i}(s)=0} P(s|y)}$$
$$= \frac{\sum_{s:b_{k,i}(s)=1} \exp\left(-\frac{1}{\sigma}\|y - Hs\|^2\right) P(s)}{\sum_{s:b_{k,i}(s)=0} \exp\left(-\frac{1}{\sigma}\|y - Hs\|^2\right) P(s)}.$$
$$(9)$$

Here "$s:b_{k,i}(s) = \beta$" means all s for which the ith bit of s_k is equal to β, and $P(s)$ is the probability that the transmitter sent s. To derive (9), use Bayes rule and the Gaussian assumption made on e [4].

Fortunately, (9) can be relatively well approximated by replacing the two sums in (9) with their largest terms. To find these maximum terms is a slightly modified version of (2), at least if all s are equally likely so that $P(s) = 1/|\mathcal{S}|^n$. Hence, if (2) can be solved, good approximations to (9) are available too. An even better approximation to (9) is obtained if more terms are retained, i.e.,

not only the largest one [7], [4]. This is naturally accomplished by many of the methods we discussed, by simply including the terms corresponding to all leaf nodes in the decision tree that the algorithm has visited. If the symbol vectors have different a priori probabilities, then under certain circumstances $P(s)$ can be incorporated by appropriately modifying y and H [6], [4].

CONCLUSIONS

The goal of this lecture has been to provide an overview of approaches to (2), in the communications receiver context. Which method is the best in practice? This depends much on the purpose of solving (2): what error rate can be tolerated, what is the ultimate measure of performance (e.g., frame-error-rate, worst-case complexity, or average complexity), and what computational platform is used. Additionally, the bits in s may be part of a larger code word and

different s vectors in that code word may either see the same H (slow fading) or many different realizations of H (fast fading). This complicates the picture, because notions that are important in slow fading (such as spatial diversity) are less important in fast fading, where diversity is provided anyway by time variations. Detection for MIMO has been an active field for more than ten years, and this research will probably continue for some time.

AUTHOR

Erik G. Larsson (erik.larsson@isy.liu.se) is a professor and head of division for communication systems in the EE department (ISY) of Linköping University, Sweden. For more information, visit www.commsys.isy.liu.se.

REFERENCES
[1] G. D. Golden, G. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, "Detection algorithm and initial laboratory results using V-BLAST space-time communications architecture," *IEE Electron. Lett.*, vol. 35, pp. 14–16, Jan. 1999.

[2] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1639–1642, July 1999.

[3] L. G. Barbero and J. S. Thompson, "Fixing the complexity of the sphere decoder for MIMO detection," *IEEE Trans. Wireless Commun.*, vol. 7, pp. 2131–2142, June 2008.

[4] E. G. Larsson and J. Jaldén, "Soft MIMO detection at fixed complexity via partial marginalization," *IEEE Trans. Signal Processing*, vol. 56, pp. 3397–3407, Aug. 2008.

[5] P. Tan and L. Rasmussen, "The application of semidefinite programming for detection in CDMA," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 1442–1449, Aug. 2001.

[6] B. Steingrimsson, Z.-Q. Luo, and K. Wong, "Soft quasi-maximum-likelihood detection for multiple-antenna wireless channels," *IEEE Trans. Signal Processing*, vol. 51, no. 11, pp. 2710–2719, Nov. 2003.

[7] B. M. Hochwald and S. Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[8] C. Windpassinger and R. Fischer, "Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction," in *Proc. IEEE Information Theory Workshop*, 2003, pp. 345–348.

[9] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inform. Theory*, vol. 48, pp. 2201–2214, Aug. 2002.

[SP]

[dsp **TIPS&TRICKS**]

AUTHOR

Maurice Givens (Maurice.Givens@gastechnology.org) is a specialist in the research and design of digital signal processing at Gas Technology Institute and an adjunct lecturer at Wright College, Chicago. He is a registered professional engineer, a Senior Member of the IEEE, and a senior member of NARTE.

REFERENCES
[1] R. Harris, D. Chabries, and P. Bishop, "A variable step (VS) adaptive filter algorithm," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-34, pp. 309–316, Apr. 1986.

[2] J. Evans, P. Xue, and B. Liu, "Analysis and implementation of variable step size adaptive algorithms," *IEEE Trans. Signal Processing*, vol. 41, pp. 2517–2535, Aug. 1993.

[3] T. Haweel and P. Clarkson, "A class of order statistic LMS algorithm," *IEEE Trans. Signal Processing*, vol. 40, pp. 44–53, Jan. 1992.

[4] T. Aboulnasr and K. Mayyas, "A robust variable step-size LMS-type algorithm: Analysis and simulations," *IEEE Trans. Signal Processing*, vol. 45, pp. 631–639, Mar. 1997.

[5] D. Pazaitis and A. Constantinides, "A novel kurtosis driven variable step-size adaptive algorithm," *IEEE Trans. Signal Processing*, vol. 47, pp. 864–872, Mar. 1999.

[6] R. Kwong and E. Johnston, "A variable step size LMS algorithm," *IEEE Trans. Signal Processing*, vol. 40, pp. 1633–1642, July 1992.

[SP]