6.D SIMULATION COMPARISON OF SIC METHODS: WHITENING, V-BLAST AND MMSE V-BLAST

1. INTRODUCTION

The simulation in this worksheet compares the performance of three successive interference cancellation algorithms: whitening, V-BLAST and MMSE V-BLAST. They differ in the order in which bits are detected and in whether they use ZF or MMSE to obtain antenna weights. All detectors work on the same received signals, for more accurate comparisons.

Notation is as follows:

K number of users M number of antennas

b length-K vector of bits from all users

A KxK diagonal array of amplitudes of all users

C MxK array of gains; Gaussian, variance 1/2; column k $\mathbf{C}^{\langle k \rangle}$ is for user k.

n length-M array of noises on the antennas; unit variance (i.e. normalised so N_o=1). With the noise normalisation, we have each amplitude as $A_k = \sqrt{2 \cdot \Gamma_k}$, where Γ_k is the user-k SNR.

Several useful procedures are in the Appendix.

2. One-Shot Test of Simulation

In this section, we build up the simulation, to see how the methods work. In Appendices B to D, they are condensed to single procedures. Section 3 uses all three over many, many trials. Highlighted regions are parameter input.

K := 3 M := 3 users and antennas, resp. $\mathbf{I_K}$:= identity(K) $\mathbf{I_M}$:= identity(M)

b := datavec(K) transmitted data (for another realisation, click on equation and press F9)

SNR of user 0 in dB dB ratio other users' power to user 0 power

ΓdB0 := 5 ΛdB := 0

array of SNRs

Γ_{dB_0} := ΓdB0 i := 1 .. K − 1 Γ_{dB_i} := ΓdB0 − ΛdB Γ := $\overrightarrow{\text{nat}\left(\Gamma_{dB}\right)}$

A := $\text{diag}\left(\overrightarrow{\sqrt{2 \cdot \Gamma}}\right)$ make the diagonal matrix of amplitudes (sqrt computed component by component)

$\rho := 1$ correlation coefficient between channel gain estimates and their true values

$\mathbf{V} := \frac{1}{\sqrt{2}} \cdot \text{gaussarray}(M, K)$ generate the matrix of channel gain *estimates*, all i.i.d., variance 1/2 (for another realisation, click on equation and press F9)

$\mathbf{C} := \rho \cdot \mathbf{V} + \sqrt{1 - \rho^2} \cdot \frac{1}{\sqrt{2}} \cdot \text{gaussarray}(M, K)$ generate the matrix of channel gains, all i.i.d., variance 1/2 (for another realisation, click on equation and press F9)

$\mathbf{F} := \mathbf{V} \cdot \mathbf{A}$ more concise notation

$\mathbf{n} := \text{gaussarray}(M, 1)$ the vector of noises, variance 1 (for another realisation, click on equation and press F9)

$\mathbf{y} := \mathbf{C} \cdot \mathbf{A} \cdot \mathbf{b} + \mathbf{n}$ vector of matched filter outputs

2.1 Whitening SIC

In this method, we first sort the signals in order of increasing power, then do the SIC starting at the last bit.

$k := 0 .. K - 1 \qquad P_k := \left(\overline{\mathbf{F}^T} \cdot \mathbf{F} \right)_{k,k}$

The next line sorts by increasing power. The 2nd column (column 1) contains the column index into **F.**

$P := \text{csort}(\text{augment}(P, \text{index}(K)), 0) \qquad \text{order} := P^{\langle 1 \rangle}$

Rearrange the columns accordingly

$\mathbf{temp}^{\langle k \rangle} := \mathbf{F}^{\langle (\text{order}_k) \rangle} \qquad \mathbf{F'} := \mathbf{temp}$

and check that it worked:

$$\mathbf{F} = \begin{pmatrix} -2.775 - 1.641i & 1.138 - 2.311i & -0.817 - 3.223i \\ 2.304 + 1.491i & 3.471 - 0.251i & 3.663 + 0.206i \\ -1.604 - 1.193i & -3.209 + 1.025i & 0.884 - 4.163i \end{pmatrix} \qquad \text{order} = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$$

$$\mathbf{F'} = \begin{pmatrix} -2.775 - 1.641i & 1.138 - 2.311i & -0.817 - 3.223i \\ 2.304 + 1.491i & 3.471 - 0.251i & 3.663 + 0.206i \\ -1.604 - 1.193i & -3.209 + 1.025i & 0.884 - 4.163i \end{pmatrix}$$

Now Cholesky factorise the reordered Gram matrix as $\overline{\mathbf{F'}^{T}}\cdot\mathbf{F'} = \mathbf{L}\cdot\overline{\mathbf{L}^{T}}$ by performing Gram-Schmidt orthogonalisation on **F**.

$$\text{orth} := \text{GS}(\mathbf{F'}) \qquad \mathbf{U} := \text{submatrix}(\text{orth}, 0, K-1, 0, K-1) \qquad \mathbf{L} := \overline{\mathbf{U}^{T}}$$

Calculate triangularised sufficient statistics

$$\mathbf{z} := \mathbf{L}^{-1}\cdot\overline{\mathbf{F'}^{T}}\cdot\mathbf{y} \quad \text{so that} \quad \mathbf{z} = \mathbf{T}\cdot\mathbf{b} + a$$

where $\mathbf{T} := \mathbf{L}^{-1}\cdot\overline{\mathbf{F'}^{T}}\cdot\mathbf{F'}$ is upper triangular - see below

$$\mathbf{T} = \begin{pmatrix} 4.682 & 2.602 - 0.629i & 4.239 + 2.21i \\ 0 & 4.788 & 0.509 + 0.023i \\ 0 & 0 & 4.417 \end{pmatrix}$$

Now do the SIC on the reordered data bits, starting from last bit (the one with highest power):

$$b'_{K-1} := \text{sgn}\left(\frac{\mathbf{z}_{K-1}}{\mathbf{T}_{K-1,K-1}}\right) \qquad \text{start the SIC}$$

$$j := K-2\,..\,0 \qquad b'_j := \text{sgn}\left(\frac{\mathbf{z}_j - \sum_{i=j+1}^{K-1} \mathbf{T}_{j,i}\cdot b'_i}{\mathbf{T}_{j,j}}\right) \qquad \text{rest of SIC}$$

and restore the order

$$\mathbf{b}_{\mathbf{w}_{\text{order}_k}} := b'_k \qquad \text{compare these:} \qquad \mathbf{b}_{\mathbf{w}} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}$$

The whitening SIC algorithm is condensed to a single procedure in Appendix B.

2.2 V-BLAST SIC

Here we follow the V-BLAST algorithm by Golden et al, Electronics Letters, 7th Jan 1999, vol. 35, no. 1, pp 14-15.

bitndx := index(K) initialise bit indexes to natural numbers (we need them to keep track of which bit we are deciding at each stage)

Select and detect the first bit

$\mathbf{G} := \left(\overline{\mathbf{F}^{T}} \cdot \mathbf{F}\right)^{-1} \cdot \overline{\mathbf{F}^{T}}$ $\mathbf{W} := \mathbf{G}^{T}$ weight matrix in $\mathbf{W}^{T} \cdot \mathbf{y}$ is the pseudoinverse of **F**

$P_V := \text{diagxtract}\left(\overline{\mathbf{W}^{T}} \cdot \mathbf{W}\right)$ vector of column norms of **W**

Sort by increasing column norm. After the sort, column 1 contains permuted column indexes, column 2 contains permuted bit indexes:

$P_V := \text{csort}\left(\text{augment}\left(\text{augment}\left(P_V, \text{index}(K)\right), \text{bitndx}\right), 0\right)$ $\text{order} := P_V^{\langle 1 \rangle}$ $\text{bitndx} := P_V^{\langle 2 \rangle}$

$$\text{order} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \qquad \text{bitndx} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

The column with index in location 0 of order has minimum norm of weight vector, i.e., min noise, i.e. max SNR, so we make a decision on that bit.

$b := \text{sgn}\left[\left(\mathbf{W}^{\langle (\text{order}_0) \rangle^{T}} \cdot \mathbf{y}\right)_0\right]$ save decision $\mathbf{b}_{\mathbf{v}_{(\text{bitndx}_0)}} := b$

Deflate the system by subtracting that bit's contribution to **y,** then removing the corresponding column of **F** and row of bitndx.

$\mathbf{y} := \mathbf{y} - \mathbf{F}^{\langle (\text{order}_0) \rangle} \cdot b$ subtract that bit's contribution

Permute columns of **F** to put the just-processed column into column 0 for easy removal. To check, look at **F,** step by step. First, before manipulations:

$$\mathbf{F} = \begin{pmatrix} -2.775 - 1.641i & 1.138 - 2.311i & -0.817 - 3.223i \\ 2.304 + 1.491i & 3.471 - 0.251i & 3.663 + 0.206i \\ -1.604 - 1.193i & -3.209 + 1.025i & 0.884 - 4.163i \end{pmatrix}$$

Reorder the columns

$$\mathbf{temp} := \mathbf{F} \qquad k := 0 .. K - 1 \qquad \mathbf{temp}^{\langle k \rangle} := \mathbf{F}^{\langle (\mathrm{order}_k) \rangle} \qquad \mathbf{F} := \mathbf{temp}$$

$$\mathbf{F} = \begin{pmatrix} 1.138 - 2.311i & -0.817 - 3.223i & -2.775 - 1.641i \\ 3.471 - 0.251i & 3.663 + 0.206i & 2.304 + 1.491i \\ -3.209 + 1.025i & 0.884 - 4.163i & -1.604 - 1.193i \end{pmatrix}$$

and reduce the order:

$$\mathbf{F} := \mathrm{submatrix}(\mathbf{F}, 0, M - 1, 1, \mathrm{cols}(\mathbf{F}) - 1) \qquad \mathrm{bitndx} := \mathrm{submatrix}(\mathrm{bitndx}, 1, \mathrm{rows}(\mathrm{order}) - 1, 0, 0)$$

$$\mathbf{F} = \begin{pmatrix} -0.817 - 3.223i & -2.775 - 1.641i \\ 3.663 + 0.206i & 2.304 + 1.491i \\ 0.884 - 4.163i & -1.604 - 1.193i \end{pmatrix} \qquad \mathrm{bitndx} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

We are ready to do it again. On to stage 2...

Select and detect the first bit

$$\mathbf{G} := \left(\overline{\mathbf{F}^{\mathrm{T}}} \cdot \mathbf{F} \right)^{-1} \cdot \overline{\mathbf{F}^{\mathrm{T}}} \qquad \mathbf{W} := \mathbf{G}^{\mathrm{T}}$$

$$P_V := \mathrm{diagxtract}\left(\overline{\mathbf{W}^{\mathrm{T}}} \cdot \mathbf{W} \right) \qquad \text{vector of column norms of } \mathbf{W}$$

Sort by increasing column norm. After the sort, column 1 contains permuted column indexes, column 2 contains permuted bit indexes:

$$P_V := \mathrm{csort}\left(\mathrm{augment}\left(\mathrm{augment}\left(P_V, \mathrm{index}(K - 1) \right), \mathrm{bitndx} \right), 0 \right) \qquad \mathrm{order} := P_V^{\langle 1 \rangle} \qquad \mathrm{bitndx} := P_V^{\langle 2 \rangle}$$

$$\mathrm{order} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \mathrm{bitndx} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

The column with index in location 0 of order has minimum norm of weight vector, i.e., min noise, i.e. max SNR, so we make a decision on that bit.

$$b := \mathrm{sgn}\left[\left(\mathbf{W}^{\langle(\mathrm{order}_0)\rangle^T}\cdot\mathbf{y}\right)_0\right] \qquad \text{save decision} \qquad \mathbf{b_v}_{(\mathrm{bitndx}_0)} := b$$

Deflate the system by subtracting that bit's contribution to **y,** then removing the corresponding column of **F** and row of bitndx.

$$\mathbf{y} := \mathbf{y} - \mathbf{F}^{\langle(\mathrm{order}_0)\rangle}\cdot b \qquad \text{subtract that bit's contribution}$$

Reorder the columns

$$\mathbf{temp} := \mathbf{F} \qquad k := 0..K-2 \qquad \mathbf{temp}^{\langle k\rangle} := \mathbf{F}^{\langle(\mathrm{order}_k)\rangle} \qquad \mathbf{F} := \mathbf{temp}$$

and reduce the order:

$$\mathbf{F} := \mathrm{submatrix}(\mathbf{F}, 0, M-1, 1, \mathrm{cols}(\mathbf{F})-1) \qquad \mathrm{bitndx} := \mathrm{submatrix}(\mathrm{bitndx}, 1, \mathrm{rows}(\mathrm{order})-1, 0, 0)$$

$$\mathbf{F} = \begin{pmatrix} -2.775 - 1.641i \\ 2.304 + 1.491i \\ -1.604 - 1.193i \end{pmatrix} \qquad \mathrm{bitndx} = (0)$$

The last stage is easy. No need to form the pseudoinverse, since only one user remaining. We have

$$b := \mathrm{sgn}\left[\left(\overline{\mathbf{F}^T}\cdot\mathbf{y}\right)_0\right] \qquad \mathbf{b_v}_{(\mathrm{bitndx}_0)} := b$$

Compare decision with transmitted bits:

$$\mathbf{b_v} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}$$

This algorithm is condensed to a single procedure in Appendix C.

MMSE V-BLAST SIC

V-BLAST does not have to use the zero forcing algorithm, as noted in the paper by Golden et al. Here we explore an MMSE variant. It should be significantly better than ZF V-BLAST when the signal powers are different, since the strongest signal, which is likely to be selected for first detection, does not have to squander its degrees of freedom on complete suppression of the other users.

Recall that the MMSE detector forms

$$\mathbf{bhat} = \overline{\mathbf{W}^{T}} \cdot \mathbf{y} \quad \text{where} \quad \mathbf{W} = \mathbf{F} \cdot \left(\overline{\mathbf{F}^{T}} \cdot \mathbf{F} + 2 \cdot \mathbf{I} \right)^{-1}$$

and that the resulting MMSEs, component by component, are

$$s_{\mathbf{e}}{}^{2} = \text{diagxtract}\left[\frac{1}{2} \cdot \mathbf{I} - \frac{1}{2} \cdot \left(\overline{\mathbf{F}^{T}} \cdot \mathbf{F} + 2 \cdot \mathbf{I} \right)^{-1} \cdot \overline{\mathbf{F}^{T}} \cdot \mathbf{F} \right]$$

Therefore, at each stage, we select for detection the bit with the smallest mean squared error; equivalently with the *largest* component in

$$\text{diagxtract}\left[\left(\overline{\mathbf{F}^{T}} \cdot \mathbf{F} + 2 \cdot \mathbf{I} \right)^{-1} \cdot \overline{\mathbf{F}^{T}} \cdot \mathbf{F} \right]$$

The remaining operations are identical to the ZF V-BLAST above. A procedure is in Appendix D.

3. THE FULL SIMULATION

Here we put the three SIC detectors head-to-head in the same environment. May the best detector win.

$$\begin{aligned}
&\text{MUDfight}\left(K, M, \mathbf{A}, \rho, N_{sim}\right) := \\
&\quad \mathbf{E} \leftarrow \text{zeros}(K, 3) \\
&\quad \text{for } n \in 1 .. N_{sim} \\
&\qquad \mathbf{b} \leftarrow \text{datavec}(K) \\
&\qquad \mathbf{V} \leftarrow \frac{1}{\sqrt{2}} \cdot \text{gaussarray}(M, K) \\
&\qquad \mathbf{C} \leftarrow \rho \cdot \mathbf{V} + \sqrt{\frac{1 - \rho^{2}}{2}} \cdot \text{gaussarray}(M, K) \\
&\qquad \mathbf{y} \leftarrow \mathbf{C} \cdot \mathbf{A} \cdot \mathbf{b} + \text{gaussarray}(M, 1) \\
&\qquad \mathbf{F} \leftarrow \mathbf{V} \cdot \mathbf{A} \\
&\qquad \mathbf{b_w} \leftarrow \text{whitenSIC}(\mathbf{y}, \mathbf{F}) \\
&\qquad \mathbf{E}^{\langle 0 \rangle} \leftarrow \mathbf{E}^{\langle 0 \rangle} + \text{errs}\left(\mathbf{b_w}, \mathbf{b}\right) \\
&\qquad \mathbf{b_v} \leftarrow \text{vblast}(\mathbf{y}, \mathbf{F}) \\
&\qquad \mathbf{E}^{\langle 1 \rangle} \leftarrow \mathbf{E}^{\langle 1 \rangle} + \text{errs}\left(\mathbf{b_v}, \mathbf{b}\right) \\
&\qquad \mathbf{b_m} \leftarrow \text{vblastmmse}(\mathbf{y}, \mathbf{F}) \\
&\qquad \mathbf{E}^{\langle 2 \rangle} \leftarrow \mathbf{E}^{\langle 2 \rangle} + \text{errs}\left(\mathbf{b_m}, \mathbf{b}\right) \\
&\quad \frac{\mathbf{E}}{N_{sim}}
\end{aligned}$$

Here is one run. We'll do several of them further below.

$$\text{MUDfight}(K, M, \mathbf{A}, \rho, 100) = \begin{pmatrix} 0.01 & 0.01 & 0 \\ 0 & 0.01 & 0 \\ 0.01 & 0.02 & 0 \end{pmatrix}$$

A column for each detector: whitening SIC, V-BLAST and MMSE V-BLAST. Row 0 is user 0, row 1 is user 1, etc.

And this precedure loops through a sequence of SNR values:

$$\text{curves}(M, G\mathbf{0dB}, L\mathbf{dB}, \rho, N_{sim}) :=$$

K ← length(L**dB**)

all ← zeros(1, 7)

for i ∈ 0 .. length(G**0dB**) − 1

  G**dB** ← G**0dB**$_i$ · ones(K, 1) + L**dB**

  A ← $\overrightarrow{\sqrt{2 \cdot \text{nat}(G\mathbf{dB})}}$

  A ← diag(A)

  BERs ← MUDfight(K, M, **A**, ρ, N_{sim})

  keep ← augment(augment(G**dB**, K · ones(K, 1)), M · ones(K, 1))

  keep ← augment(keep, N_{sim} · ones(K, 1))

  keep ← augment(keep, **BERs**)

  all ← stack(all, keep)

all

Next, we run and save the simulation for BERs at a sequence of SNR values. This group is for **equipower users.** It's in the area below that we harvest the results of our work in building the simulation.

K := 3 M := 3 users and antennas, resp. ρ := 1 correlation coefficient

k := 0 .. K − 1 L**dB**$_k$:= 0 ratio (in dB) of user i power to user 0 power

$$G\mathbf{0dB} := \begin{pmatrix} 5 \\ 10 \\ 15 \\ 20 \\ 25 \end{pmatrix} \qquad N_{sim} := 500000 \qquad L\mathbf{dB} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

results1 := curves(M, G**0dB**, L**dB**, ρ, N_{sim})

results1 := submatrix(results1, 1, rows(results1) − 1, 0, cols(results1) − 1) trim off the row of zeros

WRITEPRN("equiSIC500.txt") := results1

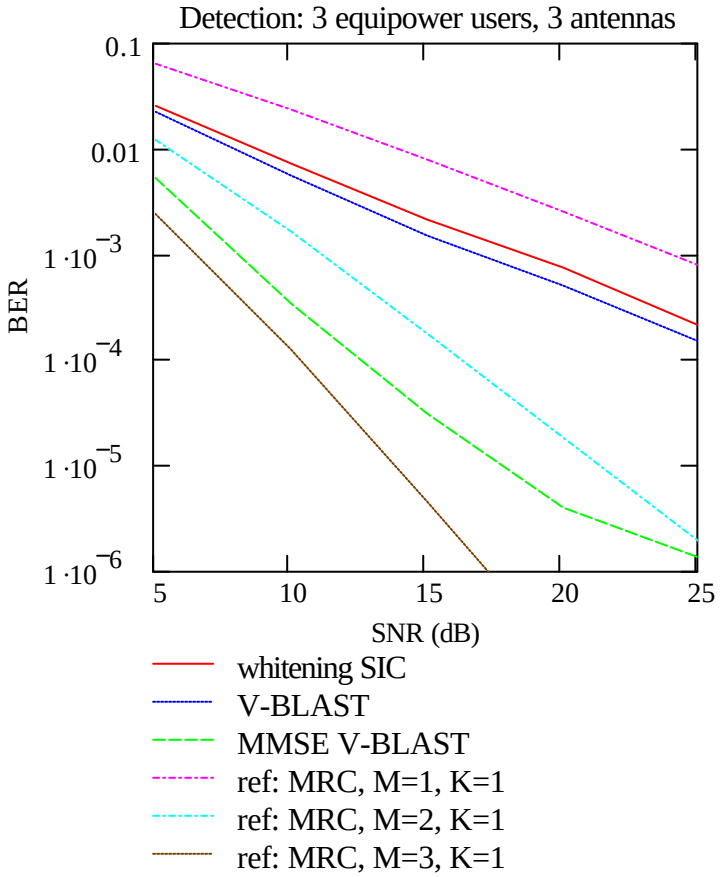
Now, read in the results of a lengthy simulation: temp := READPRN("equiSIC.txt")

Average the rows, since equipower, same BER (set up here for K=3). Transpose, to work with columns.

$$temp := temp^{T} \qquad i := 0 .. \frac{cols(temp)}{3} - 1$$

$$BER^{\langle i \rangle} := \frac{1}{3} \cdot \left(temp^{\langle 3 \cdot i \rangle} + temp^{\langle 3 \cdot i+1 \rangle} + temp^{\langle 3 \cdot i+2 \rangle} \right) \qquad BER := BER^{T}$$

Number of trials for each point on the curves: $3 \cdot BER_{0,3} = 1.5 \times 10^{6}$



Observations

* The two algorithms based on ZF (whitening and V-BLAST) have diversity order one, since they are dominated by the error probability of the first user to be detected, and that user has diversity order one.

* V-BLAST is a little better than whitening, since it makes a more intelligent choice of detection order.

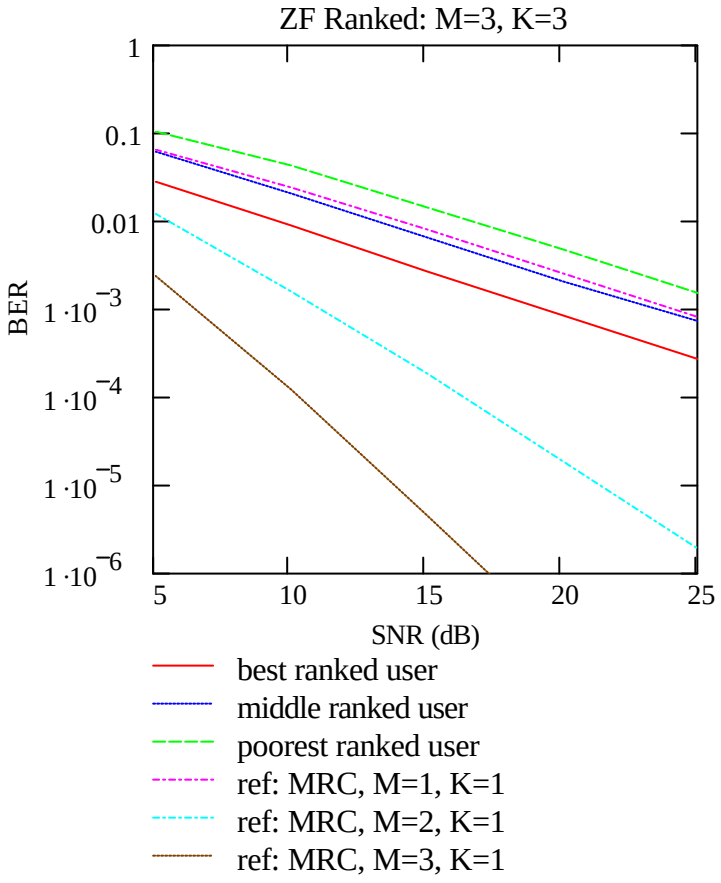
* MMSE V-BLAST is almost unbelievably good, in terms of both absolute error rate and the effective order of diversity over much of its displayed range. Its order of diversity is asymptotically one, since MMSE tends to ZF when the signals grossly dominate the noise.

These results demand explanation. Our original results in Section 6.B for linear detection of equipower signals showed that the average BER of MMSE was a little better than that of ZF, and they both tracked the single diversity MRC curve. Why would MMSE be so much better than ZF in a SIC regime?

For an answer, we look more closely at the original (non-SIC) detection by ZF and MMSE. For each realisation of the gains, we can rank the users by the same criteria that we use in the first iteration of BLAST (Appendices C, D); that is, by the diagonal elements of

$\overline{\mathbf{W}^{T} \cdot \mathbf{W}}$ V-BLAST $\qquad$ $\overline{\mathbf{F}^{T} \cdot \mathbf{W}}$ MMSE V-BLAST (note different **W**'s)

Appendix E contains simulation routines that separate out the error rates, by rank, for ZF and MMSE. The best ranked one is always the first to be selected for detection in BLAST SIC.



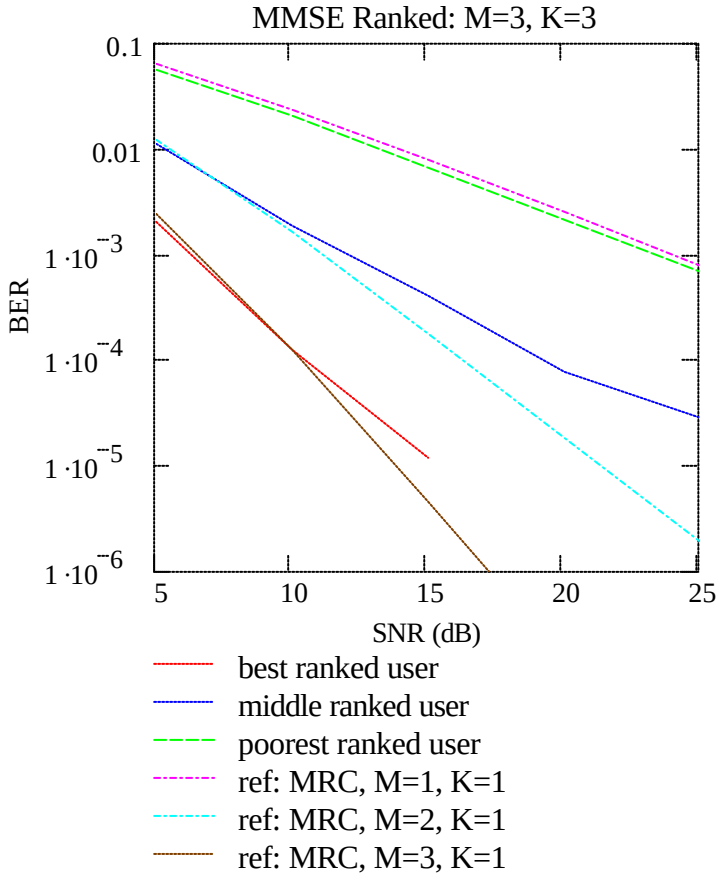
Observations

* The ranking order of users (users 0, 1, 2) changes with every realisation of the gains.

* The average of the three curves is exactly the average BER curve for ZF that we saw in Section 6.B.

* In ZF, even the instantaneously best ranked user is only a little better than the 2nd and 3rd ranked. They, in turn, are almost as bad as diversity order one detection.

* This explains why ZF V-BLAST is not hugely better than the diversity order one curve.



Observations

* The ranking order of users (users 0, 1, 2) changes with every realisation of the gains.

* The average of the three MMSE curves is exactly the average BER curve for MMSE that we saw in Section 6.B.

* In MMSE, the instantaneously best ranked user enjoys phenomenally good performance (order 3 for lower SNRs) and the 2nd user is close to diversity order 2. MMSE doesn't waste degrees of freedom suppressing users who are instantaneously at or below the noise level.

* This explains the great performance of MMSE V-BLAST. Since the first user to be selected for detection has very low BER, its removal and the deflation of the set is likely to be correct, providing the remaining users with an additional order of diversity. The same argument can be applied to the next stage.

4. NEAR-FAR EFFECTS

In this section, we examine the effects of dissimilar power levels among users. User 1 is a few dB below user 0, user 2 a few dB below user 1.

$K := 3$ $M := 3$ users and antennas, resp.

$L\mathbf{dB}_0 := 0$ $L\mathbf{dB}_1 := -10$ ratio (in dB) of user i power to user 0 power

$G\mathbf{0dB} := (5 \;\; 10 \;\; 15 \;\; 20 \;\; 25)^T$ $N_{sim} := 500000$ $L\mathbf{dB} := \begin{pmatrix} 0 \\ -5 \\ -10 \end{pmatrix}$

$results2 := curves(M, G\mathbf{0dB}, L\mathbf{dB}, \rho, N_{sim})$

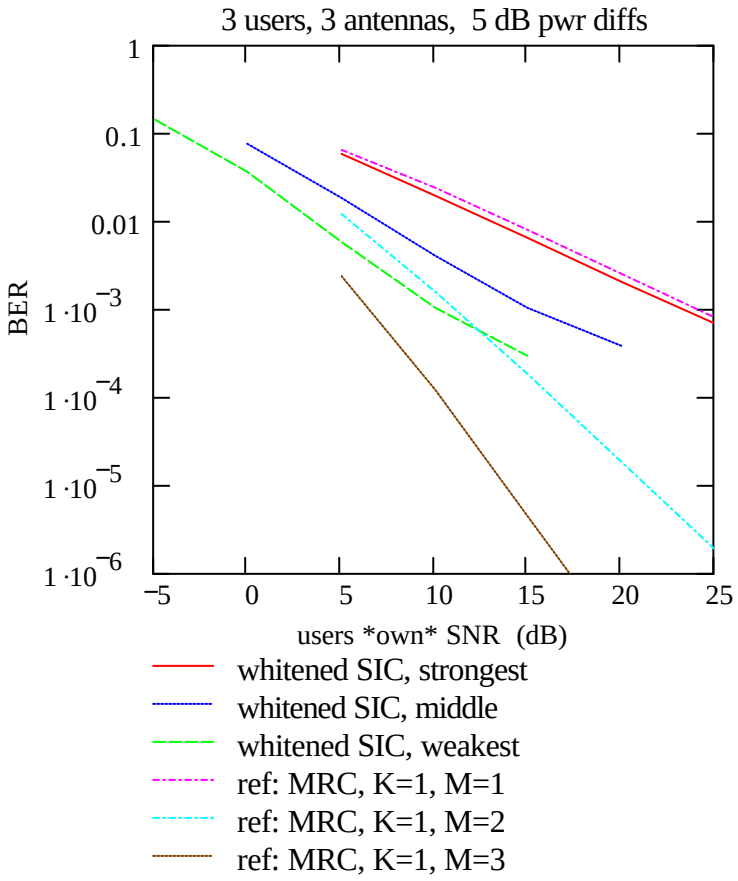
$results2 := submatrix(results2, 1, rows(results2) - 1, 0, cols(results2) - 1)$ trim off the row of zeros

$WRITEPRN("pdiffSIC500.txt") := results2$

Read in the results of one or more simulation runs:

$BER := READPRN("pdiffSIC.txt")$ $r := rows(BER)$ $i := 0, 3 .. r - 2$

We plot against the user's *own* SNR. The first is for whitening SIC. Number of trials: $BER_{0,3} = 5 \times 10^5$



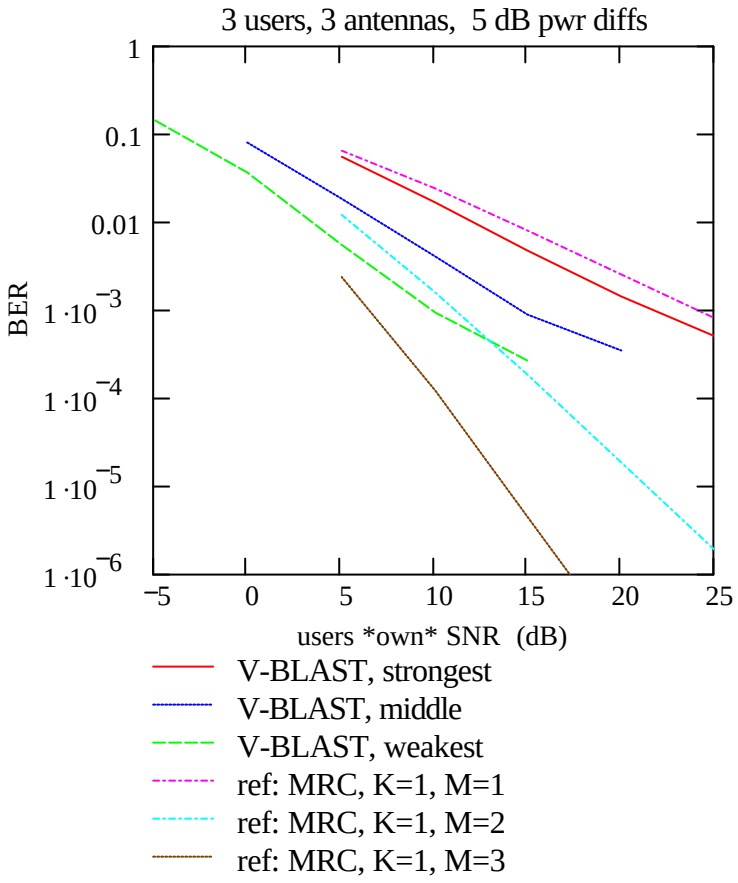
Observations

* Users are labeled strongest, middle, weakest by *mean square* power. This is not always the same as instantaneous power, which determines detection order. The strongest user is not always the first to be detected.

* The strongest user's performance degrades significantly by the presence of weaker users. Unit order of diversity, as expected.

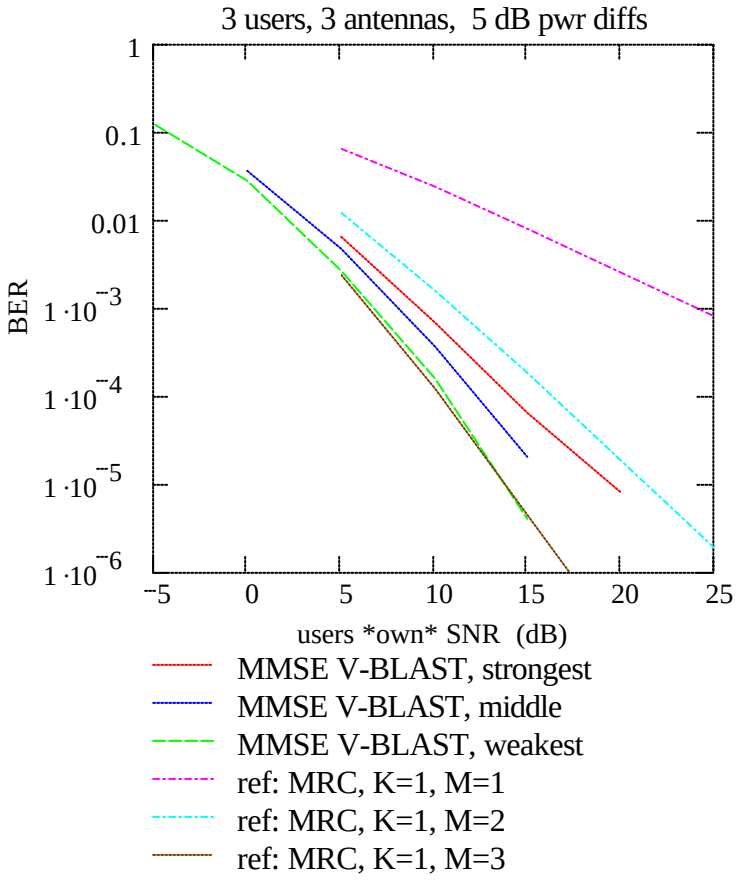
* At any operating point, the weak users have poorer BER than the strong one - they are plotted here against *own* SNR.

Next, V-BLAST, again plotted against the user's *own* SNR. First, the ZF version, then MMSE V-BLAST.



Observations

* Almost identical to whitening, because they are both based on ZF.



Observations

* All users have diversity order between 2 and 3 for the displayed range.

* The weakest user follows the diversity 3 curve, just what it would have experienced if there were no other users. Decisions on other-user bits are almost always correct (they are 5 dB and 10 dB stronger).

* The strongest user experiences degradation due to the presence of weaker users. Occasionally, they are detected first, but their instantaneous quality is unlikely to be much more than that of the strongest user, so they can adversely affect the BER of the strongest user.

It is worth noting that greater power disparities (e.g., 2nd user 10 dB lower, 3rd user 20 dB lower than s=trongest) cause all users to track the ideal MRC, K=1, M=3 curve. That is because the weaker users have less disturbing influence on the strongest user, who is almost always selected as the first to be detected.

Conclusions with respect to SIC detection of users with different powers:

* The weaker users are not much affected by presence of the stronger users, since the first decisions are likely to be correct. Caveat: this is for perfect channel state information.

* The stronger users are adversely affected by the presence of the weaker users if the power disparity is relatively minor. For large power disparities, the weaker users are too weak to make much difference to the strong user.

APPENDIX A: USEFUL FUNCTIONS

This appendix contains a simulation to determine the statistics of Hamming error for pure MMSE decisions. It begins with a few "housekeeping" functions that simplify the subsequent simulation.

Basic definitions and functions

$$\text{smaller}(x,y) \equiv \text{if}(x < y, x, y) \qquad \text{larger}(x,y) \equiv \text{if}(x > y, x, y)$$

$$\text{dB}(x) \equiv \text{if}\left(x > 10^{-14}, 10\cdot\log(x), -140\right) \qquad \text{nat}(x) \equiv 10^{0.1\cdot x}$$

$$\text{sgn}(x) \equiv \text{if}(\text{Re}(x) \geq 0, 1, -1)$$

Manipulations of binary data vectors

Hamming error vector

$$\text{errs}(\mathbf{b},\mathbf{c}) \equiv \left|\begin{array}{l} \mathbf{e} \leftarrow \mathbf{b} - \mathbf{c} \\ \text{for } i \in 0..\text{length}(\mathbf{e}) - 1 \\ \quad \mathbf{e}_i \leftarrow \text{if}\left(\mathbf{e}_i = 0, 0, 1\right) \\ \mathbf{e} \end{array}\right.$$

Conversions between integers and binary data vectors (of +1,-1)

$$\text{int\_to\_bvec}(n,N) \equiv \left|\begin{array}{l} i \leftarrow 0 \\ \text{while } i < N \\ \quad \left|\begin{array}{l} \text{rem} \leftarrow \text{mod}(n,2) \\ b_i \leftarrow \text{if}(\text{rem} = 1, 1, -1) \\ n \leftarrow \text{floor}\left(\frac{n}{2}\right) \\ i \leftarrow i + 1 \end{array}\right. \\ b \end{array}\right.$$

$$\text{bvec\_to\_int}(b) \equiv \left|\begin{array}{l} \text{sum} \leftarrow 0 \\ \text{for } i \in 0..\text{rows}(b) - 1 \\ \quad \text{sum} \leftarrow \text{sum} + 2^{i-1}\cdot\left(b_i + 1\right) \\ \text{sum} \end{array}\right.$$

Generate random vectors and arrays

Generate random data vector

$$\text{datavec}(N) \equiv \text{int\_to\_bvec}\left(\text{floor}\left(\text{rnd}\left(2^N\right)\right), N\right)$$

Generate array (N_r x N_c) of variance 1 complex Gaussian

$$\text{cgauss}(x) \equiv \sqrt{-2\cdot\ln(\text{rnd}(1))}\cdot\exp\left(j\cdot\text{rnd}\left(2\cdot\pi\right)\right)$$ unit variance complex Gaussian

$$\text{gaussarray}\left(N_r, N_c\right) \equiv \left|\begin{array}{l} \text{for } ir \in 0..N_r - 1 \\ \quad \text{for } ic \in 0..N_c - 1 \\ \quad\quad A_{ir,ic} \leftarrow \text{cgauss}(ir) \\ A \end{array}\right.$$

Array manipulations

Make an all-zeros array

$$\text{zeros}\left(N_r, N_c\right) \equiv \left|\begin{array}{l} \text{for } ir \in 0..N_r - 1 \\ \quad \text{for } ic \in 0..N_c - 1 \\ \quad\quad A_{ir,ic} \leftarrow 0 \\ A \end{array}\right.$$

Make an all-ones array

$$\text{ones}\left(N_r, N_c\right) \equiv \left|\begin{array}{l} \text{for } ir \in 0..N_r - 1 \\ \quad \text{for } ic \in 0..N_c - 1 \\ \quad\quad A_{ir,ic} \leftarrow 1 \\ A \end{array}\right.$$

find smallest/largest in an array; return the value and its index:

$$\text{smallest}(x) \equiv \left|\begin{array}{l} \text{small} \leftarrow 10^{15} \\ \text{for } i \in 0..\text{rows}(x) - 1 \\ \quad \text{if } x_i < \text{small} \\ \quad\quad \left|\begin{array}{l} \text{small} \leftarrow x_i \\ \text{i\_small} \leftarrow i \end{array}\right. \\ (\text{small}\quad \text{i\_small})^T \end{array}\right.$$

$$\text{largest}(x) \equiv \left|\begin{array}{l} \text{large} \leftarrow -10^{15} \\ \text{for } i \in 0..\text{rows}(x) - 1 \\ \quad \text{if } x_i > \text{large} \\ \quad\quad \left|\begin{array}{l} \text{large} \leftarrow x_i \\ \text{i\_large} \leftarrow i \end{array}\right. \\ (\text{large}\quad \text{i\_large})^T \end{array}\right.$$

vector containing its own indexes (for sort algorithms)

$$\text{index}(N) \equiv \left|\begin{array}{l} \text{for } n \in 0..N - 1 \\ \quad \text{ind}_n \leftarrow n \\ \text{ind} \end{array}\right.$$

extract the diagonal of a square matrix, return as vector

$$\text{diagxtract}(\mathbf{X}) \equiv \left|\begin{array}{l} \text{for } i \in 0..\text{cols}(\mathbf{X}) - 1 \\ \quad d_i \leftarrow \mathbf{X}_{i,i} \\ d \end{array}\right.$$

Gram-Schmidt orthogonalisation. Performs G-S on the vectors comprising the columns of C. The orthonormal set forms the columns of Φ and the corresponding coefficients are in the upper triangular U, so that $C=\Phi U$. The arrays U and Φ are returned, stacked.

$$
\begin{aligned}
&GS(C) \equiv \left|
\begin{array}{l}
K \leftarrow \mathrm{cols}(C) \\
U_{0,0} \leftarrow \left|C^{\langle 0 \rangle}\right| \\
\Phi^{\langle 0 \rangle} \leftarrow \dfrac{C^{\langle 0 \rangle}}{U_{0,0}} \\
\text{for } k \in 1..K-1 \\
\quad \left|
\begin{array}{l}
\text{for } i \in 0..k-1 \\
\quad U_{i,k} \leftarrow \left(\overline{\left(\Phi^{\langle i \rangle^T}\right)} \cdot C^{\langle k \rangle}\right)_0 \\
e \leftarrow C^{\langle k \rangle} - \sum_{i=0}^{k-1} U_{i,k} \cdot \Phi^{\langle i \rangle} \\
U_{k,k} \leftarrow |e| \\
\Phi^{\langle k \rangle} \leftarrow \dfrac{e}{U_{k,k}}
\end{array}
\right. \\
\mathrm{stack}(U, \Phi)
\end{array}
\right.
\end{aligned}
$$

APPENDIX B: THE WHITENING SIC PROCEDURE

The steps of the whitening SIC algorithm from Section 2 are here condensed into a single procedure.

$\text{whitenSIC}(\mathbf{y}, \mathbf{F}) \equiv$

$K \leftarrow \text{cols}(\mathbf{F})$

for $k \in 0 .. K-1$

$P_k \leftarrow \left(\overline{\mathbf{F}^{T}} \cdot \mathbf{F}\right)_{k,k}$

sort signals by increasing power; column 1 of **P** holds the ordering

$P \leftarrow \text{csort}(\text{augment}(P, \text{index}(K)), 0)$

$\text{order} \leftarrow P^{\langle 1 \rangle}$

reorder columns of **F** to form **F'**

for $k \in 0 .. K-1$

$\text{temp}^{\langle k \rangle} \leftarrow \mathbf{F}^{\langle (\text{order}_k) \rangle}$

$\mathbf{F}' \leftarrow \text{temp}$

Cholesky factorise the Gram matrix by doing Gram-Schmidt on columns of **F'**. Lower triangular L.

$\text{orth} \leftarrow \text{GS}(\mathbf{F}')$

$\mathbf{L} \leftarrow \overline{\text{submatrix}(\text{orth}, 0, K-1, 0, K-1)^{T}}$

sufficient statistics **z**

$\mathbf{z} \leftarrow \mathbf{L}^{-1} \cdot \overline{\mathbf{F}'^{T}} \cdot \mathbf{y}$

and upper triangular **T**

$\mathbf{T} \leftarrow \mathbf{L}^{-1} \cdot \overline{\mathbf{F}'^{T}} \cdot \mathbf{F}'$

SIC starts with highest indexed data bit

$b'_{K-1} \leftarrow \text{sgn}\left(\frac{\mathbf{z}_{K-1}}{\mathbf{T}_{K-1,K-1}}\right)$

for $j \in K-2 .. 0$

and moves down to bit 0

$b'_j \leftarrow \text{sgn}\left(\frac{\mathbf{z}_j - \sum_{i=j+1}^{K-1} \mathbf{T}_{j,i} \cdot b'_i}{\mathbf{T}_{j,j}}\right)$

restore the order of decisions

for $k \in 0 .. K-1$

$\mathbf{b_w}_{\text{order}_k} \leftarrow b'_k$

$\mathbf{b_w}$

APPENDIX C: THE V-BLAST ALGORITHM

This procedure condenses the operations in Section 2.

$$
\begin{aligned}
&\text{vblast}(\mathbf{y},\mathbf{F}) \equiv \\
&\quad K \leftarrow \text{cols}(\mathbf{F}) \\
&\quad M \leftarrow \text{rows}(\mathbf{F}) \\
&\quad \text{bitndx} \leftarrow \text{index}(K) \\
&\quad \text{for } k \in 0..K-2 \\
&\qquad \mathbf{W} \leftarrow \mathbf{F}\cdot\left(\overline{\mathbf{F}^{T}}\cdot\mathbf{F}\right)^{-1} \\
&\qquad P_V \leftarrow \text{diagxtract}\left(\overline{\mathbf{W}^{T}}\cdot\mathbf{W}\right) \\
&\qquad P_V \leftarrow \text{csort}\left(\text{augment}\left(\text{augment}\left(P_V,\text{index}(K-k)\right),\text{bitndx}\right),0\right) \\
&\qquad \text{order} \leftarrow P_V^{\langle 1\rangle} \\
&\qquad \text{bitndx} \leftarrow P_V^{\langle 2\rangle} \\
&\qquad b \leftarrow \text{sgn}\left[\left(\overline{\mathbf{W}^{\langle(\text{order}_0)\rangle^{T}}}\cdot\mathbf{y}\right)_0\right] \\
&\qquad \mathbf{b_V}_{(\text{bitndx}_0)} \leftarrow b \\
&\qquad \mathbf{y} \leftarrow \mathbf{y} - \mathbf{F}^{\langle(\text{order}_0)\rangle}\cdot b \\
&\qquad \mathbf{temp} \leftarrow \mathbf{F} \\
&\qquad \text{for } i \in 0..K-k-1 \\
&\qquad\quad \mathbf{temp}^{\langle i\rangle} \leftarrow \mathbf{F}^{\langle(\text{order}_i)\rangle} \\
&\qquad \mathbf{F} \leftarrow \mathbf{temp} \\
&\qquad \mathbf{F} \leftarrow \text{submatrix}(\mathbf{F},0,M-1,1,\text{cols}(\mathbf{F})-1) \\
&\qquad \text{bitndx} \leftarrow \text{submatrix}(\text{bitndx},1,\text{rows}(\text{bitndx})-1,0,0) \\
&\quad \mathbf{b_V}_{(\text{bitndx}_0)} \leftarrow \text{sgn}\left[\left(\overline{\mathbf{F}^{T}}\cdot\mathbf{y}\right)_0\right] \\
&\quad \mathbf{b_V}
\end{aligned}
$$

APPENDIX D: THE MMSE V-BLAST ALGORITHM

This procedure is like standard VBLAST, but uses MMSE, instead of ZF, and selects the bit on the basis of minimum mean squared error. Equivalently, maximises mean sq value of the estimate of the data bit.

$$
\begin{aligned}
&\text{vblastmmse}(\mathbf{y},\mathbf{F}) \equiv \\
&\quad K \leftarrow \text{cols}(\mathbf{F}) \\
&\quad M \leftarrow \text{rows}(\mathbf{F}) \\
&\quad \text{bitndx} \leftarrow \text{index}(K) \\
&\quad \text{for } k \in 0..K-2 \\
&\qquad \mathbf{W} \leftarrow \mathbf{F}\cdot\left(\overline{\mathbf{F}^{T}}\cdot\mathbf{F} + 2\,\text{identity}(K-k)\right)^{-1} \\
&\qquad P_V \leftarrow -\text{diagxtract}\left(\overline{\mathbf{F}^{T}}\cdot\mathbf{W}\right) \\
&\qquad P_V \leftarrow \text{csort}\left(\text{augment}\left(\text{augment}\left(P_V, \text{index}(K-k)\right), \text{bitndx}\right), 0\right) \\
&\qquad \text{order} \leftarrow P_V^{\langle 1\rangle} \\
&\qquad \text{bitndx} \leftarrow P_V^{\langle 2\rangle} \\
&\qquad b \leftarrow \text{sgn}\left[\left(\overline{\mathbf{W}^{\langle(\text{order}_0)\rangle^{T}}}\cdot\mathbf{y}\right)_0\right] \\
&\qquad \mathbf{b_v}_{(\text{bitndx}_0)} \leftarrow b \\
&\qquad \mathbf{y} \leftarrow \mathbf{y} - \mathbf{F}^{\langle(\text{order}_0)\rangle}\cdot b \\
&\qquad \mathbf{temp} \leftarrow \mathbf{F} \\
&\qquad \text{for } i \in 0..K-k-1 \\
&\qquad\quad \mathbf{temp}^{\langle i\rangle} \leftarrow \mathbf{F}^{\langle(\text{order}_i)\rangle} \\
&\qquad \mathbf{F} \leftarrow \mathbf{temp} \\
&\qquad \mathbf{F} \leftarrow \text{submatrix}(\mathbf{F}, 0, M-1, 1, \text{cols}(\mathbf{F})-1) \\
&\qquad \text{bitndx} \leftarrow \text{submatrix}(\text{bitndx}, 1, \text{rows}(\text{bitndx})-1, 0, 0) \\
&\quad \mathbf{b_v}_{(\text{bitndx}_0)} \leftarrow \text{sgn}\left[\left(\overline{\mathbf{F}^{T}}\cdot\mathbf{y}\right)_0\right] \\
&\quad \mathbf{b_v}
\end{aligned}
$$

APPENDIX E: DECISIONS ORDERED BY INSTANTANEOUS BEST TO WORST USER

$$\text{ZFranked}(\mathbf{y},\mathbf{F},\mathbf{b}) \equiv \left[\begin{array}{l} K \leftarrow \text{cols}(\mathbf{F}) \\ M \leftarrow \text{rows}(\mathbf{F}) \\ \text{bitndx} \leftarrow \text{index}(K) \\ \mathbf{W} \leftarrow \mathbf{F}\cdot\left(\overline{\mathbf{F}^{T}}\cdot\mathbf{F}\right)^{-1} \\ P_V \leftarrow \text{diagxtract}\left(\overline{\mathbf{W}^{T}}\cdot\mathbf{W}\right) \\ P_V \leftarrow \text{csort}\left(\text{augment}\left(P_V,\text{bitndx}\right),0\right) \\ \text{order} \leftarrow P_V^{\langle 1\rangle} \\ \mathbf{decn} \leftarrow \overline{\mathbf{W}^{T}}\cdot\mathbf{y} \\ \mathbf{decn} \leftarrow \overrightarrow{\text{sgn}(\mathbf{decn})} \\ \mathbf{etemp} \leftarrow \text{errs}(\mathbf{decn},\mathbf{b}) \\ \text{for } i\in 0..K-1 \\ \quad \mathbf{e}_i \leftarrow \mathbf{etemp}_{(\text{order}_i)} \\ \mathbf{e} \end{array}\right]$$

Returns a vector of decision errors for pure ZF, but ranked, so that the first (row 0) is for the users with the best weight vector, etc. The average of the three error rates is the BER of all users in ZF, irrespective of ranking.

$$\text{MMSEranked}(\mathbf{y},\mathbf{F},\mathbf{b}) \equiv \left|\begin{array}{l} K \leftarrow \text{cols}(\mathbf{F}) \\ M \leftarrow \text{rows}(\mathbf{F}) \\ \text{bitndx} \leftarrow \text{index}(K) \\ \mathbf{W} \leftarrow \mathbf{F}\cdot\left(\overline{\mathbf{F}^{T}}\cdot\mathbf{F}+2\,\text{identity}(K)\right)^{-1} \\ P_V \leftarrow -\text{diagxtract}\left(\overline{\mathbf{F}^{T}}\cdot\mathbf{W}\right) \\ P_V \leftarrow \text{csort}\left(\text{augment}\left(P_V,\text{bitndx}\right),0\right) \\ \text{order} \leftarrow P_V^{\langle 1\rangle} \\ \mathbf{decn} \leftarrow \overline{\mathbf{W}^{T}}\cdot\mathbf{y} \\ \mathbf{decn} \leftarrow \overrightarrow{\text{sgn}(\mathbf{decn})} \\ \mathbf{etemp} \leftarrow \text{errs}(\mathbf{decn},\mathbf{b}) \\ \text{for } i\in 0..K-1 \\ \quad \mathbf{e}_i \leftarrow \mathbf{etemp}_{(\text{order}_i)} \\ \mathbf{e} \end{array}\right.$$

Returns a vector of decision errors for pure MMSE, but ranked, so that the first (row 0) is for the users with the best weight vector, etc. The average of the three error rates is the BER of all users in ZF, irrespective of ranking.

$$\text{MUDrank}(K, M, \mathbf{A}, \rho, N_{sim}) \equiv$$

$\mathbf{E} \leftarrow \text{zeros}(K, 2)$

for $n \in 1 .. N_{sim}$

$\mathbf{b} \leftarrow \text{datavec}(K)$

$\mathbf{V} \leftarrow \frac{1}{\sqrt{2}} \cdot \text{gaussarray}(M, K)$

$\mathbf{C} \leftarrow \rho \cdot \mathbf{V} + \sqrt{\frac{1 - \rho^2}{2}} \cdot \text{gaussarray}(M, K)$

$\mathbf{y} \leftarrow \mathbf{C} \cdot \mathbf{A} \cdot \mathbf{b} + \text{gaussarray}(M, 1)$

$\mathbf{F} \leftarrow \mathbf{V} \cdot \mathbf{A}$

$\mathbf{E}^{\langle 0 \rangle} \leftarrow \mathbf{E}^{\langle 0 \rangle} + \text{ZFranked}(\mathbf{y}, \mathbf{F}, \mathbf{b})$

$\mathbf{E}^{\langle 1 \rangle} \leftarrow \mathbf{E}^{\langle 1 \rangle} + \text{MMSEranked}(\mathbf{y}, \mathbf{F}, \mathbf{b})$

$\frac{\mathbf{E}}{N_{sim}}$

simulation run to get the BERs ranked by instantaneous weight vector, best to worst.

Traces out the curves.

$$\text{ranker}(M, G\mathbf{0dB}, L\mathbf{dB}, \rho, N_{sim}) \equiv$$

$K \leftarrow \text{length}(L\mathbf{dB})$

$\text{all} \leftarrow \text{zeros}(1, 6)$

for $i \in 0 .. \text{length}(G\mathbf{0dB}) - 1$

$G\mathbf{dB} \leftarrow G\mathbf{0dB}_i \cdot \text{ones}(K, 1) + L\mathbf{dB}$

$A \leftarrow \overrightarrow{\sqrt{2 \cdot \text{nat}(G\mathbf{dB})}}$

$\mathbf{A} \leftarrow \text{diag}(A)$

$\mathbf{BERs} \leftarrow \text{MUDrank}(K, M, \mathbf{A}, \rho, N_{sim})$

$\text{keep} \leftarrow \text{augment}(\text{augment}(G\mathbf{dB}, K \cdot \text{ones}(K, 1)), M \cdot \text{ones}(K, 1))$

$\text{keep} \leftarrow \text{augment}(\text{keep}, N_{sim} \cdot \text{ones}(K, 1))$

$\text{keep} \leftarrow \text{augment}(\text{keep}, \mathbf{BERs})$

$\text{all} \leftarrow \text{stack}(\text{all}, \text{keep})$

all