Contents

Chapter 13

The Gaussian Vector Multiple Access Channel

The Gaussian multiple-access channel (MAC) first appeared in Section 12.2, where examples added the sum of two or more users' signals to Gaussian noise before processing by a single dimension of a common receiver. This chapter returns to this Gaussian MAC and more completely analyzes the common output with possibly multiple dimensions for inputs and outputs, and thus generalizes to the "Vector MAC," which is abbreviated in these notes as **MAC**[1] This **MAC** will be completely handled by the GDFE theory of Chapter 5.

Section 13.1 models the **MAC**, particularly describing the H matrix. This detailed model allows a more thorough specification of **MAC** -input implementation of a particular point $\boldsymbol{b}$ within the rate region $c(\boldsymbol{b})$, finding a generalization of Section 12.2's simultaneous water-filling for the maximum **MAC** rate sum. Section 13.2 introduces the simple **iterative water-filling** algorithm for determining a simultaneous water-filling solution. Iterative water-filling then can be used to maximize the users' rate sum on any **MAC**. Iterative water-filling can be used with no order, or equivalently its results can be used with any of $U!$ orders, and will for each situation produce a possibly different set of users' input covariances that all have the same maximum rate sum. Each such ordered-decoder maximum-rate-sum point $\boldsymbol{b}$ is then a **vertex** of that maximum rate sum. Section 13.2 also describes a fixed-margin (FM) iterative water-filling procedure that will produce a set of input covariances that achieve any rate vector $\boldsymbol{b} \in c(\boldsymbol{b})$, although there will be no absolute guarantee for such points that the corresponding energy constraints for all the users will be satisfied or that any user, or the set of users, necessarily uses minimum sum energy. This FM IW will have uses as a component of other procedures.

Section 13.3 then proceeds to **Vector DMT** solutions that simplify the **MAC** implementation for many practical situations and allow essentially a tone-by-tone approach to the **MAC**. In particular, time(dimension)-sharing will yield to frequency-sharing. An otherwise difficult rate-region construction associated with dimension-sharing of various user-order rate-point choices simplifies, with large numbers of tones, to a single best order(same on all tones) for implementation of any rate-region point. This best rate-point solution is no longer SWF at non-rate-sum-maximizing points. An alternate procedure (of M. Mohseni) determines the input covariances and best order simultaneously in Section 13.4. Section 13.5 progresses to tracing of the rate region or determining the correct the best energy and bit distribution for any point with the **MAC** rate region. Section 13.5 also suggests a distributed implementation that may be important for applications where the central **MAC** controller does not attempt to implement complex algorithms for time-variation of the channels and maintains the use of bit swapping to keep a solution close to optimal with minimal complexity.

[1] Thus, the boldface **MAC** implies vector and Gaussian.

13.1 The Gaussian Vector MAC Model

This section develops a consistent vector model for the **MAC** and generalizes SWF for this model.

13.1.1 Vector Model Basics

Figure 13.1 illustrates the basic vector AWGN channel with the linear channel matrix H. L_x denotes the number of non-temporal input dimensions per user (non temporal means not associated with stacking of N time samples to form a packet, so like number of antennas or number of lines in a DSL system). In situations where different users may have different numbers of dimensions, the largest number of said dimensions defines L_x (and dummy dimensions are inserted into the analysis on other users). Similarly, L_y is the number of non-temporal dimensions at the common **MAC** output. Each element of the channel matrix H_u is $L_yN \times L_xN$ (or more precisely $L_yN \times L_x(N+\nu)$ with guard bands[2], see Chapter 4). The channel is thus described by the $L_yN \times L_x(N+\nu)U$ matrix H. If all outputs and all inputs could

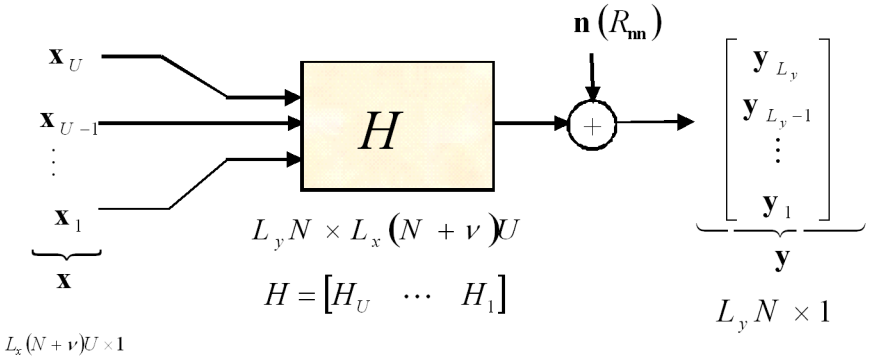

Figure 13.1: The Gaussian **MAC**.

be coordinated as vectors, then the channel is truly a single-user channel at the physical layer even if many users' signals are carried at a higher level of use. In this vector single-user case, the vector coding of Chapter 4, with $R_{\boldsymbol{nn}}^{-1/2}H = F\Lambda M^*$ determined by singular value decomposition, can be used with transmit signal

$$\boldsymbol{x} = M\boldsymbol{X} \quad , \tag{13.1}$$

with receive matched matrix-filter output

$$\boldsymbol{Y} = F^*\boldsymbol{y} \quad , \tag{13.2}$$

and with a water-filling distribution of energy computed for up to $\tilde{N} = \min(L_yN, L_x(N+\nu)U)$ channel input dimensions. Defining the channel signal to noise gains as

$$g_n \triangleq \lambda_n^2 \quad , \tag{13.3}$$

the water-filling distribution is well-known as

$$\mathcal{E}_n + \frac{1}{g_n} = \text{constant} \ , \ n = 1, ..., \tilde{N} \tag{13.4}$$

$$\mathcal{E}_n \geq 0 \ . \tag{13.5}$$

A good low-gap code Γ can be applied to all subchannels with corresponding single-user decoders for that same code's use on the AWGN channel. With such good codes, the vector-coded transmission system will perform as close to capacity as that code would perform as when that same code is applied

[2]If guard bands are used, they are presumed of the same length and to occur at the same time for all users.

to a simple AWGN with no crosstalk nor ISI. The overall sum of all users' number of bits per symbol is then approximated as (and becomes exact if $\Gamma = 1$)[3]

$$b = \sum_{n=1}^{\tilde{N}} \frac{1}{2} \log_2 \left(1 + \frac{\mathcal{E}_n \cdot g_n}{\Gamma} \right) \quad ; \quad \bar{b} = \frac{b}{L_x \cdot (N + \nu)} \tag{13.6}$$

with overall SNR

$$\text{SNR}_{VC} = \frac{2^{2\bar{b}} - 1}{\Gamma} \quad . \tag{13.7}$$

The number of bits is the capacity $\bar{b} = \bar{c}$ when $\Gamma = 1$ (0 dB) and then

$$\text{SNR}_{VC} = 2^{2\bar{c}} - 1 \quad . \tag{13.8}$$

When the input energy distribution is not water-filling, there is a mutual information $\bar{I} < \bar{c}$. With good codes on each of the dimensions, the vector coding system will perform for that non-water-fill energy distribution (input covariance) as

$$\text{SNR}_{VC} = 2^{2\bar{I}} - 1 \quad , \tag{13.9}$$

which is the highest level of performance for this particular choice of Gaussian input energy distribution. The energy constraint will sometimes be denoted by a diagonal matrix

$$\mathcal{E}_{vec} = \text{diag}\,\{\boldsymbol{E}\} \quad . \tag{13.10}$$

Any GDFE with the same input energy distribution and independent input dimensions, or more precisely the same $R_{\boldsymbol{xx}} = M \text{diag}(\boldsymbol{E}) M^* = G_v S_v G_v^*$, will perform at the same level given by (13.9). That mutual information (from Chapter 5) is

$$\bar{I} = \frac{1}{2L_x(N+\nu)} \log_2 \frac{|R_{\boldsymbol{yy}}|}{|R_{\boldsymbol{nn}}|} = \frac{1}{2L_x(N+\nu)} \log_2 \frac{|HR_{\boldsymbol{xx}}H^* + R_{\boldsymbol{nn}}|}{|R_{\boldsymbol{nn}}|} \quad . \tag{13.11}$$

An interesting almost-MAC like channel occurs when inputs can be coordinated but multiple energy constraints exist instead of a single sum-energy constraint. Systems with integer or quantized information units (like integer bit restrictions) can use Levin-Campello (LC) loading algorithms as in Chapter 4. Such LC algorithms readily incorporate any type of power-spectral-density, peak, or sum-energy constraints expressed by ($m_{u,n}$ can be viewed as the gain to the n^{th} dimension by the u^{th} user)

$$\sum_{n=1}^{L_x(N+\nu)U} |m_{u,n}|^2 \cdot e_n \leq \mathcal{E}_u \quad , \tag{13.12}$$

where e_n is an individual dimensional energy constraint (using lower case e_n to distinguish energy for dimension n from energy per user). The weights $m_{u,n}$ could be viewed as the elements of the u^{th} column of the matrix M in the single-user-with-dimensional-energy-constraints problem, but need not be so constrained and could be any set of weights in general. Such constraints in the multi-user channel might correspond to individual antenna-driver (or line-driver) maximum average energy, or even dimensional-peak-energy, constraints. LC algorithms simply need evaluate the constraints in (13.12) after each bit has been assigned the next least energy position. The incremental energy tables of Chapter 4 for each tone are then updated to reflect an infinite cost in the position of any subsequent bit allocation that would cause violation of the constraints. Such multi-energy-constraint channels may have practical interest and fall somewhere in between a multiple-input-multiple-output single-user channel and a **MAC**. This approach assumes, however, that the SVD remains optimum for vector coding, which may only be approximately true. The loss of optimality of a single SVD is more pronounced in the case of the full **MAC**.

[3]Tacitly assumes real dimensions, there is no factor of 1/2 if H is a complex matrix.

For the **MAC**, the inputs are more limited by the lack of coordination between users in addition to having individual-user energy constraints. Thus, the water-filling solution of Equation (13.5) may be impossible to generate within the uncoordinated-input constraint of the **MAC** because of the structure of the M matrix. The equivalent of the data rate for the single-user channel is the sum of all users' data rates. Thus, for whatever choice of energy distribution or input covariances $\{R_{\boldsymbol{xx}}(u)\}_{u=1,\ldots,U}$ by the users, the best possible sum-of-data-rates performance is also given by (13.9). Again, the performance for this choice of input autocorrelation $R_{\boldsymbol{xx}}$ (understood to be a more limited set in **MAC**'s than in single-user channels and block diagonal) has sum-rate performance as in (13.9) when a GDFE receiver is used for these independent **MAC** inputs[4].

The single-user total-energy constraint $\mathcal{E}_{\boldsymbol{x}}$ now for the **MAC** becomes a vector of constraints, as in (13.10). The sum-rate formulation of capacity is convenient and restated here as

$$c(\boldsymbol{b}) = \bigcup_{R_{\boldsymbol{xx}}(u)}^{conv} \bigcup_{\boldsymbol{u}\subset\boldsymbol{U}}^{conv} \left\{ \boldsymbol{b} \mid \sum_{u\in\boldsymbol{u}} b_u \le I(\boldsymbol{x}_u; \boldsymbol{y}/\boldsymbol{x}_u \in \boldsymbol{U}\setminus\boldsymbol{u}) \right\} \tag{13.13}$$

$$= \bigcup_{R_{\boldsymbol{xx}}(u)}^{conv} \bigcup_{\boldsymbol{u}\subset\boldsymbol{U}}^{conv} \left\{ \boldsymbol{b} \mid \sum_{u\in\boldsymbol{u}} b_u \le \frac{1}{2}\log_2 \frac{|\, H\cdot R_{\boldsymbol{xx}}\cdot H^* + R_{\boldsymbol{nn}}\,|}{|\,\sum_{u\in\boldsymbol{U}\setminus\boldsymbol{u}} H_u\cdot R_{\boldsymbol{xx}}(u)\cdot H_u^* + R_{\boldsymbol{nn}}\,|} \right\} \quad . \tag{13.14}$$

The **MAC** input autocorrelation matrix is restricted to be a block diagonal matrix

$$R_{\boldsymbol{xx}} = \begin{bmatrix} R_{\boldsymbol{xx}}(U) & 0 & \ldots & 0 \\ 0 & R_{\boldsymbol{xx}}(U-1) & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & R_{\boldsymbol{xx}}(1) \end{bmatrix} \quad . \tag{13.15}$$

The individual energy constraints are imposed by

$$\mathrm{trace}\{R_{\boldsymbol{xx}}(u)\} \le \mathcal{E}_u \quad . \tag{13.16}$$

13.1.2 The SWF rate-sum criterion for the Gaussian Vector MAC

Rate-sum or mutual-information maximization for the **MAC** fortunately follows the SWF principle of Section 12.2. In the scalar MAC case, any user's water filling treats all the others as noise, presuming all are using single-user-AWGN capacity-achieving codes. The Gaussian vector MAC follows a generalized version of this same basic principle.

The maximization of mutual information in (13.11) reduces to

$$\max_{\{R_{\boldsymbol{xx}}(u)\}_{u=1,\ldots,U}} |H\cdot R_{\boldsymbol{xx}}\cdot H^* + R_{\boldsymbol{nn}}| \tag{13.17}$$

where $R_{\boldsymbol{xx}}(u)$ is the autocorrelation for user u and the inputs are independent as in (13.15). By rewriting the maximization in the form

$$\max_{\{R_{\boldsymbol{xx}}(u)\}} |H\cdot R_{\boldsymbol{xx}}(u)\cdot H^* + \underbrace{\sum_{i\ne u} H_i\cdot R_{\boldsymbol{xx}}(i)\cdot H_i^* + R_{\boldsymbol{nn}}}_{R_{noise}(u)}| \tag{13.18}$$

the problem separates into U optimizations where each user views all the rest as "noise," as indicated by $R_{noise}(u)$ in (13.18), specifically

$$R_{noise} \triangleq \sum_{i\ne u} H_i\cdot R_{\boldsymbol{xx}}(i)\cdot H_i^* + R_{\boldsymbol{nn}} \quad . \tag{13.19}$$

Each of $R_{\boldsymbol{xx}}(u)$ is then determined by the usual water-filling on the vector-coded channel $R_{noise}^{-1/2}(u)\cdot H_u$. The overall rate-maximization problem is convex in the U autocorrelation matrices and thus has a solution that can be found by various "descent" or gradient methods. One such method, "iterative water-filling," has a strong intuitive appeal and appears in Section 13.2. Then, formally:

[4] Independent inputs allow GDFE performance to be canonical.

Theorem 13.1.1 (Optimality of SWF for the MA channel) *The autocorrelation matrices $R^o_{\boldsymbol{xx}}(u)$ that are simultaneously water-filling, that is each $R^o_{\boldsymbol{xx}}(u)$ is the solution to a single-user water-filling problem while all other $R^{(o)}_{\boldsymbol{xx}}(i \neq u)$ are treated as noise, maximize the rate sum for the Gaussian MA channel.*

proof: See the preceding paragraph and note that the optimization problem is separable for each of the $R_{\boldsymbol{xx}}(u)$ as a well-known water-filling problem. **QED.**

The exact determination of the water-filling energies can first form the singular-value-decomposition of the u^{th} user's noise-whitened channel equivalent

$$R^{-1/2}_{noise}(u) \cdot H_u = F_u \cdot \Lambda_u \cdot M^*_u \quad . \tag{13.20}$$

Then the energies are assigned according to (where $g_{u,n} = \lambda^2_{u,n}$)

$$\mathcal{E}_{u,n} + \frac{1}{g_{u,n}} = \text{constant } \forall \; u \; , \tag{13.21}$$

such that

$$\sum_{n=1}^{L_x(N+\nu)} \mathcal{E}_{u,n} = \mathcal{E}_u \quad , \tag{13.22}$$

and

$$\mathcal{E}_{u,n} \geq 0 \quad . \tag{13.23}$$

The u^{th} user's input is constructed as

$$\boldsymbol{x}_u = M_u \cdot \boldsymbol{X}_u \quad , \tag{13.24}$$

where the elements of $\boldsymbol{X}_u$ are independent and each has energy $\mathcal{E}_{u,n}$.

Implications of the Chain Rule for various decoding orders

Successive decoding (or the GDFE) follows the Chain Rule as in Chapter 12, where

$$I(\boldsymbol{x};\boldsymbol{y}) = I(\boldsymbol{x}_1;\boldsymbol{y}) + I(\boldsymbol{x}_2;\boldsymbol{y}/\boldsymbol{x}_1) + ... + I(\boldsymbol{x}_U;\boldsymbol{y}/[\boldsymbol{x}_1 \; \boldsymbol{x}_2 \; ... \; \boldsymbol{x}_{U-1}]) \tag{13.25}$$

for any of the $U!$ possible order(s). The given inputs in each term of the chain rule are those that have already been decoded by the receiver. An alternate form of the rate sum is[5]

$$\begin{aligned}
I(\boldsymbol{x};\boldsymbol{y}) &= \frac{1}{2}\log_2\left(\frac{|\sum_{u=1}^{U} H_u \cdot R_{\boldsymbol{xx}}(u) \cdot H^*_u + R_{\boldsymbol{nn}}|}{|\sum_{u=2}^{U} H_u \cdot R_{\boldsymbol{xx}}(u) \cdot H^*_u + R_{\boldsymbol{nn}}|}\right) \quad (b_1) \\
&+ \frac{1}{2}\log_2\left(\frac{|\sum_{u=2}^{U} H_u \cdot R_{\boldsymbol{xx}}(u) \cdot H^*_u + R_{\boldsymbol{nn}}|}{|\sum_{u=3}^{U} H_u \cdot R_{\boldsymbol{xx}}(u) \cdot H^*_u + R_{\boldsymbol{nn}}|}\right) \quad (b_2) \\
&+ \\
&\vdots \\
&+ \frac{1}{2}\log_2\left(\frac{|H_U \cdot R_{\boldsymbol{xx}}(U) \cdot H^*_U + R_{\boldsymbol{nn}}|}{|R_{\boldsymbol{nn}}|}\right) \quad . \; (b_U)
\end{aligned} \tag{13.26}$$

These equations could be written with the use of the noise-equivalent channel $\bar{H}_u \triangleq R^{-1/2}_{\boldsymbol{nn}} \cdot H_u$ as

$$\begin{aligned}
I(\boldsymbol{x};\boldsymbol{y}) &= \frac{1}{2}\log_2\left(\frac{|\sum_{u=1}^{U} \bar{H}_u R_{\boldsymbol{xx}}(u) \bar{H}^*_u + I|}{|\sum_{u=2}^{U} \bar{H}_u R_{\boldsymbol{xx}}(u) \bar{H}^*_u + I|}\right) \quad (b_1) \\
&+ \frac{1}{2}\log_2\left(\frac{|\sum_{u=2}^{U} \bar{H}_u R_{\boldsymbol{xx}}(u) \bar{H}^*_u + I|}{|\sum_{u=3}^{U} \bar{H}_u R_{\boldsymbol{xx}}(u) \bar{H}^*_u + I|}\right) \quad (b_2)
\end{aligned}$$

[5] Again, no factor of 1/2 appears if the channel is complex.

$$
\begin{aligned}
&+ \\
&\vdots \\
&+ \quad \frac{1}{2}\log_2\left(|\bar{H}_U R_{\boldsymbol{xx}}(U)\bar{H}_U^* + I|\right) \quad .\,(b_U)
\end{aligned}
\tag{13.27}
$$

or with the order-dependent equivalent noise-normalized channels defined as:

$$
\tilde{\bar{H}}_u \triangleq \tilde{R}_{noise}^{-1/2}(u) \cdot H_u \tag{13.28}
$$

with

$$
\tilde{R}_{noise}(u) \triangleq \sum_{i=u+1}^{U} H_i R_{\boldsymbol{xx}} H_i^* + R_{\boldsymbol{nn}} \quad , \tag{13.29}
$$

and thus

$$
\begin{aligned}
I(\boldsymbol{x};\boldsymbol{y}) \quad &= \quad \frac{1}{2}\log_2\left(|\tilde{\bar{H}}_1 R_{\boldsymbol{xx}}(1)\tilde{\bar{H}}_1^* + I|\right) \quad (b_1) \\
&+ \quad \frac{1}{2}\log_2\left(|\tilde{\bar{H}}_2 R_{\boldsymbol{xx}}(2)\tilde{\bar{H}}_2^* + I|\right) \quad (b_2) \\
&+ \\
&\vdots \\
&+ \quad \frac{1}{2}\log_2\left(|\tilde{\bar{H}}_U R_{\boldsymbol{xx}}(U)\tilde{\bar{H}}_U^* + I|\right) \quad .\,(b_U)
\end{aligned}
\tag{13.30}
$$

The chain-rule decomposition explicitly shows the "other users later in order" as noise $\tilde{R}_{noise}$. A GDFE would be equivalent, but as in Section 12.2, the other-user noise would be tacitly but automatically, included in the minimum-mean-square error quantities.

There are $U!$ possible decoding orders in (13.26) and (13.30) that all provide the same rate sum, but usually correspond to different rate tuples $\boldsymbol{b}$. While each term in (13.26) appears to view other user's signals as noise, this noise only includes others that are not previously decoded in the successive decoding, or equivalently excludes earlier users in the selected order. There is a difference between the R_{noise} (no tilde) that helps determine SWF and the $\tilde{R}_{noise}$ that can be used to determine the individual user data rates after a SWF has been found. The rate sum that would be produced in (13.27) above would not be correct if R_{noise} were used and more specifically the rates of the users are NOT computed with ALL the other users as noise except for the first user rate in any chosen order.

An interesting special case occurs when $L_x = 1$ and $L_y = U$, as a direct map to Chapter 5's GDFE. The individual terms in the chain-rule sum are the bits/dimension to be used in encoding for each user and add to the rate sum of $I(\boldsymbol{x};\boldsymbol{y})$ in a GDFE that corresponds to a $U \times U$ square non-singular channel. That is, the GDFE is used as the receiver and each user corresponds to its own dimension. There may be up to $U!$ distinct SWF solutions that provide the same rate sum for the GDFE (meaning as is well known from Chapter 5 that the ordering or indexing of dimensions in the GDFE does not change the single-user data rate). Each order corresponds to a $\boldsymbol{b}$ that is a vertex for the corresponding $\{R^o_{\boldsymbol{xx}}(u)\}$. It is possible in the general **MAC** to have $[L_x(N+\nu)U]!$ different orderings of user dimensions, all providing the same rate sum. However, this text considers only $U!$ orders for reasons that will eventually become apparent in Section 13.4.

The diagram in Figure 13.2 applies with the coefficients g_u determined by a GDFE Cholesky factorization. The GDFE allows, in the absence of error propagation, symbol-by-symbol decoding without loss in SNR, although with Gaussian signals the decoder itself would theoretically span all time to decode the signal before it could be subtracted from the next user in succession. However, that subtraction will use the "white" independent part of the input rather than a correlated part (so a $\boldsymbol{v}_u$ rather than an $\boldsymbol{x}_u$ in the terminology of Chapter 5). The usual approach to successive decoding requires $\boldsymbol{x}_u$, which may not be white and indeed would add complexity to the decoders used. With the GDFE, the decoders are those of only the applied Gaussian codes and need know nothing of the channel or "undecoded" users "noise" correlation. When $\rho(H) < U$, then some users will share at least one dimension, and a MMSE-DFE with all uncanceled users viewed as Gaussian noises is used to estimate each user. This

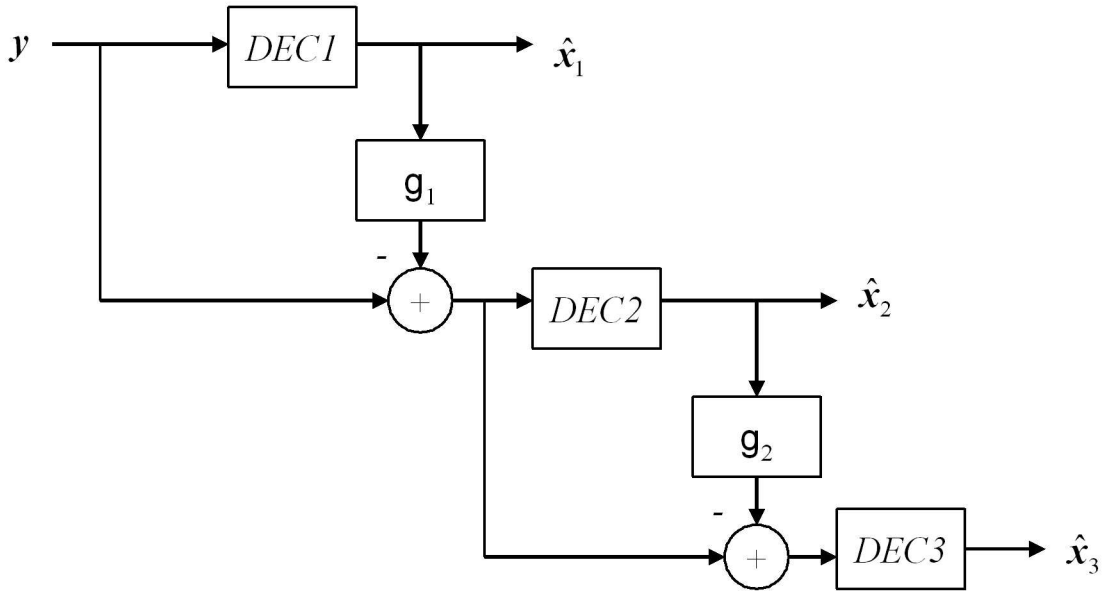

Figure 13.2: GDFE Successive decoding for 3 users.

formulation arises directly and naturally from the GDFE without need for bookkeeping after an order is selected.

There is one special GDFE case where all orders have the same rate tuple. In this case, the users are separated orthogonally from one another – in particular, this corresponds to the one FDM solution when $L_x = L_y = 1$ in Chapter 12 as $N \to \infty$. When L_x or L_y exceeds one, such an orthogonal solution need not necessarily exist as was first shown by former EE479 student W. Rhee (even with $N \to \infty$).

Alternate view of the existence of the SWF point

This subsection further investigates satisfaction of the SWF criterion, leading towards the iterative-water-filling algorithm of Section 13.2.

First, the convergence of a sequence of water-filling steps where each treats all other users as noise is established heuristically.[6] Figure 13.3 illustrates the basic rate region for two users. The 2 users $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ contribute to the common MA output $\boldsymbol{y}$. Then, by the chain rule,

$$I(\boldsymbol{x};\boldsymbol{y}) = I(\boldsymbol{x}_1;\boldsymbol{y}) + I(\boldsymbol{x}_2;\boldsymbol{y}/\boldsymbol{x}_1) = I(\boldsymbol{x}_2;\boldsymbol{y}) + I(\boldsymbol{x}_1;\boldsymbol{y}/\boldsymbol{x}_2) \quad , \tag{13.31}$$

and there are two ways to achieve the same rate sum $I(\boldsymbol{x};\boldsymbol{y})$, that is $U! = 2$ orders. Since either order can be used, the receiver can decode $\boldsymbol{x}_1$ first, and water-fill for user x_1 first with user x_2 as noise. Reversing the order of decoding maintains the rate sum, but sends the implementation to point B on the same (red) pentagon. But, with the opposite order and maintaining user 1's spectrum, user 2 now water fills. Since user 1 is decoded first and does not change spectrum, user 1's rate is maintained. But user 2 must increase its rate (because of water-filling) and so it moves up on the blue pentagon. Again maintaining the rate sum, the order can again be reversed so that the upper corner point on the blue pentagon is now achieved. Since user 2 now maintains the same spectra, user 2's rate does not change. However, user 1's rate increases to point c if another iteration of water filling is used. Clearly the pentagons must keep moving up and to the right unless the rate sum line is SWF everywhere. That SWF everywhere could correspond to a single point (shown in purple), or in cases where multiple SWF may exist a 45^o

[6] Since the SWF condition corresponds to a convex optimization, and descent algorithm can be used and will thus converge, but the objective here is a more heuristic explanation.

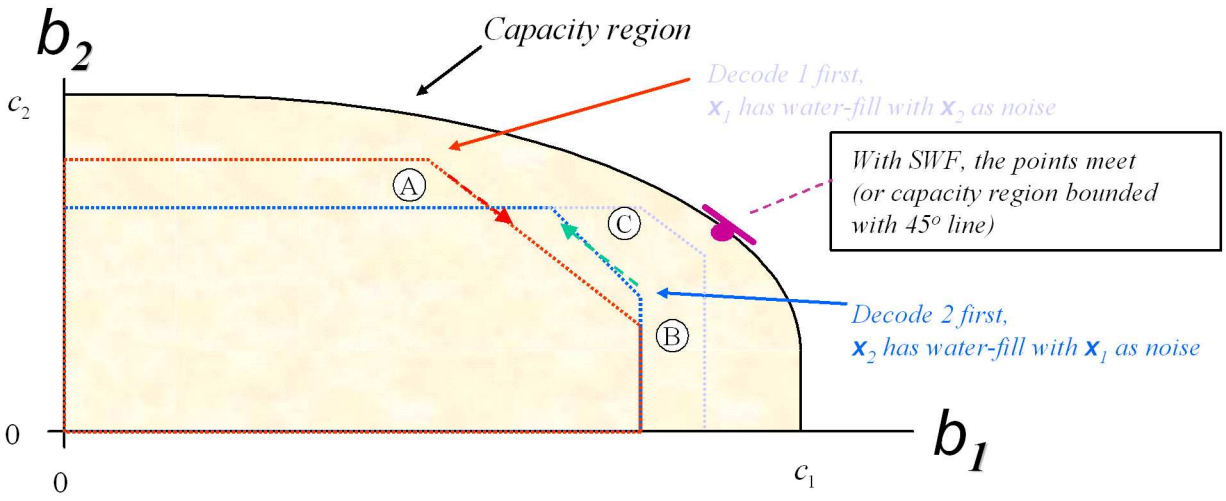

Figure 13.3: Illustration that SWF solution achieves MA channel maximum rate sum.

line that bounds the capacity region. Thus, the result is established for $U = 2$. By setting a single user $\boldsymbol{w}$ to correspond to the rate sum $\bar{b}_1 + \bar{b}_2$ and introducing a 3rd user $\boldsymbol{x}_3$, the concept can be repeated with $\boldsymbol{w}$ replacing $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$ replacing $\boldsymbol{x}_2$. Thus by induction, the iteration of water-filling must converge to a rate-sum-maximizing point and thus SWF.

In general, the receiver implementation will be successive decoding (or a GDFE), and there may be many orderings that achieve the maximum rate sum if the rate sum boundary is a plane or hyperplane. These orderings may all have different constituent user rates. In practical situations where the system is not undersampled so that no individual water-fill band is the entire frequency band of the **MAC**, then usually only one rate-sum-maximizing point exists and all orders would produce this single point that necessarily then has the same constituent rates for all users also. For $L_x = L_y = 1$, this single point will be frequency-division multiplexed as $N \to \infty$ for a linear-time-invariant channel with stationary Gaussian noise. However, for finite N, general block channels, and/or $L_y \geq 1$, the single point with different orders need not necessarily correspond to an orthogonal use of available dimensions.

EXAMPLE 13.1.1 ((.8,.6) AWGN revisited) The capacity region for an AWGN MA channel with $L = N = 1$ and $U = 2$ with $P_1 = .8$, $P_2 = .6$ and $\sigma^2 = .0001$ is shown in Figure 13.4. This channel was studied in Example 12.2.1 previously in Chapter 12. This region is a pentagon. Since the two user's channels are flat, flat energy use on both simultaneously satisfies the SWF criterion trivially. Both corner points are rate-sum maxima for the MA channel. The value summing each rate when the other is viewed as noise is .74+.32=1.06 and is not equal to the maximum rate sum for the MA channel of 6.64, again indicating that individual user rates must be associated with one of the $U! = 2$ orders.

To extend the analysis of this channel, the designer could let $N \to \infty$ while keeping $L_x = L_y = 1$. With an infinite N, there are an infinite number of simultaneous water-filling solutions possible, some of which are shown in Figure 13.5. Solution (a) corresponds to both users flat over the entire band and use of a GDFE at one of the corner points. Solution (b) corresponds to an FDMA solution that is also SWF and has 47.6% of the bandwidth allocated to User 1 and remaining 52.4% of the bandwidth allocated to User 2. Both are flat over their ranges, and clearly both satisfy SWF criteria. The GDFE degenerates into 2 independent decoders (no feedback or successive decoding necessary) in this FDM case. Solution (c) corresponds to a mixture of the two that is part FDMA and part sharing of bandwidth – thus a mixture of solution a and solution b. Because the boundary of this

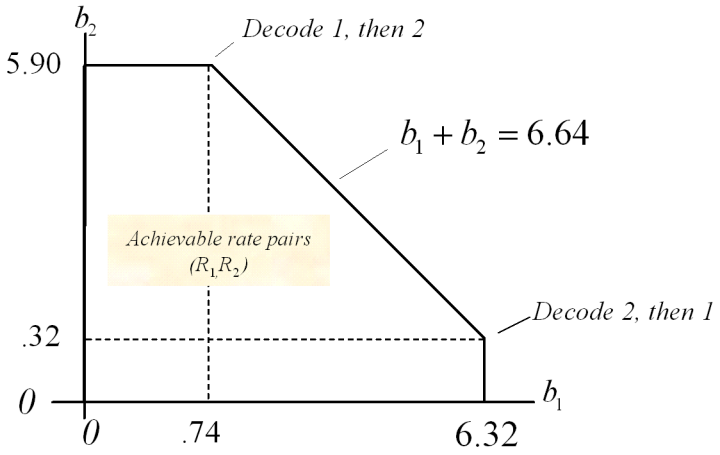

Figure 13.4: Simple MA AWGN Example 13.1.1.

capacity region is a non-zero length flat line with slope -1, several SWF solutions are possible for this example, each corresponding to a point on this line. The representative decoder

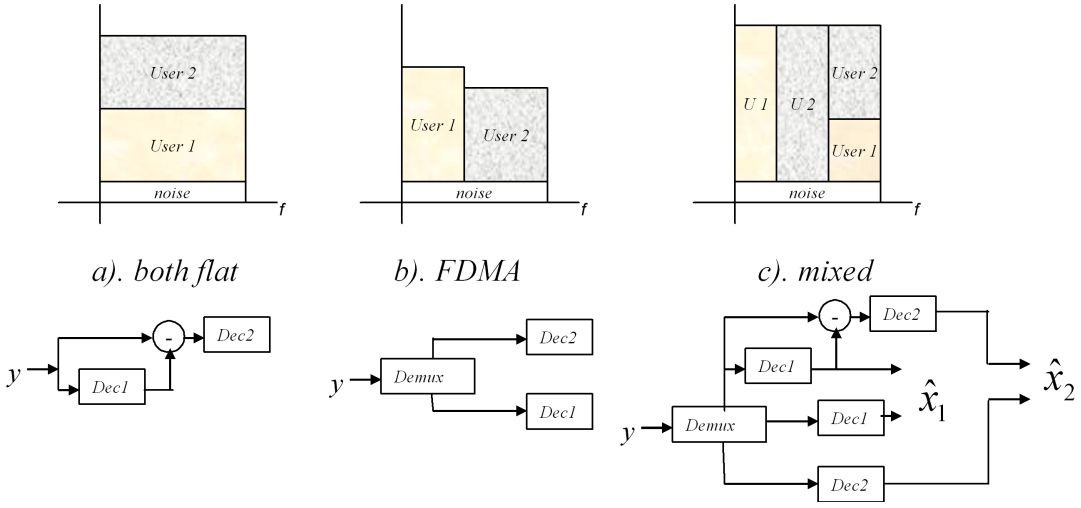

Figure 13.5: Example SWF spectra for Example 13.1.1.

structure is also shown below each choice of spectra.

13.1.3 The use of GDFE's in the MAC

The input autocorrelation matrix for a **MAC** is always block diagonal and can be factored in a block diagonal form in terms of a white input $\boldsymbol{v}$ as

$$R_{\boldsymbol{xx}} = \begin{bmatrix} R_{\boldsymbol{xx}}(U) & 0 & \dots & 0 \\ 0 & R_{\boldsymbol{xx}}(U-1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R_{\boldsymbol{xx}}(1) \end{bmatrix} \tag{13.32}$$

$$= \underbrace{\begin{bmatrix} A_U & 0 & ... & 0 \\ 0 & A_{U-1} & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & ... & 0 & A_1 \end{bmatrix}}_{A} \cdot \tag{13.33}$$

$$\underbrace{\begin{bmatrix} R_{\boldsymbol{vv}}(U) & 0 & ... & 0 \\ 0 & R_{\boldsymbol{vv}}(U-1) & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & ... & 0 & R_{\boldsymbol{vv}}(1) \end{bmatrix}}_{R_{\boldsymbol{vv}}} \cdot \underbrace{\begin{bmatrix} A_U^* & 0 & ... & 0 \\ 0 & A_{U-1}^* & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & ... & 0 & A_1^* \end{bmatrix}}_{A^*} \tag{13.34}$$

where $R_{\boldsymbol{vv}}(u)$ are all diagonal matrices. If singularity occurs in any of the inputs, then the Generalized Cholesky decomposition discussed in Chapter 5 can be applied for the corresponding input factorization if a triangular decomposition is desired. Channels with $\rho(HA) > UN$ need have no dimension sharing of users. The GDFE theory of Chapter 5 applies directly and follows formulation of a canonical forward channel matrix

$$R_f = A^* \cdot H^* \cdot R_{\boldsymbol{nn}}^{-1} \cdot H \cdot A \quad , \tag{13.35}$$

from which a canonical backward channel matrix is formed and factored

$$R_b^{-1} = R_f + R_{\boldsymbol{vv}}^{-1} = G^* \cdot S_0 \cdot G \quad . \tag{13.36}$$

The GDFE feedback section (or successive decoding) then is G with bias removal to a $G_U = I + [[\boldsymbol{SNR}_u + I] \cdot [\boldsymbol{SNR}_u]^{-1} \cdot [G - I]$, where $\boldsymbol{SNR} = R_{\boldsymbol{vv}} R_{\boldsymbol{ee}}^{-1} - R_{\boldsymbol{vv}} S_0$ and $\boldsymbol{SNR}_u = R_{\boldsymbol{vv}} S_0 - I$. The overall feed-forward processing is

$$W = [\boldsymbol{SNR}_u + I] \cdot [\boldsymbol{SNR}_u]^{-1} \cdot S_0^{-1} \cdot G^{-*} \cdot A^* \cdot H^* \cdot R_{\boldsymbol{nn}}^{-1} \quad . \tag{13.37}$$

When $\rho(HA) < UN$, the GDFE auto includes other-users' noise on all dimensions/decisions for a given order.

13.2 Iterative Water Filling

This section introduces several forms of iterative water-filling. Iterative water-filling was noted by Rhee and Ginis as a mechanization of more general convex-optimization solution to the MAC rate-sum maximization problem. Essentially iterative water-filling has each user compute a water-fill spectra while other users' spectra are presumed held constant. Iterative water-filling proceeds to execute water filling for each user successively, perhaps several cycles for all users. The process will converge eventually to an SWF solution $\{R^o_{\boldsymbol{xx}}\}_{u=1,\ldots,U}$. Iterative water filling is a simple implementation of what would otherwise be a mathematical, but insight lacking, procedure had it not been for their observation. A second benefit beyond simplicity is the recognition that essentially each user can autonomously determine its own input autocorrelation without a need for central control, which may have advantages in practice with time variation or under various coordination limitations.

13.2.1 The IW algorithm and related distributed implementations

The **Rate Adaptive (RA) Iterative Water-filling (IW)** of Figure 13.6 is an iterative algorithm that converges to an SWF-criterion satisfying point for the **MAC**. In Figure 13.6, no order is presumed so all users view all others as noise and no successive decoding is presumed in the calculations. The process may take several iterations of the outer loop, but will converge as is shown shortly. In Figure 13.6, u is a user index, j is an index that counts the number of times that U water-fillings have been performed (one for each user), and j_{max} is a number chosen sufficiently large that the designer thinks the algorithm has converged.

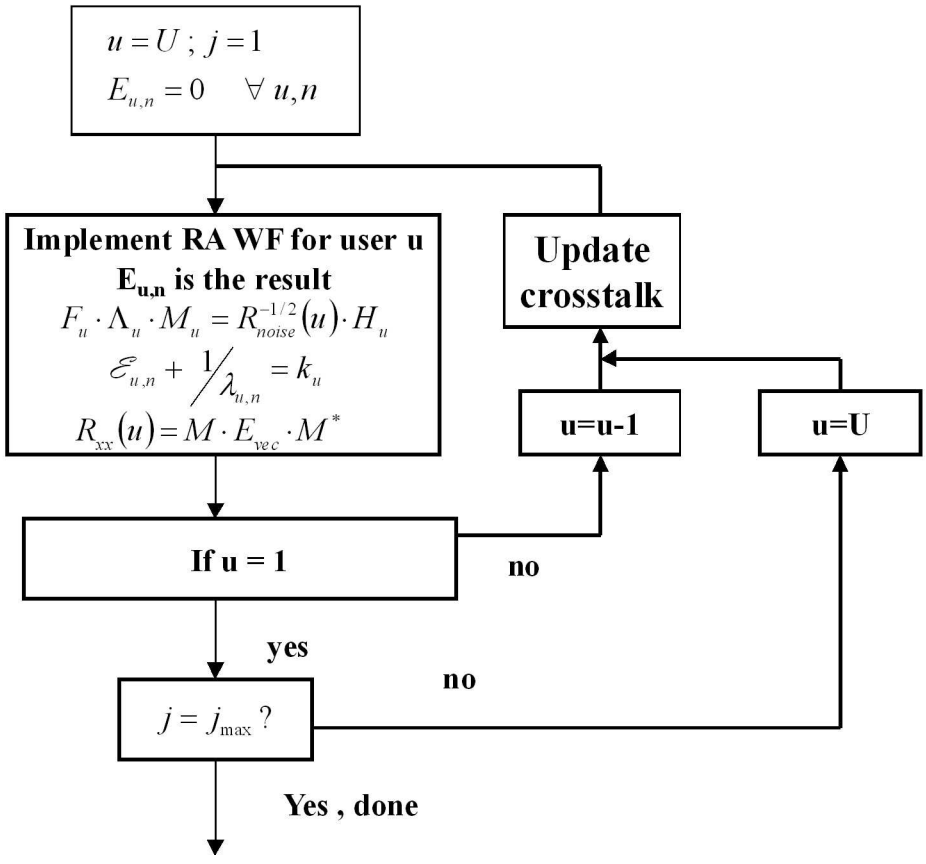

Figure 13.6: Flowchart of (rate adaptive) Iterative Water-filling.

This procedure converges as was illustrated in Section 13.1. Choosing an order allows an implemen-

tation and will then specify each user's constituent rate in the maximum rate sum. The consequent water-filling autocorrelation matrix for any user (denoted $R^o_{\boldsymbol{xx}}(u)$) is derived from viewing a total noise of $R_{noise}(u) = \sum_{i \neq u} H_i \cdot R^o_{\boldsymbol{xx}}(i) \cdot H_i^* + R_{\boldsymbol{nn}}$.

The global convergence for **MAC** is established by again using the convexity of the **MAC** problem:

Theorem 13.2.1 (Convergence of IW algorithm to limiting point) *The IW algorithm always converges to a limiting solution that satisfies the SWF criterion on the* **MAC**. **proof (Yu):** *First the problem is convex, so there is a solution. Any user's data rate will be increased (or not decreased if no further improvement possible) by running water-fill with all other users as noise. In at least one order, this user is first and the maximum rate sum is valid for this and all orders. All other users will not reduce their rates in this order because they all remove this first user in successive decoding and are thus unaffected by this first user's new water-fill spectrum. The same is true for the next iterative water-filling order (with at least one, different, order). Thus the rate-sum is non-decreasing at each step and with this convex system, the procedure eventually will reach the maximum.* **QED.**

EXAMPLE 13.2.1 (simple AWGN channel) This example returns to the ($P_1 = .8,\ P_2 = .6$) MA channel. The first step of IW produces $\mathcal{E}_1 = 1$ and $\bar{b}_1 = 6.32$ bits/dimension. Using this flat spectrum as additional noise for the second step, IW produces $\mathcal{E}_2 = 1$ and $\bar{b}_2 = .32$ bits/dimension.

Starting with user 2 would produce the other corner point of the capacity region.

Distributed Implementations

The IW algorithm in Figure 13.6 does not directly show that coordination is not necessary. Indeed that algorithm is using information about all the channels, input powers, spectral densities, etc to compute other spectra. Such an algorithm could of course be easily programmed in centralized software (Matlab) for the analysis of any particular set of $\tilde{\tilde{H}}_u$ matrices and power constraints. In actual implementation, a master matlab program or analysis may not be possible. Each user's "loading" might independently execute a water-filling algorithm based on the measured $\tilde{\tilde{H}}_u$ for that channel only, and the measured noise autocorrelation $R_{noise}(u)$ (which includes all the other users as if they were noise). Such field implementation produces the same result but requires very little coordination other than users knowing who proceeds them in order. Since the **MAC** allows receiver coordination and the receiver can actually measure all the signals and noises, then this distributed feature is not so interesting, but may be of interest to simplify practical implementation.

Loading algorithms in practice are likely to be continuously implemented in operation as transmissions systems could turn on or off, as well as during initialization. Chapter 4 suggested in practice that methods such as the Levin-Campello or others might be used basically to approximate water-filling with discrete bit loading. The question then arises as to their convergence.

Lemma 13.2.1 (Global Convergence of Levin-Campello Method for MA channel) *The use of sufficiently fast bit-swapping according to the Levin-Campello algorithm of Chapter 4 necessarily converges to within $U-1$ information units of the maximum rate sum for the* **MAC**. *For large N, the overall loss per dimension thus goes to zero.*

proof: *One need only granulate time finely enough to force any update of the LC algorithm to be executed distinctly. Any such step necessarily then increases the rate for that user and since all other users remain at their sum values, the rate increases. Because of the granularity it is possible that up to $U-1$ of the users could be each 1 bit (information unit) away from the best solution. For large $N >> U$, this is negligible. In practice, the LC algorithms cannot instantaneously measure the updated noise and change the bit distribution through a bit swap. However, on average, simple execution will ensure that on average the rate sum is increasing. Thus finite-execution time simply might slow convergence (in several senses) of the algorithm, but not prevent it as long as all channels themselves are otherwise stationary (or vary sufficiently slowly).*

Caution needs to be exercised in interpreting chain-rule concepts when the gap is non-zero, as in the following Example 13.2.2.

EXAMPLE 13.2.2 (non-zero gap) A $U = 2$ **MAC** with $N = 1$ has user SNRs $\text{SNR}_1 = \frac{\bar{\mathcal{E}}_1}{\sigma^2} = 22$ dB and $\text{SNR}_2 = \frac{\bar{\mathcal{E}}_2}{\sigma^2} = 29$ dB. A gap of 9.5 dB for $P_e = 10^{-7}$ is used for both users, $\Gamma_1 = \Gamma_2 = \Gamma = 9.5$ dB. Both users use QAM with symbol rates of 10 MHz and the noise is AWGN.[7] The maximum data rates for the two users are

$$R_{1,last} \leq 10^7 \cdot \log_2\left(1 + \frac{\text{SNR}_1}{\Gamma_1}\right) = 44.2 \text{ Mbps} \tag{13.38}$$

$$R_{2,last} \leq 10^7 \cdot \log_2\left(1 + \frac{\text{SNR}_2}{\Gamma_2}\right) = 64.9 \text{ Mbps} \tag{13.39}$$

and represent corner points for the Γ-dependent rate region that correspond to decoding the other user first. The rates for decoding first are

$$R_{1,first} = 10^7 \cdot \log_2\left(1 + \frac{\text{SNR}_1}{(1 + \text{SNR}_2) \cdot \Gamma_1}\right) = 0.3 \text{ Mbps} \tag{13.40}$$

$$R_{2,first} = 10^7 \cdot \log_2\left(1 + \frac{\text{SNR}_2}{1 + \text{SNR}_1) \cdot \Gamma_2}\right) = 6.4 \text{ Mbps} \tag{13.41}$$

The larger rate sum corresponds to decoding user 1 first and user 2 last and is 64.9+.3 = 65.2 Mbps. Order is important to maximum rate sum when the gap is non-zero. The other order would have produced a rate sum of 44.2+6.4 = 50.6 $<$ 65.2 Mbps. When the gap is zero, either order would produce the rate sum

$$R_{sum,\Gamma=0} = 10^7 \cdot \log_2(1 + \text{SNR}_1 + \text{SNR}_2) = 99 \text{ Mbps} \quad . \tag{13.42}$$

Also

$$R_{max\ sum} = 65.2 < 10^7 \cdot \log_2\left(1 + \frac{\text{SNR}_1 + \text{SNR}_2}{\Gamma}\right) = 67.5 \text{ Mbps} \quad . \tag{13.43}$$

The latter expression on the right in (13.43) is an easy mistake to make if one errors by applying the chain rule with non-zero gaps.

The area of non-zero gaps with iterative water-filling has been addressed by Jagannathan in his Stanford dissertation. Problem 13.10 investigates this area further.

13.2.2 Fixed-Margin (Min Power) Iterative Water-filling

Fixed-margin (FM) water-filling is reviewed in Figure 13.7. The gap is again as in RA water-filling and used for actual systems that cannot operate exactly at capacity, but this section initially assumes that $\Gamma = 0$ dB. FM water-filling is very close to margin-adaptive[8] water-filling in Section 4.3. The only difference is the last step in Figure 4.9 of Chapter 4,

$$\gamma_{max} = \frac{\mathcal{E}_u}{\sum_{n=1}^{N^*} \mathcal{E}_{u,n}} \quad , \tag{13.44}$$

is omitted in FM water-filling. Thus, the power is minimized for the given data rate of $\bar{b}_u$ that is desired (for the given gap). Thus, FM introduces an element of "politeness" to the other users.

FM IW follows Figure 13.6 with RA replaced by an FM box containing the algorithm of Figure 13.7, except that the gains g_i now corresponding to the singular value decomposition of $\tilde{\tilde{H}}_u$ for each u in some given order selected for FM IW. The FM IW always converges for some selected order in one pass through all users. The converged multiuser $R_{\boldsymbol{xx}}(u)$ $u = 1, ..., U$ point obtained may not satisfy the original energy constraints unfortunately. Several orders (or all $U!$) can be attempted. If any, or any time-shared combination thereof, satisfies the energy constraints, a solution has been found. If not, the methods of Sections 13.4 and 13.5 are necessary.

[7] Note - no 1/2 is used in this case because the channels are complex.

[8] Take care that we used the acronym MA to stand for "margin adaptive" in Chapter 4, while it means "multiple access" here so we fully write margin adaptive in Chapters 13-15 where appropriate.

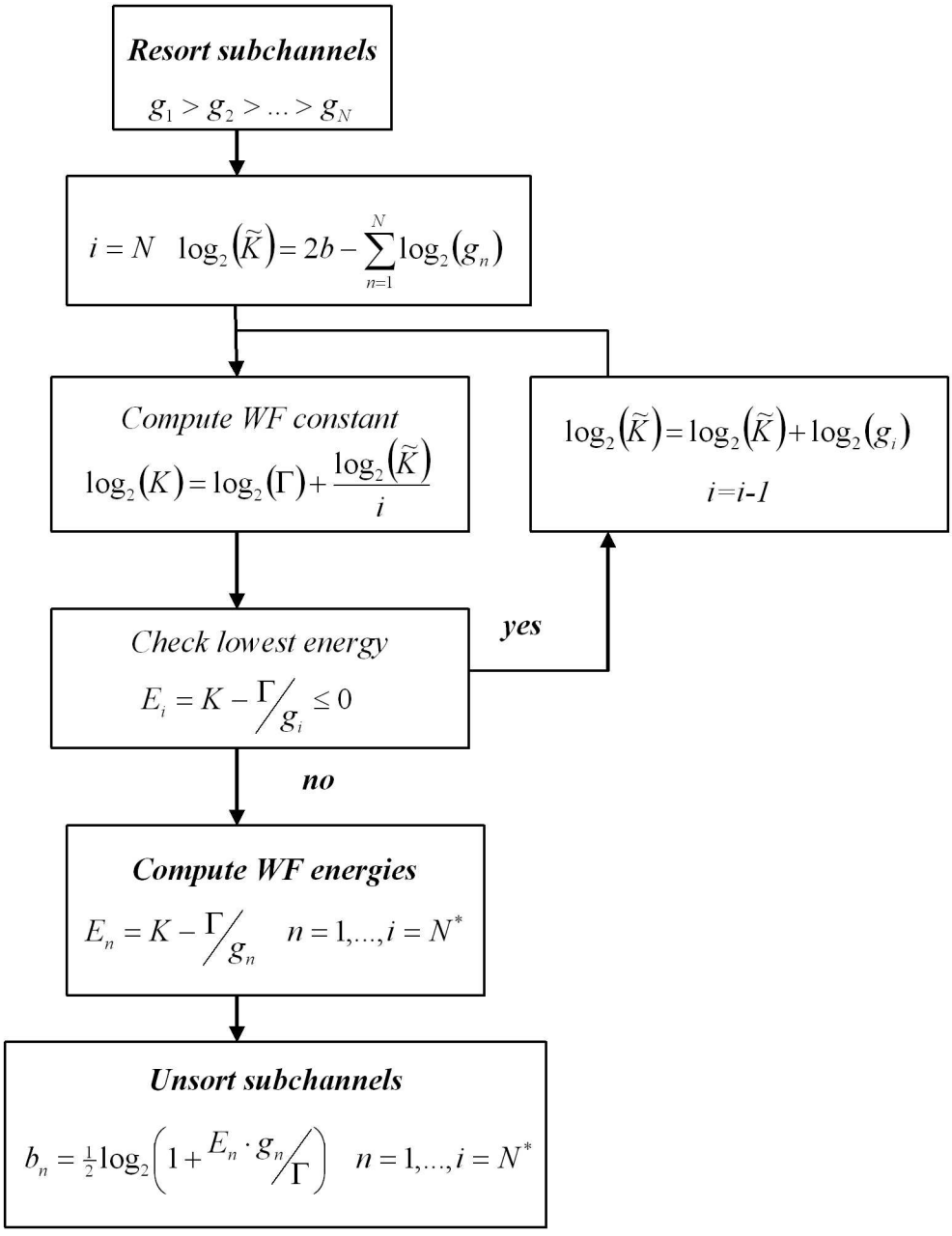

Figure 13.7: Fixed-Margin Water-filling for any user u.

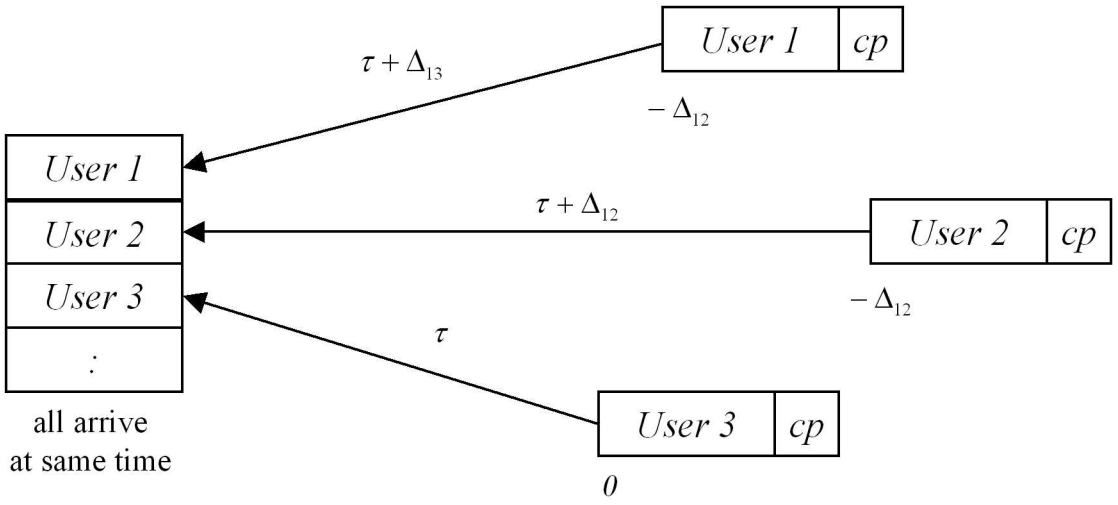

Figure 13.8: Illustration of Vector DMT timing alignment.

13.3 Vector DMT for Multiple Access

Vector DMT systems were first introduced in Chapter 5 for a linear time-invariant channel. The assumption of linear time invariance is continued throughout this section. Figure 13.8 illustrates the synchronization of multiple-access DMT symbols at a common receiver. Each transmitter uses the same size DMT symbol and offsets symbol boundaries so that all users arrive at a common receiver symbol boundary. Such alignment presumes the receiver supplies a common clock through a reverse channel to all transmitters (essentially telling these transmitters to advance or delay their transmit symbol boundary until all are aligned at the receiver). Such alignment can be achieved in up-channel direction (and simultaneously the down-channel direction) as described in Section 4.6 of Chapter 4 on digital duplexing or "zippering." M**MAC** 's need only synchronize in the "up" direction. A bi-directional alignment will be important for systems that use Vector DMT in both directions and thus also have a vector **BC** , as in Chapter 14. There are L_x IFFT's implemented at each of the U users' transmitters. There are L_y FFT's implemented at the common receiver.

Such alignment will, if the common cyclic extension of DMT partitioning is longer than the length of any of the response entries corresponding to each and all of the $\tilde{\bar{H}}_u$ (that is $\nu T' \geq$ length $\left\{\max_{u,i}\left(\tilde{\bar{h}}_{u,i}(t)\right)\right\}$, lead to no intersymbol interference and to crosstalk on any particular tone n that is a function ONLY of other users' signals on that same tone n of other users. Each tone of the L_y receivers' FFT outputs can then be modeled in Figure 13.9 as

$$\underbrace{\boldsymbol{Y}_n}_{L_y \times 1} = \underbrace{H_n}_{L_y \times L_x U} \cdot \underbrace{\boldsymbol{X}_n}_{L_x U \times 1} + \underbrace{\boldsymbol{N}_n}_{L_y \times 1} \quad , \tag{13.45}$$

where

$$H_n = [H_{U,n} \ldots H_{1,n}] \tag{13.46}$$

$$\boldsymbol{X}_n = \begin{bmatrix} \boldsymbol{X}_{U,n} \\ \vdots \\ \boldsymbol{X}_{1,n} \end{bmatrix} \tag{13.47}$$

$$\boldsymbol{Y}_n = \begin{bmatrix} y_{L_y,n} \\ \vdots \\ y_{1,n} \end{bmatrix} \tag{13.48}$$

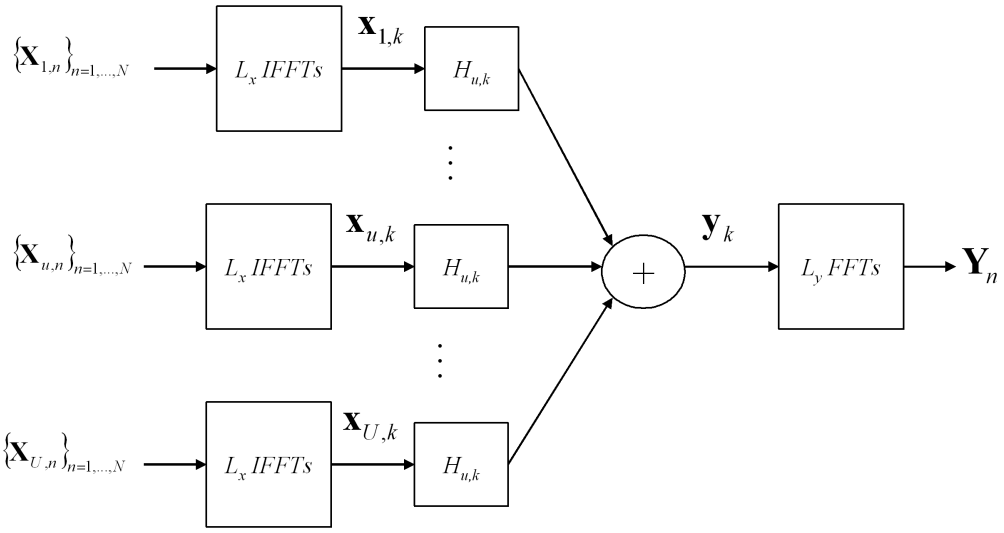

Figure 13.9: Illustration of the Vector DMT system.

$$\boldsymbol{X}_{u,n} = \begin{bmatrix} x_{u,L_x,n} \\ \vdots \\ x_{u,1,n} \end{bmatrix} \quad . \tag{13.49}$$

The $(l_y, l_x)^{th}$ entry of $H_{u,n}$ is the DFT of the response from line/antenna l_x of user u to line/antenna l_y of the common output. The energy constraints become

$$\sum_n \text{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\} \leq \mathcal{E}_u \; \forall \; u = 1, ..., U \quad . \tag{13.50}$$

This tone-indexed model, as illustrated in Figure 13.10, for DMT leads to tremendous computational reduction with respect to the full successive decoding (or GDFE) structure of Section 13.1. Essentially that structure is now applied independently to each tone. Effectively, N small successive-decoding channels of size $L_y \times L_x \cdot U$ replace a giant successive decoding of size $L_y \cdot N \cdot L_x \times N \cdot U$. The GDFE computational advantage when $L_x = 1$ and $L_y = U$ is a complexity of $U \cdot N \cdot log_2(N) + NU^2$ versus the much larger $(N \cdot U)^2$, or if $N = 128$ and $U = 4$, the savings is a factor of about 50 times less computation (262,144 vs 5,632)[9].

For further analysis, given an order of decoding with Vector DMT, the $L_y \times L_x$ matrix

$$\tilde{\bar{H}}_{u,n} \triangleq \tilde{R}_{noise}^{-1/2}(u,n) \cdot H_{u,n} \quad , \tag{13.51}$$

and

$$\tilde{R}_{noise}(u,n) = R_{\boldsymbol{NN}}(n) + \sum_{i=u+1}^{U} H_{i,n} \cdot R_{\boldsymbol{XX}}(i,n) \cdot H_{i,n}^* \quad , \tag{13.52}$$

can be constructed. There are $U!$ possible user orders that could be used in computing such an $\tilde{R}_{noise}(u,n)$. All correspond to valid GDFE's for the tone.

Vector coding code be applied to each of these tones through singular value decomposition:

$$\tilde{\bar{H}}_{u,n} = \tilde{F}_{u,n} \cdot \tilde{\Lambda}_{u,n} \cdot \tilde{M}_{u,n}^* \quad , \tag{13.53}$$

[9] One can only hope that the uninformed who incorrectly claimed less complexity for single-carrier approaches will be staggered by such overwhelming complexity reduction for multi-carrier in the multiple-access case!

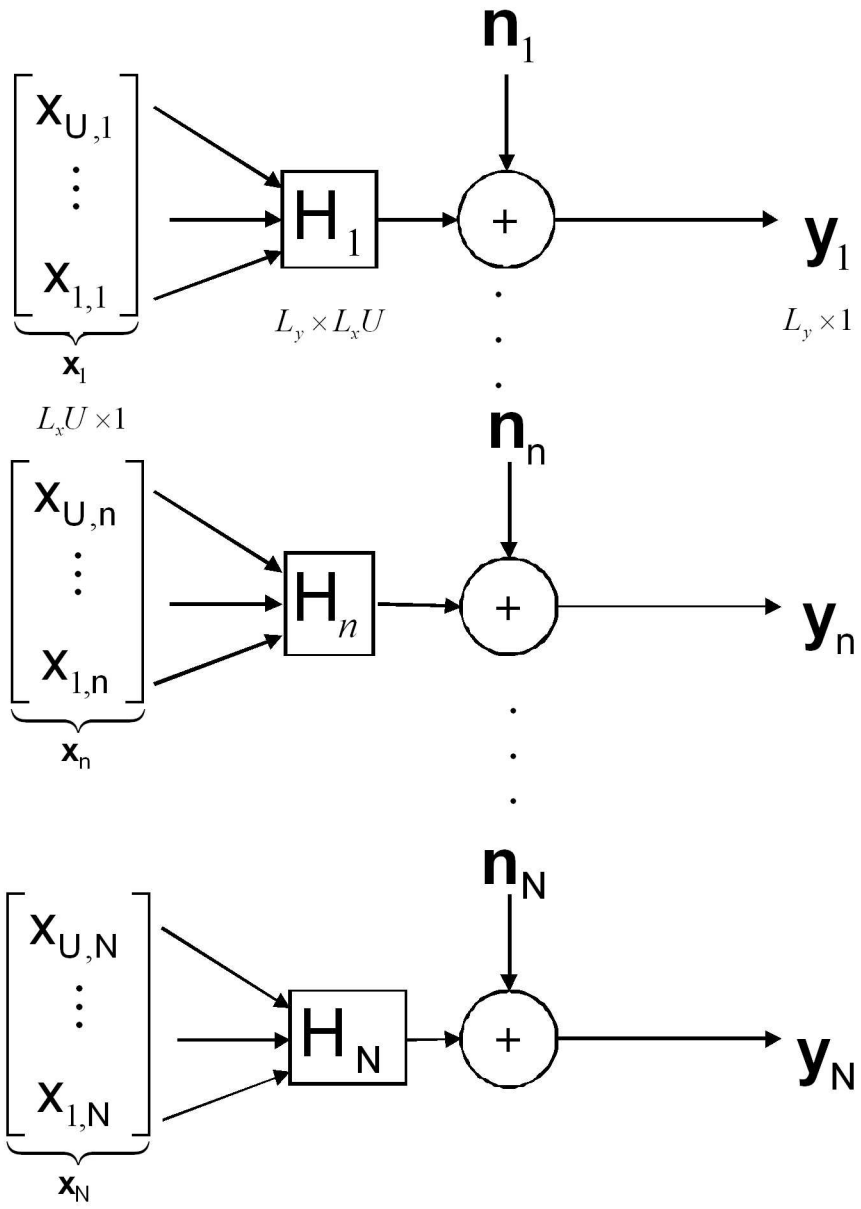

Figure 13.10: Tonal Equivlaent of the Vector DMT system.

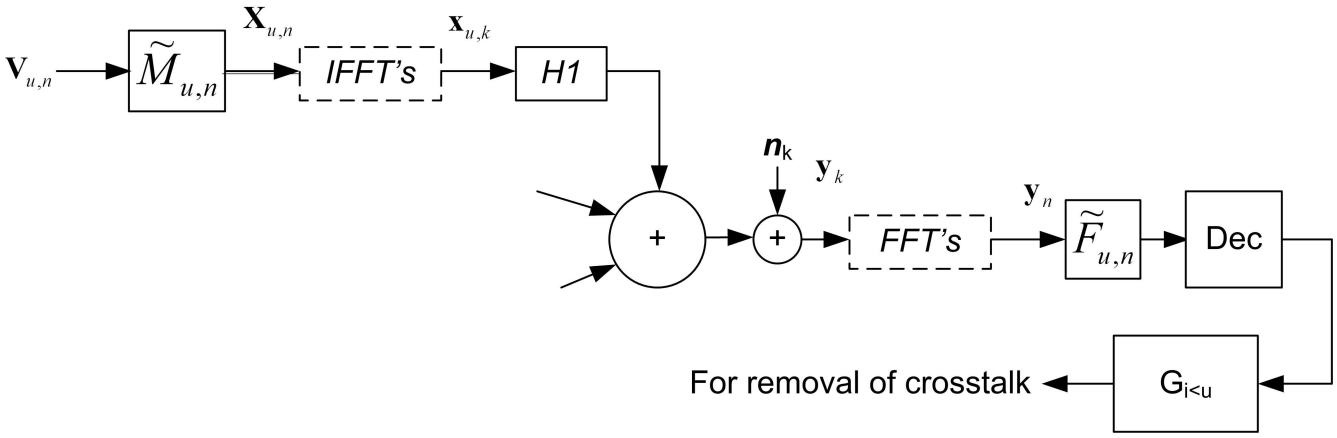

Figure 13.11: Vector-DMT with equivalent noise-whitened appoach.

so that an input is designed where $\tilde{M}_{u,n}$ is a transmit matrix such that

$$R_{\boldsymbol{XX}}(u,n) = \tilde{M}_{u,n} \cdot R_{\boldsymbol{vv}}(u,n) \cdot \tilde{M}^*_{u,n} \tag{13.54}$$

with $R_{\boldsymbol{vv}}(u,n)$ diagonal,

$$\tilde{g}_{u,l,n} \triangleq \tilde{\lambda}^2_{u,l,n} \; \forall \; l = 1, ..., L_x \quad . \tag{13.55}$$

Figure 13.11 illustrates the implementation, which is in effect a special form of both a GDFE and successive decoding. This figure shows one user, which has transmit and receiver vector-coding matrix filters designed with noise whitening based on the equivalent noise $\tilde{R}_{noise}(u)$. Subsection 13.3.2 will use the GDFE directly on each tone to simplify the design.

13.3.1 SWF for the Vectored DMT MAC

Simultaneous water-filling for Vector DMT follows the more general case for the maximum rate sum. Any user u water-fills with $R_{noise}(u,n) = \sum_{i \neq u} H_{i,n} \cdot R_{\boldsymbol{XX}}(i,n) \cdot H^*_{i,n} + R_{\boldsymbol{NN}}$ as noise. The equivalent channels are thus determined by $\bar{H}_{u,n} = R^{-1/2}_{noise}(u,n) \cdot H_{u,n} = F_{u,n} \cdot \Lambda_{u,n} \cdot M^*_{u,n}$. The energy distribution is determined by

$$\mathcal{E}_{u,l,n} + \frac{1}{g_{u,l,n}} = \text{constant}_u \; \forall n, l \quad . \tag{13.56}$$

The input autocorrelation matrix for user u is then formed by

$$R^o_{\boldsymbol{XX}}(u,n) = M_{u,n} \cdot \text{diag} \cdot \{\mathcal{E}_{vec}(u,n)\} \cdot M^*_{u,n} \; \forall \; n \quad . \tag{13.57}$$

Such $R^o_{\boldsymbol{XX}}$ can be determined by the iterative water-filling in Section 13.2. To determine a rate-sum=maximizing bit distribution, each input autocorrelation can be factored so that

$$R^o_{\boldsymbol{XX}}(u,n) = P_{u,n} \cdot \tilde{\mathcal{E}}_{vec}(u,n) \cdot P^*_{u,n} \tag{13.58}$$

where $\tilde{\mathcal{E}}_{vec}$ is diagonal matrix of input energies on each dimension. Then an order is selected and $\tilde{R}_{noise}(u,n) = \left[\sum_{i=u+1}^{U} H_{u,n} \cdot R^o_{\boldsymbol{xx}}(u,n) \cdot H^*_{u,n}\right] + R_{\boldsymbol{NN}}$ formed for this order. Then

$$\tilde{\bar{H}}_{u,n} = \tilde{R}^{-1/2}_{noise}(u,n) \cdot H_{u,n} \cdot M_{u,n} = \tilde{F}_{u,n} \cdot \tilde{\Lambda}_{u,n} \cdot \tilde{M}^*_{u,n} \quad . \tag{13.59}$$

Defining $\tilde{g}_{u,l,n} \triangleq \lambda^2_{u,l,n}$, then the number of bits carried by user u in that chosen order is

$$b_u = \sum_n b_{u,n} = \sum_n \sum_{l=1}^{L_x} \log_2 \left(1 + \tilde{\mathcal{E}}_{u,l,n} \cdot \tilde{g}_{u,l,n}\right) \quad . \tag{13.60}$$

(a factor of 1/2 is introduced if the channel is real baseband, but this happens in DMT only for the DC and Nyquist tones, which are often not used anyway.[10]) This number of bits will be different with order for each user, but the overall rate sum

$$b = \sum_{u=1}^{U} b_u = \sum_{u=1}^{U} \sum_{n} b_{u,n} \tag{13.61}$$

is always the same and is constantly equal to the maximum for all $(U!)^N$ orders. For the case of $L_x > 1$, the existence of an FDM solution is not guaranteed. When $L_x = 1$, there exists a rate-sum vector $\boldsymbol{b}_{max}$ for which all orders provide the same b_u set - an FDM point, as $N \to \infty$.

13.3.2 Tonal GDFE's

While $R^o_{\boldsymbol{XX}}$ may maximize the rate sum, a GDFE exists for any set of input autocorrelation matrices. To implement the Vector-DMT GDFE with known $R_{\boldsymbol{XX}}(n) = A_n R_{\boldsymbol{vv}}(n) A_n^*$ with $R_{\boldsymbol{vv}}(n)$ diagonal, the receiver forms

$$\begin{aligned} \boldsymbol{Z}_n &= \left[A_n^* \cdot H_n^* \cdot R^{-1}_{\boldsymbol{NN}}(n)\right] \cdot \boldsymbol{Y}_n & (13.62)\\ &= A_n^* \cdot H_n^* \cdot R^{-1}_{\boldsymbol{NN}}(n) \cdot H_n \cdot A_n \boldsymbol{V}_n + \boldsymbol{N}'_n & (13.63)\\ &= R_{f,n} \cdot \boldsymbol{V}_n + \boldsymbol{N}'_n \quad , & (13.64) \end{aligned}$$

a forward canonical channel for tone n. The entity $\tilde{R}_{noise}(u)$ does not appear explicitly in the GDFE design, which instead uses directly $R_{f,n}$. The given order of the GDFE, which is implied by the ordering of inputs in $\boldsymbol{V}_n$, does assign later users as noise. Different orders then simply correspond to re-indexing the GDFE dimensions in the channel model. There are N such forward canonical channels that act independently, one for each tone. The canonical backward channel is characterized by the backward matrix with Cholesky factorization

$$R_{b,n}^{-1} = R_{f,n} + R^{-1}_{\boldsymbol{vv}}(n) = G_n^* \cdot S_{0,n} \cdot G_n \quad , \tag{13.65}$$

The upper triangular Cholesky factor G_n determines the feedback section. The diagonal matrix $\boldsymbol{SNR}$ contains the biased SNR's for the estimation of all dimensions of all users and is

$$\boldsymbol{SNR}_n = R_{\boldsymbol{vv}}(n) \cdot S_{0,n} \quad . \tag{13.66}$$

The unbiased feedforward section that processes the channel output vector $\boldsymbol{Y}_n$ is

$$W_n = [\boldsymbol{SNR}_n] \cdot [\boldsymbol{SNR}_n - I]^{-1} \cdot S_{0,n}^{-1} \cdot G_n^{-*} \cdot A_n^* \cdot H_n^* R^{-1}_{\boldsymbol{NN}}(n) \quad . \tag{13.67}$$

The unbiased[11] feedback section is

$$G_n^{unb} = I + \boldsymbol{SNR}_n \left[\boldsymbol{SNR}_n + I\right]^{-1} \left[G_n - I\right] \quad . \tag{13.68}$$

When the channel rank is $N \cdot U \cdot L_x$, then there is no "other user" noise and each user occupies a dimension, directly analogous to Chapter 5's estimation of each dimension. When the rank is less, other users naturally become significant constituents of most dimensions' error signals. Figure 13.12 illustrates the GDFE structure for each tone. Decisions are made on each element of the vector $\boldsymbol{V}_n$ in succession from bottom (user 1, antenna/line 1) to top (user U)according to the users' order. A GDFE structure exits for each and every order.

[10] For a real baseband channel, the number of tones is typically $N/2$ because of conjugate symmetry. We shall tacitly assume the designer knows the correct number of DMT tones, and we often write $\sum_n$ to avoid the notational complication.

[11] A superscript of "unb" is used to denote "unbiased" to avoid confusion with the use of U as a user index.

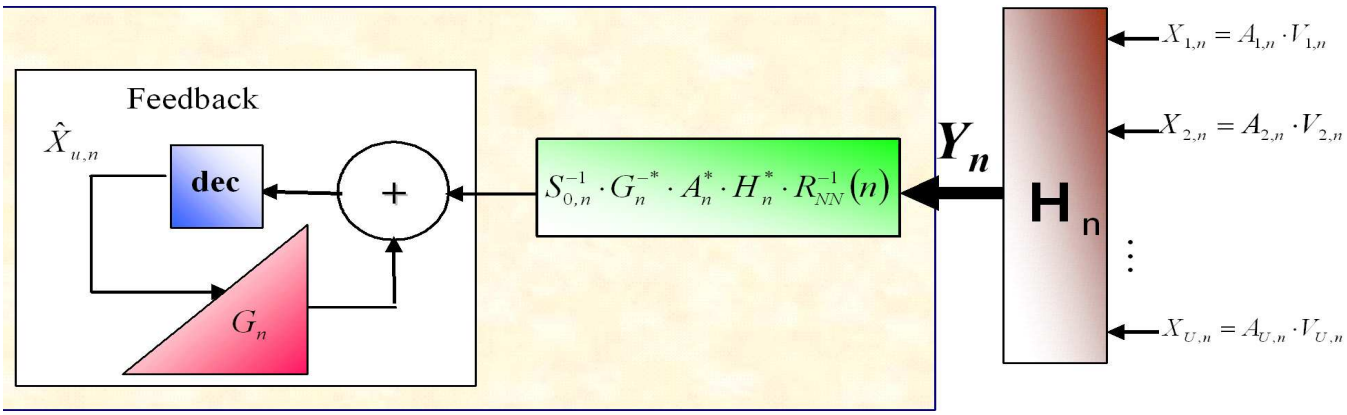

Figure 13.12: Tonal GDFE

Column dominance

A special case of the GDFE occurs when $R_{\boldsymbol{nn}}(n) = I$ and $H_n \cdot A_n$ is column dominant. Column dominance means that the diagonal element of each column is much greater than the other elements in that column, so if $P_n = H_n \cdot A_n$, then

$$|P_n(u,u)| >> |P_n(i \neq u, u)| \; \forall i \neq u \quad . \tag{13.69}$$

In this case, then R_f and thus R_b^{-1} become approximately diagonal matrices, and then $G \rightarrow I$. There is no feedback in this situation, and order is irrelevent in the GDFE – all orders produce the same result. Indeed each user best uses a water-filling energy distribution with respect to the same noise. This situation occurs in some DSL channels when the upstream noise is "white." The "other user" noise is completely eliminated by the action of the feedforward matrix.

The bit rate on tone n for user u's l^{th} input line/antenna is

$$b_{u,l,n} = \log_2 \left(\mathcal{E}_{u,l,n} \cdot S_{0,u,l,n} \right) \quad . \tag{13.70}$$

The bit rate for user u is then

$$b_u = \sum_n \sum_{l=1}^{L_x} b_{u,l,n} \quad . \tag{13.71}$$

The rate sum for tone n is

$$b_n = \sum_{u=1}^{U} b_{u,n} \tag{13.72}$$

and is the rate for tone n's GDFE.

The ZF-GDFE may also be of interest and when H_n has rank $L_x = U$ nonsingular (nor close to singular) because the ZF-GDFE then works at nearly the same performance as the GDFE (MMSE). The procedure for design is for the "Q-R" factorization (G_n is monic upper triangular, Q_n is orthogonal, and D_n is diagonal)

$$R_{\boldsymbol{NN}}^{-1/2} H_n = Q_n \cdot D_n \cdot G_n \tag{13.73}$$

and the feedback section as G_n and the feedforward section as $W_n = D_n^{-1} \cdot Q_n^*$, thus avoiding some of the matrix algebra at essentially no loss in performance when ZF is close to MMSE. In this form, no matched filtering is directly used. Following, this same form and including the noise-whitening, the entire feedforward matrix is $D_n^{-1} \cdot Q_n^* \cdot R_{\boldsymbol{NN}}^{-1/2}$. The unbiased SNRs are computed by

$$\text{SNR}_{u,l,n} = \mathcal{E}_{u,l,n} \cdot [D_{u,l,n}]^2 \quad . \tag{13.74}$$

The complexity of the tonal GDFE is dramatically less with Vector DMT than would be the case with single-carrier type modulation. The tonal DFE complexity in multiply-accummulate operations is

$$\text{Tonal GDFE operations} = N \cdot (LU)^2 + L \cdot N \log_2(N) \tag{13.75}$$

while the complexity of the full GDFE without DMT is

$$\text{Full GDFE operations} = (NLU)^2 \quad . \tag{13.76}$$

For instance, with $N = 1024$, $L = 1$, and $U = 10$, the complexity is roughly 100,000 operations per symbol for vector DMT, or 100 per sample. By contrast, the full GDFE would require approximately 100 million operations per symbol or roughly 100,000 operations per sample, a factor of 1000 times more complexity (some level of zeros might exist in the GDFE matrices, but this is of course channel dependent and unreliable as a complexity-reducing mechanisum). Thus, the complexity advantages of DMT in the single-user case magnify greatly with multiple users.

13.3.3 Spatial Correlation of Noise

Spatial correlation refers to the possibly non-diagonal structure of the matrix $R_{\boldsymbol{NN}}(n)$ on any (or many/all) of the independent tone-indexed matrix channels of the vector-DMT system. The GDFE easily handles such correlation in concept. System designers may often assume such non-crosstalk noise to be "white" or diagonal. For any given set of diagonal values of the noise, a "white" assumption may reduce estimates of system capacity greatly.

It is clear from Equation (13.63) that a singular or near-singular $R_{\boldsymbol{NN}}(n)$ with consequent tiny or zero determinant in the denominator could produce a very high data rate sum. For vector-MA systems, such near-singular spatial noise correlation may be common: Figure 13.13 illustrates the basic situation where a common noise source impinges on each of two channels (lines or antennae). There is a high correlation of the noise on the two channels because the source is the same. If this were the only noise, the capacity would be infinite. One receiver could measure the noise and then subtract the noise from the other channel output if receiver coordination is allowed as in the **MAC** , leaving noise-free operation with infinite capacity. Figure 13.13 also illustrates the basic channel from noise to receivers (itself a single-user channel with one input and two outputs).

If the noise in the upper portion of Figure 13.13 is white on the input, then the autocorrelation matrix is

$$R_{\boldsymbol{NN}}(n) = \begin{bmatrix} h_2^2 & h_2 h_1 \\ h_1 h_2 & h_1^2 \end{bmatrix} \quad , \tag{13.77}$$

which is singular, leading to infinite capacity. Suppose instead the situation in the lower portion of Figure 13.13 occurs with two independent noise sources. Then, the noise correlation becomes

$$R_{\boldsymbol{NN}}(n) = \begin{bmatrix} \left(h_{2a}^2 + h_{2b}^2\right) \cdot \sigma_a^2 & h_{2a} \cdot h_{1a}\sigma_a^2 + h_{2b} \cdot h_{1b}\sigma_b^2 \\ h_{2a} \cdot h_{1a}\sigma_a^2 + h_{2b} \cdot h_{1b}\sigma_b^2 & \left(h_{1a}^2 + h_{1b}^2\right)\sigma_b^2 \end{bmatrix} \quad , \tag{13.78}$$

which is generally not singular. Indeed, as more and more noise sources contribute, on average, the $R_{\boldsymbol{NN}}$ matrix will become less singular (and capacity decreases). However, if for the case of two noise sources, $L_y = 3$, then $R_{\boldsymbol{NN}}(n)$ is singular again and the capacity is large/infinite. In general, if the number of independent noise sources is less than L_y, then the capacity is infinite. For practical systems, each receiver front-end will have some small independent noise so infinite capacity never occurs – however data rates may be very high if the number of external noise sources is less than L_y (presuming receiver front-end noise is small, and will essentially determine capacity). The spatial-singularity effect cannot occur if $L_y = 1$. Thus vector multiple-access systems may have enormous capacity in both wireless and wireline applications. In wireless systems, if the number of receive antennae exceeds the number of "other users" (i.e., from adjacent cells) then if the system can be optimized by vector-DMT, the **Uplink** capacity will consequently be enormous. In wireline DSL systems, this result means that upstream capacity can be enormous even in the presence of uncoordinated crosstalk, as long as the number of lines coordinated exceeds the number of significantly contributing crosstalkers outside the **MAC**. Such results can also

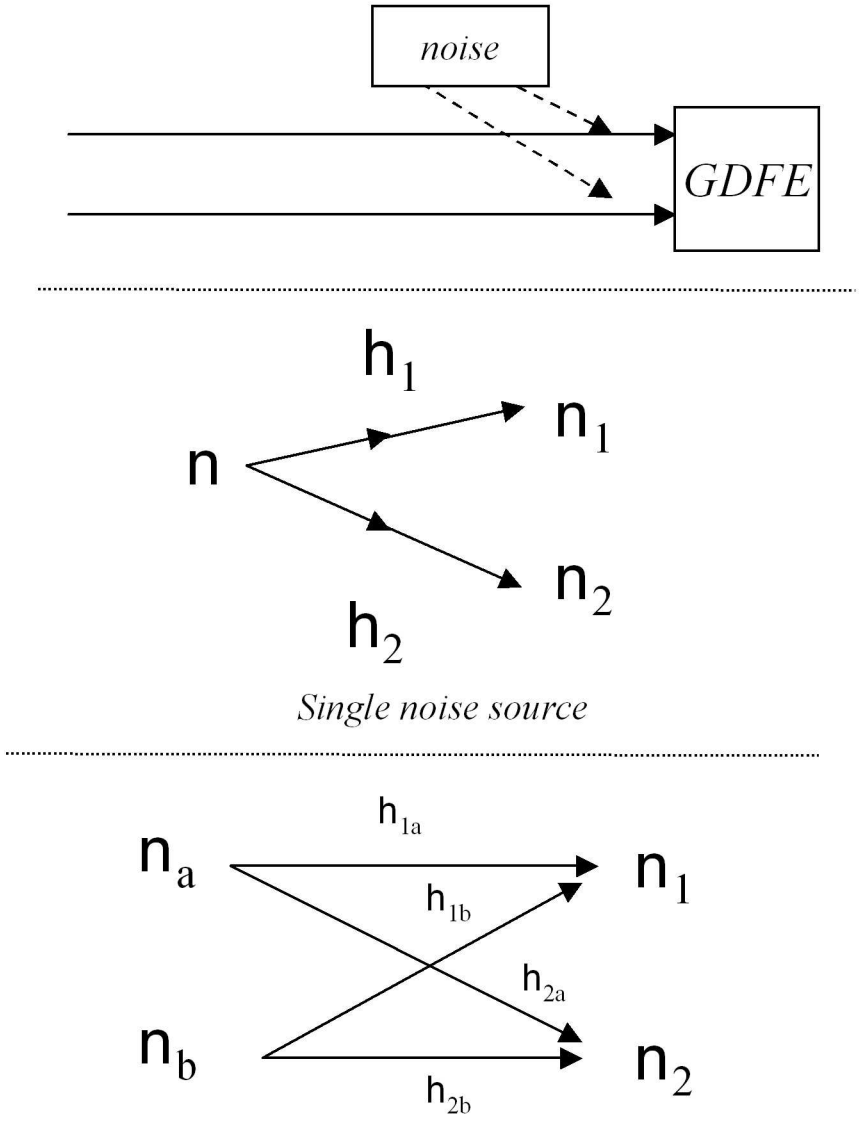

Figure 13.13: Spatial noise correlation.

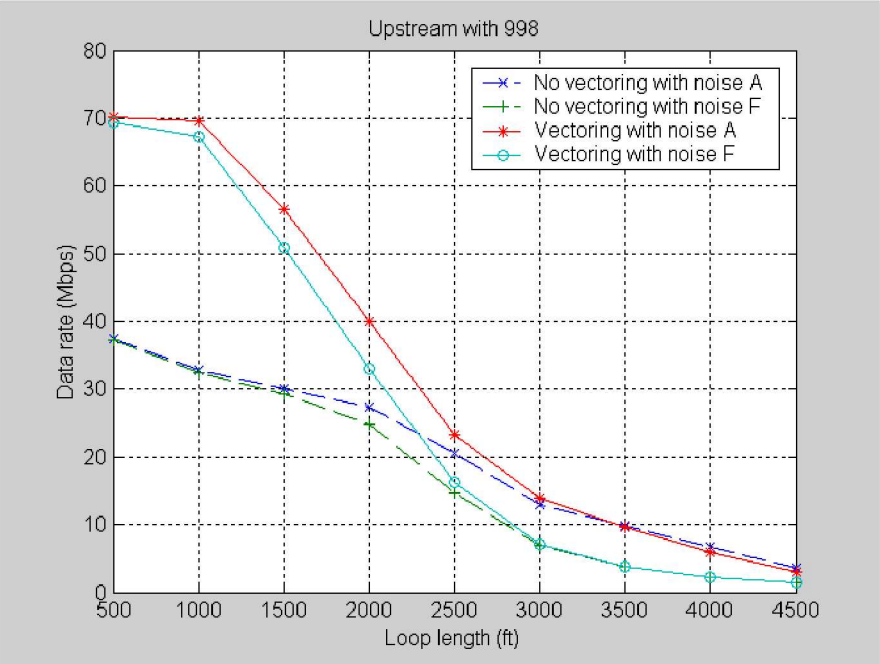

Figure 13.14: VDSL with and without vectoring - no spatial correlation of other noises.

indirectly be used to help down-direction transmissions because if the upstream direction has very high rates, then more bandwidth or dimensions can be allocated for downstream transmission (where spatial correlation cannot be exploited as well unless the receiver also uses multiple lines/antennae).

EXAMPLE 13.3.1 (VDSL) VDSL is a zippered DMT system as described in Section 4.6. The upstream direction is a vectored multiple-access channel if the DSLAM uses coordination. This system then is vectored DMT if all upstream DMT transmissions use the same master clock. The tone spacing is 4.3125 kHz with a cyclic extension of 640 samples on a sampling clock of 16 $\times$ 2.208 MHz. Up to 4096 tones can be used in either direction. Two frequency plans have been used for a frequency-division separation of upstream and downstream bands. The so-called 998 plan of North America allows up and down transmission below 138 kHz (tone 32), and also up-only transmission between 4 MHz and 5.2 MHz and between 8.5 MHz and 17.6 MHz. Two uncorrelated and uncoordinated types of crosstalk noise noise A - (less severe) and noise F (more severe) are used for testing.

Two options are investigated in Figure 13.14. In the lower curves, the upstream data rate for simple DMT use with water-filling in each of the FDM bands is used upstream with no vectored coordination or GDFE. The upper curve is the data rate achieved with vectoring and the same noises. At short line lengths, the self-crosstalk from same-direction VDSL signals dominates (while at longer distances it does not). No spatial correlation of the Noise A or Noise F was used in Figure 13.14.

Figure 13.15 illustrates the situation if the other crosstalkers are fewer in number than the number of vectored users in the upstream direction. The data rate plotted is the overall rate sum divided by the number of users since all the lines have the same channel in the configuration investigated. Again, yet an even higher data rate is achieved at short lengths. In this case the other crosstalkers overlap the band used by the upstream VDSL. No frequency plan was used and the loops simply adapted to best band use with 14.5 dBm of upstream transmit power. As can be seen in Figure 13.15, yet further data rate increase is large when spatial correlation of noise can be fully exploited. This example is revisted for bi-directional downstream vector BC and upstream vector MAC in Chapter 14.

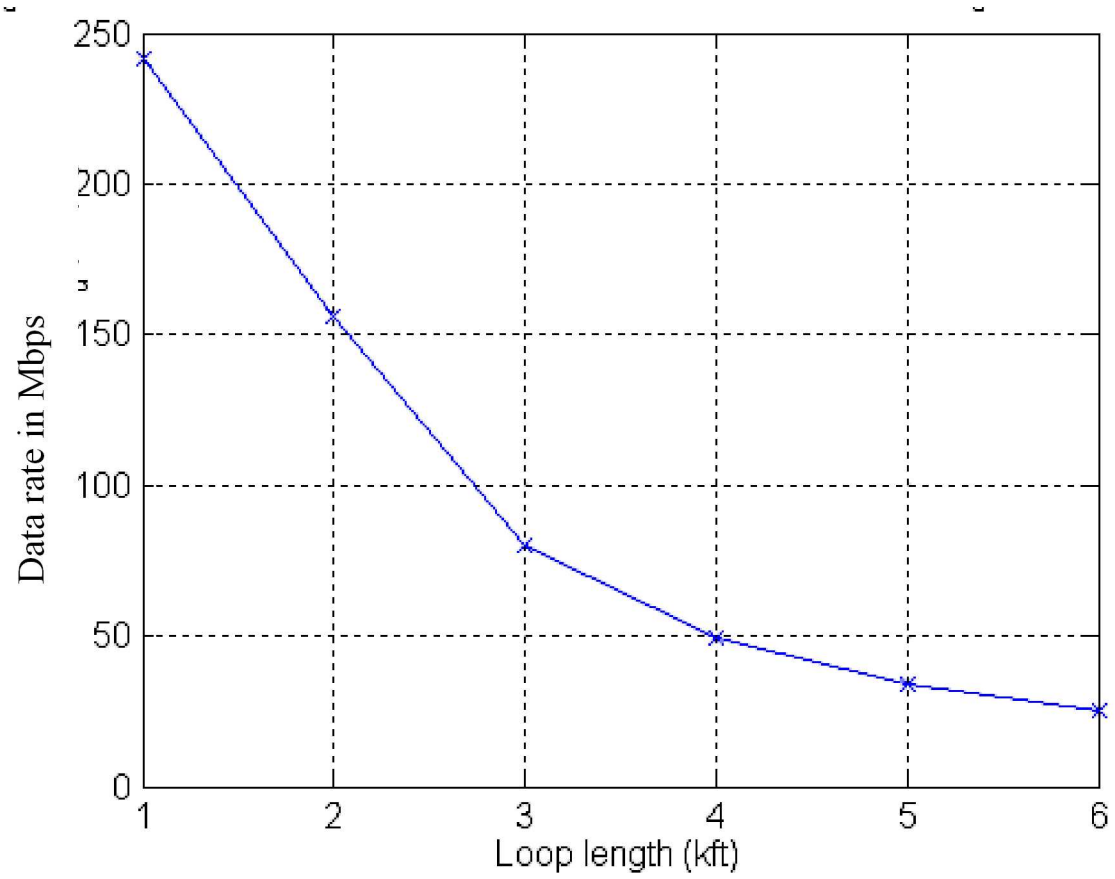

Figure 13.15: VDSL with vectoring - spatial correlation of other noises have fewer sources than the number of lines terminated on the DSLAM.

13.4 Minimization of an energy sum - Mohseni's Algorithm

This section uses some optimization-theory concepts to find the minimum weighted sum of energies (or traces of input autocorrelation matrices in general) and bit distributions of a vector DMT design that corresponds to any data rate vector $\boldsymbol{b}$. Such energy-sum minimization may be of interest when "radiation" into other systems (i.e., those that do not share the same receiver and are not part of the **MAC**) is to be somehow contained. The minimization of a weighted energy sum is also an intermediate step that enables the solution for finding the energy and bit distributions corresponding to all users to realize any point within a capacity rate region for a **MAC** in Section 13.5.

This section first poses the problem and introduces some convex/concave optimization concepts that lead to an algorithm by Mohseni that finds the minimized weighted energy sum. Essentially the same algorithm can be used to solve some related problems like maximization of a weighted rate sum and the inclusion of various constraints in the capacity formulation.

13.4.1 Minimizing a Weighted Energy Sum

The basic problem addressed is the determination of a set of U user inputs that minimize a weighted sum of energies to implement a transmission system that achieves a given set of user data rates specified by a rate tuple $[b_1\ b_2\ ...b_U]$ where $\boldsymbol{b}$ is considered to be fixed and $R_{\boldsymbol{XX}}(u)$ to be variable.

$$\begin{aligned} \min_{\{R_{\boldsymbol{xx}}(u)\}} \quad & \textstyle\sum_{u=1}^{U} w_u \cdot \text{trace}\left\{R_{\boldsymbol{xx}}(u)\right\} \\ ST: \quad & \boldsymbol{b} \succeq [b_{1,min}\ b_{2,min}\ ...b_{U,min}] \succeq \boldsymbol{0} \quad . \end{aligned} \tag{13.79}$$

While ad-hoc in terms of radiated energy, a designer with some knowledge of the details of the transmission environment outside the **MAC** would adjust the weights in the sum of energies to reflect a desire of minimum radiated energy. If one user's energy use were suspected to be more offensive, it could be weighted higher in the sum. The vector $\boldsymbol{w} \succeq 0$ controls such weighting, and it would a reasonable assumption to force this vector to have a unit sum of its elements (but not necessary). The vector $\boldsymbol{w} = [11...1]$ corresponds to the energy sum. More theoretically, this problem forces a particular solution for a given $\boldsymbol{b}$ in general. The solution does not have to be contained in the rate region $c(\boldsymbol{b})$, but the extension to such an additional constraint on individual user energies is subsequently easily addressed in Section 13.5. The energies of individual users are reflected in the trace $\left\{R_{\boldsymbol{XX}}(u)\right\}$ terms when $L_x > 1$. When $L_x = 1$, these terms reduce to the individual energies of users $\mathcal{E}_u$, making the analogy to a minimum sum energy more obvious.

The criterion of 13.79 can be specialized to VDMT as

$$\begin{aligned} \min_{\left\{R_{\boldsymbol{XX}}(u,n)\right\}} \quad & \textstyle\sum_{u=1}^{U}\sum_{n=1}^{N} w_u \cdot \text{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\} \\ ST: \quad & \boldsymbol{b} = \textstyle\sum_{n=1}^{N} [b_{1,n}\ b_{2,n}\ ...b_{U,n}] \succeq \boldsymbol{b}_{min} \succeq \boldsymbol{0} \quad . \end{aligned} \tag{13.80}$$

Equation (13.79) could be obtained from Problem (13.80) by setting $N = 1$ mathematically, likely meaning in practice that the actual size of $R_{\boldsymbol{XX}} \rightarrow R_{\boldsymbol{xx}}$ in general could be quite large, losing the benefit of VDMT's reduction of the problem into many smaller problems of less complexity. Thus solving (13.80) solves the original problem if $N = 1$. The tonal decomposition will be helpful also in simplifying the solution to the overall problem, as in Subsection 13.4.2. Both criteria are concave and always have a solution as should be obvious because using sufficient energy will always obtain the rate tuple for some order of decoding in a **MAC**.

The so-called "Lagrangian" for (13.80) is formed as

$$L(\boldsymbol{R}_{\boldsymbol{XX}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{u=1}^{U} \left(w_u \cdot \left[\sum_n \text{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\} \right] - \theta_u \cdot \left\{ \left[\sum_n b_{u,n} \right] - b_u \right\} \right) \quad , \tag{13.81}$$

with an additional implied convex constraint that any tone must have a rate tuple $\boldsymbol{b}_n$ that lies within the capacity region for the tone and any given set of $R_{\boldsymbol{XX}}(u,n), u = 1, ..., U$ for that tone, namely

$$\boldsymbol{b}_n \in c_n(R_{\boldsymbol{XX}}(u,n), \bar{H}_n) \tag{13.82}$$

where the capacity region for tone and given noise, channel, and input correlation is provided in Section 13.1 and generated by rate-sum bounds for the $2^U - 1$ possible rate sums. The non-negative quantities θ_u are rate-control constants that contribute nothing to the Lagrangian's value when the rate constraints are met, but cause the Lagrangian to reflect the constraint in a way that causes increase in its value when the constraints are not met. Minimization of the Lagrangian for the maximum non-negative values of $\boldsymbol{\theta}$ leads to a solution of the overall problem if the point satisfies the implied data-rate constraint. The implied-constraint region is convex because it is a capacity region for that tone and given $R_{\boldsymbol{XX}}(u,n)$ set. Any rate tuple for tone n outside this set is not achievable on this tone, and all rate tuples within the set are achievable. This implied constraint is a key to simplification of a algorithmic solution.

An interesting observation is that the solution to the dual problem

$$\begin{aligned} \max_{\left\{R_{\boldsymbol{XX}}(u,n)\right\}} \quad & \textstyle\sum_{n=1}^{N} \theta_u \cdot b_{u,n} \\ ST: \quad & \boldsymbol{E} \succeq \textstyle\sum_{u=1}^{U} \sum_n \operatorname{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\} \succeq \boldsymbol{0} \quad , \end{aligned} \tag{13.83}$$

which has essentially the same Lagrangian (other than constants that disappear under differentiation), thus could be optimized by essentially the same algorithm. Furthermore, when the minimum energy sum has all equal weights, so $\boldsymbol{w} = \boldsymbol{1}$, the weighted rate-sum-maximization almost never has all equal weights, and vice-versa. Tracing the weighted rate-sum is a mechanism for generating the capacity region, and thus the minimum weighted energy sum is also closely related to rate-region generation. Thus, a designer might address the problem

$$\text{Given:}\boldsymbol{b}, \boldsymbol{w} \quad \text{Find:}\boldsymbol{R}_{\boldsymbol{XX}}, \boldsymbol{\theta} \tag{13.84}$$

or equivalently the problem

$$\text{Given:}\boldsymbol{E}, \boldsymbol{\theta} \quad \text{Find:}\boldsymbol{b}, \boldsymbol{w} \quad . \tag{13.85}$$

13.4.2 Tonal Decomposition of the Lagrangian

The finite sums over indices u and n in (13.81) can be exchanged to obtain

$$L(\boldsymbol{R}_{\boldsymbol{XX}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{n=1}^{N} \left\{ \underbrace{\sum_{u=1}^{U} \left[w_u \cdot \operatorname{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\} - \sum_{u=1}^{U} \theta_u \cdot b_{u,n} \right]}_{L_n(\boldsymbol{R}_{\boldsymbol{XX}}(u,n), \boldsymbol{b}_n, \boldsymbol{w}, \boldsymbol{\theta})} \right\} + \theta_u \cdot b_u \quad , \tag{13.86}$$

where $L_n(\boldsymbol{R}_{\boldsymbol{XX}}(n), \boldsymbol{b}_n, \boldsymbol{w}, \boldsymbol{\theta})$ is a tonal Lagrangian term. The boldface $\boldsymbol{R}$ in $\boldsymbol{R}_{\boldsymbol{XX}}$ denotes all users included. This term for the user bit distribution in $\boldsymbol{b}$, or equivalently b_u values, and any given $\boldsymbol{\theta}$ depends only on tone n quantities. Thus, the overall "maximin" can be decomposed as

$$L^* = \max_{\boldsymbol{\theta}} \sum_{n=1}^{N} L_{min}(\boldsymbol{\theta}, n) = \max_{\boldsymbol{\theta}} L_{min}(\boldsymbol{\theta}) \tag{13.87}$$

where

$$L_{min}(\boldsymbol{\theta}, n) = \min_{\{R_{\boldsymbol{XX}}(u,n)\}, b_{u,n}} L_n(\boldsymbol{R}_{\boldsymbol{XX}}(n), \boldsymbol{b}_n, \boldsymbol{w}, \boldsymbol{\theta}) \quad . \tag{13.88}$$

The minimization part is thus N independent problems that may be each individually investigated for a given $\boldsymbol{\theta}$. The outer maximization can be later considered after each of the internal minimizations for any given $\boldsymbol{\theta}$ has been addressed.

As above, there is an implied **GDFE constraint** (or successive decoding constraint) that the bit vector and the autocorrelation for any tone must satisfy the known capacity relation for that tone with U users:

$$\boldsymbol{b}_n \in \left\{ \boldsymbol{b}_n \mid 0 \le \sum_{\boldsymbol{u} \subseteq \boldsymbol{U}} b_{u,n} \le \log_2 \left| \left(\sum_{u=1}^{U} \bar{H}_{u,n} R_{\boldsymbol{XX}}(u,n) \bar{H}_{u,n}^* \right) + I \right| \right\} = c_n\left(\left\{\boldsymbol{R}_{\boldsymbol{XX}}(n)\right\}, \bar{H}_n\right) \quad , \tag{13.89}$$

where again

$$\bar{H}_{u,n} = R_{\boldsymbol{NN}}^{-1/2}(n) \cdot H_{u,n} \quad . \tag{13.90}$$

Thus, there are N independent problems to be solved, each of the equivalent form:

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & \textstyle\sum_{u=1}^{U} \theta_u \cdot b_{u,n} - w_u \cdot \text{trace}\{R_{\boldsymbol{XX}}(u,n)\} = -L_{min}(\boldsymbol{\theta}, n) \\ ST: \quad & \boldsymbol{b} \in c_n\left(\left\{R_{\boldsymbol{XX}}(u,n)\right\}, \bar{H}_n\right) \quad . \end{aligned} \tag{13.91}$$

A fact noted by Tze is that for any $\boldsymbol{\theta} \succeq 0$ within the tone's capacity region that

$$\max_{\left\{R_{\boldsymbol{XX}}(u,n)\right\}} \left(\sum_{u=1}^{U} \theta_u \cdot b_{u,n} \right) \tag{13.92}$$

can only occur at one of the vertices of the region in (13.89). Any point between vertices is convex combination, and thus optimization selects the vertex with the largest θ_u to go last in order, and so forth. Thus, $\boldsymbol{\theta}$ determines the order for the problem of maximizing a convex combination of the rates. Given such a best order $\pi(u)$, it would be such that

$$\theta_{\pi^{-1}(U)} \geq \theta_{\pi^{-1}(U-1)} \geq ... \geq \theta_{\pi^{-1}(1)} \quad . \tag{13.93}$$

Since the second term on the right in (13.91) does not depend on θ, then a constraint is that a vertex point must maximize the overall tone criterion in (13.91) for each tone. Furthermore, the constant $\boldsymbol{\theta}$ (with respect to tone index n) relates that the best order is the same on all tones.

To minimize the expression on the right in (13.91), the relationship between the bits per user/tone $b_{u,n}$ and tone autocorrelation $R_{\boldsymbol{XX}}(u,n)$ can be inferred from the known best order (that is at given $\boldsymbol{\theta}$), and the tonal capacity relation becomes

$$b_{u,n} = \log_2 \left| \sum_{i=1}^{u} \bar{H}_{\pi^{-1}(i),n} R_{\boldsymbol{XX}}(\pi^{-1}(i), n) \bar{H}^*_{\pi^{-1}(i),n} + I \right| - \log_2 \left| \sum_{i=1}^{u-1} \bar{H}_{\pi^{-1}(i),n} R_{\boldsymbol{XX}}(\pi^{-1}(i), n) \bar{H}^*_{\pi^{-1}(i),n} + I \right| \quad . \tag{13.94}$$

When computing the sum $\sum_{u=1}^{U} \theta_u \cdot b_{u,n}$, each "log" term in the equation above appears twice, with different value of θ_i so with simplification the sum becomes

$$\sum_{u=1}^{U} \theta_u \cdot b_{u,n} = \sum_{u=1}^{U} \left[\theta_{\pi^{-1}(u)} - \theta_{\pi^{-1}(u)+1}\right] \cdot \left| \sum_{i=1}^{u} \bar{H}_{\pi^{-1}(i),n} R_{\boldsymbol{XX}}(\pi^{-1}(i), n) \bar{H}^*_{\pi^{-1}(i),n} + I \right| \quad . \tag{13.95}$$

Equation (13.95) permits (13.91) to be written purely as a function of the tone autocorrelation $R_{\boldsymbol{XX}}(u,n)$ matrices, $u = 1, ..., U$. This concave function can then be maximized as in Subsection 13.4.3 to find the best set of autocorrelation matrices for this tone and the given value of $\boldsymbol{\theta}$.

13.4.3 Mohseni's Algorithm and minPMAC program

Mohseni's algorithm computes the solution to the overall sum-energy minimization problem by iterating between a step of finding the best autocorrelation matrices $R_{\boldsymbol{XX}}(u,n)$ for each tone that minimize (13.91) and adding them for a fixed value of $\boldsymbol{\theta}$ and a subsequent step of finding the best $\boldsymbol{\theta}$ for the given set of $R^o_{\boldsymbol{XX}}(u,n)$. With appropriate initialization, this iterative alternation between $R_{\boldsymbol{XX}}(u,n)$ and $\boldsymbol{\theta}$ optimization will rapidly converge to the overall optimum solution if each step uses appropriate convex optimization procedures that converge to individual best solutions based on the given parameter(s). It can be summarized in two steps after initialization of $\boldsymbol{\theta} = \mathbf{1}$:

Rxx step For given $\boldsymbol{\theta}$ and each tone $n = 1, ..., N$, compute $L_{min}(\boldsymbol{\theta}, n)$ in (13.88) for each tone via some convex minimization procedure, most likely a steepest descent procedure as in Subsection 13.4.3. Sum the $L_{min}(\boldsymbol{\theta}, n)$ over n and add $\boldsymbol{\theta}^* \boldsymbol{b}_u$ as in (13.86) to form $L(\boldsymbol{R_{XX}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta})$ for the determined set of all users' tonal autocorrelation matrices.

Order Step Determine for the given outputs $L(\boldsymbol{R}_{\boldsymbol{XX}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta})$, which is evaluated at the best set of all users' tonal autocorrelation matrices from step 1, the value of $\boldsymbol{\theta}$ that maximizes this expression for the given $\boldsymbol{R}_{\boldsymbol{XX}}$, $\boldsymbol{b}$, and $\boldsymbol{w}$. Use this new value of $\boldsymbol{\theta}$ and return to step 1. This step typically uses an elliptical or "sub-gradient" optimization algorithm as in Subsection 13.4.3.

The algorithm converges if each step uses convergent algorithm for the intermediate convex optimization. The algorithm can be terminated when successively computed $L(\boldsymbol{R}_{\boldsymbol{XX}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta})$ on successive step 2's are sufficiently close to each other, or when all user bit constraints are met.

Order step and initialization

The "order step" determines the best order and may be implemented with an elliptical algorithm.

The minimization from an Rxx step of Mohseni's algorithm produces a set of $R^o_{\boldsymbol{XX}}(u,n)$ and corresponding related set of $b^o_{u,n}$. These can be substituted into the Lagrangian expression to create a function of $\boldsymbol{\theta}$, $L^o(\boldsymbol{\theta})$, that is

$$L^o(\boldsymbol{\theta}) = \sum_{n=1}^{N} \sum_{u=1}^{U} L_n(R^o_{\boldsymbol{XX}}(u,n), b^o_{u,n}, \boldsymbol{w}, \boldsymbol{\theta}) \quad . \tag{13.96}$$

This function can be evaluated at any argument, so an offset from the fixed $\boldsymbol{\theta}$ of the Rxx step is selected, that is $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} + \boldsymbol{\Delta\theta}$. The function $L^o(\boldsymbol{\theta} + \boldsymbol{\Delta\theta})$ by direct algebraic manipulation is then at this new argument

$$L_{min}(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}) \leq L^o(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}) = L_{min}(\boldsymbol{\theta}) + \sum_{u=1}^{U} \Delta\theta_u \cdot \left[b_u - \sum_{n=1}^{N} b_{u,n} \right] = L_{min}(\boldsymbol{\theta}) + \boldsymbol{\Delta\theta}^* \Delta\boldsymbol{b} \quad , \tag{13.97}$$

where $L_{min}(\boldsymbol{\theta})$ is the value from (13.88). If $\Delta\boldsymbol{b} = 0$, then the best $\boldsymbol{\theta}$ is already known and equal to the value used in the immediately preceeding Rxx step, and the algorithm is finished. If not, the expression (13.97) then allows that[12]

$$L_{min}(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}) \leq L_{min}(\boldsymbol{\theta}) + \boldsymbol{\Delta\theta}^* \Delta\boldsymbol{b} = L^o(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}) \quad . \tag{13.98}$$

So the objective is to determine such a $\boldsymbol{\Delta\theta}$ that maximizes $L^o(\boldsymbol{\theta} + \boldsymbol{\Delta\theta})$, and then a new $\boldsymbol{\theta}$ is generated for return to step Rxx. Essentially, the components of $\Delta\boldsymbol{b}$, sign and magnitude, give an indication of desirable values of $\boldsymbol{\theta}$ that upon a subsequent Rxx step would push $\Delta\boldsymbol{b}$, with any consequent change in order, to zero. $\Delta\boldsymbol{b}$ is known as a "sub-gradient" in optimization theory for the function $L_{min}(\bullet)$.

The ellipsoid method exploits the sub-gradient in an iterative procedure that will converge to the optimum value of $\boldsymbol{\theta}^o$. The algorithm must start with an ellipsoid $O(\boldsymbol{\theta})$ that contains $\boldsymbol{\theta}^o$. The algorithm iteration index will be k so at any iteration k, the estimate of $\boldsymbol{\theta}^o$ is $\boldsymbol{\theta}_k$ and will be viewed as the center of an ellipsoid, $O_k(\boldsymbol{\theta})$, viewed as a function of $\boldsymbol{\theta}$. An ellipsoid of smaller volume, $O_{k+1}(\boldsymbol{\theta})$ at iteration $k+1$ is recursively determined as the smallest containing the intersection of O_k with the half-space $(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^* \Delta\boldsymbol{b} \geq 0$. Defining $\boldsymbol{\Delta\theta}_k \triangleq (\boldsymbol{\theta} - \boldsymbol{\theta}_k)$, the new ellipsoid is then

$$O_{k+1} = O_{min} \left\{ O_k(\boldsymbol{\theta}) \bigcap \boldsymbol{\Delta\theta}^*_k \Delta\boldsymbol{b} \geq 0 \right\} \quad . \tag{13.99}$$

Each ellipse can be specified as

$$O_k = \left\{ \boldsymbol{\theta} \mid (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^* A_k^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}_k) \geq 1 \right\} \quad , \tag{13.100}$$

where A_k is a positive definite symmetric matrix that specifies the ellipsoid axes. The ellipsoid-normalized rate-difference sub-gradient vector is defined as

$$\Delta\tilde{\boldsymbol{b}}_k \triangleq \frac{\Delta\boldsymbol{b}_k}{\sqrt{\Delta\boldsymbol{b}^*_k \cdot A_k^{-1} \cdot \Delta\boldsymbol{b}_k}} \quad , \tag{13.101}$$

[12] If $\boldsymbol{\Delta\theta}^* \Delta\boldsymbol{b} < 0$, then $L_{min}(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}) \leq L_{min}(\boldsymbol{\theta})$ and $\boldsymbol{\theta} + \boldsymbol{\Delta\theta}$ cannot be the optimal solution. Recall the idea is to find a $\boldsymbol{\Delta\theta}$ that does minimize L_{min}.

which is used to simplify the recursions to get a new ellipse covering the intersection of the half plane and the old ellipse:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{1}{U+1} A_k \cdot \Delta\tilde{\boldsymbol{b}}_k \quad \text{new center} \tag{13.102}$$

$$A_{k+1} = \frac{U^2}{U^2-1}\left[A_k - \frac{2}{U+1} A_k \cdot \Delta\tilde{\boldsymbol{b}}_k \cdot \Delta\tilde{\boldsymbol{b}}_k^* \cdot A_k\right] \quad \text{new axes} \quad . \tag{13.103}$$

The volume of the ellipsoid decreases as

$$\text{vol}\,(O_{k+1}) \leq e^{-\frac{1}{2U}} \text{vol}\,(O_k) \quad , \tag{13.104}$$

implying exponential convergence in terms of the number of iterations within any single order step of Mohseni's algorithm.

To determine the initial ellipsoid that contains $\boldsymbol{\theta}^o$ on any order step, any order is selected and a user bit distribution is selected so that

$$b_{i\neq u} = b_i^o \tag{13.105}$$

$$b_u = b_u^o + 1 \quad . \tag{13.106}$$

A single pass[13] FM iterative water-filling for the selected order and bit distribution in (13.105) and (13.106) is made through all users to obtain this bit distribution, thus generating a set of $\{R_{\boldsymbol{XX}}(u,n)\}$. This set of autocorrelation functions is substituted into the Lagrangian equation, which must always be non-negative for $\boldsymbol{\theta}^o \succeq 0$, so

$$0 \leq L_{min}(\boldsymbol{\theta}^o) \leq \left[\sum_{u=1}^{U}\sum_{n} w_u \cdot \text{trace}\left(R_{\boldsymbol{XX}}(u,n)\right)\right] + \underbrace{\left[\sum_{u=1}^{U}\left(b_u - \sum_n b_{u,n}\right)\right]}_{-1} \cdot \theta_u^o \tag{13.107}$$

must hold. Rearranging this equation leads to

$$0 \leq \theta_u^o \leq \sum_{u=1}^{U}\sum_{n} w_u \cdot \text{trace}\left(R_{\boldsymbol{XX}}(u,n)\right) = \theta_{u,max} \quad . \tag{13.108}$$

The step in (13.108) is repeated for each user (that is incrementing one of the users by one bit as in (13.105) and (13.106)) to generate a $\theta_{u,max}$ as in (13.108). Thus by executing U single-pass FM IW's for each of the U bit distributions (each incrementing one user by one bit while others are held constant), a box containing $\boldsymbol{\theta}^o$ is obtained. The ellipsoid that covers this box is then defined by

$$A_0^{-1} = \begin{bmatrix} \left(\frac{1}{\theta_{1,max}}\right)^2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \left(\frac{1}{\theta_{U,max}}\right)^2 \end{bmatrix} \quad . \tag{13.109}$$

Interior point or discrete-quantization algorithm

The Rxx step can be implemented with a gradient descent method independently for each tone. Basically, $l \triangleq L_{min}(\boldsymbol{\theta}, n)$ is minimized by moving in the direction of the negative gradient of L according to

$$L_{k+1} = L_k - \mu\left(\bigtriangledown^2 L_k\right)^{-1} \bigtriangledown L_k \tag{13.110}$$

where $\bigtriangledown^2 L$ is the 2nd-derivative matrix or "Hessian" of L and $\bigtriangledown L$ is the gradient of L, both respect to the positive-definite-matrix entries of $R_{\boldsymbol{XX}}(u,n)$. The positive step size μ is chosen to be less than half the inverse of the trace of the Hessian. The Hessian is complicated but contained within the software in Subsection 13.4.3. The proper Hessian calculation should also include constraints that ensure the autocorrelation matrices are positive semi-definite.

[13]The water-filling need be executed only once for each user because a single pass using $\tilde{R}_noise(u)$ for some order will produce a solution, which is all that is necessary for initialization.

Matlab Implementation of Mohseni's Algorithm

Mehdi Mohseni has graciously donated a "minPMAC" program for the case where $L - x = 1$.

The call of the program is basically

```
(E, theta, bun) = minPMAC (G, bu, w)
```

The inputs to the function are

- w is a $U \times 1$ vector of weights for the energy sum (typically w=ones(U,1).
- bu is a $U \times 1$ vector of non-negative bits per symbol for each user.
- G is an $L_y \times U \times N$ tensor of $L_y \times U$ matrices $[G_1\ G_2\ ...\ G_N]$ with $G_n = R_{\boldsymbol{NN}}^{-1/2}(n) \cdot H_n$ from a vector DMT system. (This is the same as $\tilde{H}$.)

The outputs from the function are

- E is a $U \times N$ matrix of energies $\mathcal{E}_{u,n}$.
- theta is a $U \times 1$ vector of rates that determines the best order.
- b is a $U \times N$ matrix of $b_{u,n}$.

These programs are available to all at the EE479 web site. The main program is again minPMAC and is listed here for the curious reader. **minPMAC** is the main program to run, all the inputs and outputs are described in the function, it basically runs ellipsoid algorithm.

```
function [E, theta, b] = minPMAC(G, bu, w);
% This is the ellipsoid method part and the main function to call.

% the inputs are:
% 1)  G, an Ly by U by N channel matrix. Ly is the number of receiver dimensions/tone,
%     U is the total number of users and N is the total number of tones.
%     G(:,:,n) is the channel matrix for all users on tone n
%     and G(:,u,n) is the channel for user u on tone n. This code assumes each user
%     only has single transmit dimension/user, thus G(:,u,n) is a column vector.
% 2)  w, a U by 1 vector containing the weights for each user's power.
% 3)  bu, a U by 1 vector containing the target rates for all the users.

% the outputs are:
% 1)  E, a U by N matrix containing the powers for all users and over all tones
%     that minimizes the weighted-sum power. E(u,:) is the power allocation for
%     user u over all tones.
% 2)  theta, the optimal U by 1 dual variable vector containing optimal weights
%     of rates. theta determines the decoding order. (Largest theta is decoded last.)
% 3)  b, a U by N matrix containing the rates of all users on all tones after convergence.


bu = bu  * log(2);   %conversion from bits to nuts
err = 1e-6;
w_0 = 1000;
count = 0;
[Ly, U, N] = size(G);

theta = w_0 * ones(U,1);
A = eye(U) * (U * w_0^2);
g = w_0 * ones(U,1);
```

```
while sqrt(g' * A * g) > err

    ind = find(theta < zeros(U,1));

    while ~isempty(ind)
        g = zeros(U,1);
        g(ind(1)) = -1;
        gt = g / sqrt(g' * A * g);
        theta = theta - 1 / (U + 1) * A * gt;
        A = U^2 / (U^2 - 1) * (A - 2 / (U + 1) * A * gt * gt' * A);
        ind = find(theta < zeros(U,1));
    end
    [f, b, E] = Lag_dual_f(G, theta, w, bu);
    g = sum(b,2) - bu;
    gt = g / sqrt(g' * A * g);
    theta = theta - 1 / (U + 1) * A * gt;
    A = U^2 / (U^2 - 1) * (A - 2 / (U + 1) * A * gt * gt' * A);
    count = count+1
end

b = b /log(2);      %conversion from nuts to bits
```

The subroutine **minPtone** is the core of the code (Interior Point method). It optimizes the weighted sum of rates minus weighted sum of the powers over the capacity region. It just solves it over one tone and the results are combined in Lag_dual_f to compute the dual function over all tones.

```
function [f, b, e] = minPtone(G, theta, w)
% minPtone maximizes f = sum_{u=1}^U theta_u * b_u - sum_{u=1}^U w_u * e_u
% subject to b \in C_g(G,e)

% the inputs are:
% 1)  G, an Ly by U channel matrix. Ly is the number of receiver antennas,
%     U is the total number of users.
%     G(:,u) is the channel for user u. In this code we assume each user
%     only has single transmit antenna, thus G(:,u) is a column vector.
% 2)  theta, a U by 1 vector containing the weights for the rates.
% 3)  w, a U by 1 vector containing the weights for each user's power.

% the outputs are:
% 1)  f, the minimum value (or maximum value of the -1 * function).
% 2)  b, a U by 1 vector containing the rates for all users
%     that optimizes the given function.
% 3)  e, a U by 1 vector containing the powers for all users
%     that optimizes the given function.


[Ly, U] = size(G);
[stheta, ind] = sort(-theta);
stheta = -stheta;
sG = G(:,ind);
sw = w(ind);
```

```
NT_max_it = 1000;                                % Maximum number of Newton's
                                                 % method iterations
dual_gap = 1e-6;
mu = 10;                                         % step size for t
alpha = 0.01;                                    % back tracking line search parameters
beta = 0.5;

count = 1;
nerr = 1e-5;                                     % acceptable error for inner loop
                                                 % Newton's method

e = ones(U,1);                                   % Strictly feasible point;

t = .1;
l_p = 1;                                         % for newton's method termination



while (1+U)/t > dual_gap
   t =  t * mu;
   l_p = 1;
   count = 1;
   while l_p > nerr & count < NT_max_it

      f_val = eval_f(t * stheta, sG, e, t * sw);
                                                 % calculating function value

                                                 % calculating the hessian and gradient
      [g, h] = Hessian(t * stheta, sG, e, t * sw);

      de = -h\g;                                 % search direction

      l_p = g' * de;                             % theta(e)^2 for Newton's method

      s = 1;                                     % checking e = e+s*de feasible
                                                 % and also back tracking algorithm

      e_new = e + s * de;



      if e_new > zeros(U,1)
          f_new = eval_f(t * stheta, sG, e_new, t * sw);
          feas_check = (f_new > f_val + alpha * s * g' * de);
      else
          feas_check = 0;
      end




      while ~feas_check
```

```
        s = s * beta;
        if s < 1e-40
            l_p = nerr/2;
            break
        end
        e_new = e + s * de;

        if e_new > zeros(U,1)
            f_new = eval_f(t * stheta, sG, e_new, t * sw);
            feas_check = (f_new > f_val + alpha * s * g' * de);
        else
            feas_check = 0;
        end

    end

    e = e + s * de;                          % update e
    count = count + 1;                       % number of Newtons method iterations
  end

end


M = eye(Ly) + sG(:,1) * sG(:,1)' * e(1);
b = zeros(U,1);
b(1) =  0.5 * log(det(M));
for u = 2:U
    b(u) = -0.5 * log(det(M));
    M = M + sG(:,u) * sG(:,u)' * e(u);
    b(u) = b(u) + 0.5 * log(det(M));
end


b(ind) = b;
e(ind) = e;
f = theta' * b - w' * e;
```

Lag_dual_f sums the individual tones to compute the overall Lagrangian:

```
function [f, b, E] = Lag_dual_f(G, theta, w, bu);
% this function computes the Lagrange dual function by solving the
% optimization problem (calling the function minPtone) on each tone.
% the inputs are:
% 1)  G, an Ly by U by N channel matrix. Ly is the number of receiver antennas,
%     U is the total number of users and N is the total number of tones.
%     G(:,:,n) is the channel matrix for all users on tone n
%     and G(:,u,n) is the channel for user u on tone n. In this code we assume each user
%     only has single transmit antenna, thus G(:,u,n) is a column vector.
% 2)  theta, a U by 1 vector containing the weights for the rates.
% 3)  w, a U by 1 vector containing the weights for each user's power.
% 4)  bu, a U by 1 vector containing the target rates for all the users.

% the outputs are:
% 1)  f, the Lagrange dual function value.
% 2)  b, a U by N vector containing the rates for all users and over all tones
```

```
%     that optimizes the Lagrangian. b(u,:) is the rate allocation for user
%     u over all tones.
% 3)  E, a U by N vector containing the powers for all users and over all tones
%     that optimizes the Lagrangian. E(u,:) is the power allocation for
%     user u over all tones.


[Ly, U, N] = size(G);
f = 0;
b = zeros(U,N);
E = zeros(U,N);

% Performing optimization over all N tones,
for i = 1:N
    [temp, b(:,i), E(:,i)] = minPtone(G(:,:,i), theta, w);
    f = f + temp;
end

f = f - theta' * bu;
```

The subroutines **eval_f** and **Hessian** are two functions that compute the value of the cost function, the gradient and the Hessian of it. I have added the expressions for the gradient and the Hessian inside the Hessian file.

```
function f = eval_f(theta, G, e, w)
% This function evaluates the value of the function,

% f(e) = (theta_1 - theta_2)/2 * log det(I + G_1 * G_1^H * e_1) +
% (theta_2 - theta_3)/2 * log det(I + G_1 * G_1^H * e_1 + G_2 * G_2^H * e_2) + ...
% (theta_{U-1} - theta_U)/2 * log det(I + G_1 * G_1^H * e_1 + ... +
%    G_{U-1} * G_{U-1}^H * e_{U-1}) +
% theta_U/2 * log det(I + G_1 * G_1^H * e_1 + ... + G_U * G_U^H * e_U) - w^T * e
% + sum_{u=1}^U log(e_u)
% theta should be in decreasing order, theta_1 >= theta_2 >= ... >=theta_U.

% the inputs are:
% 1)  theta, a U by 1 vector of weights for the rates.
% 2)  G, an Ly by U channel matrix. G(:,u) is the channel
%     vector for user u. Again each user has just one transmit antenna.
% 3)  e, a U by 1 vector containing each user's power.
% 4)  w, a U by 1 vector containing weights for each user's power

% the output is f the function value given above.

[Ly,U] = size(G);

theta = 0.5 * (theta - [theta(2:U); 0]);

M = zeros(Ly,Ly,U);                          % M(:,:,i) = (I + sum_{u=1}^i G_u * G_u^H * e_u)
                                            % M is computed recursively

M(:,:,1) = eye(Ly) + G(:,1) * G(:,1)' * e(1);
f = theta(1) * log(det(M(:,:,1))) + log(e(1)) - w(1) * e(1);
for u = 2:U
    M(:,:,u) = M(:,:,u-1) + G(:,u) * G(:,u)' * e(u);
```

```
    f = f + theta(u) * log(det(M(:,:,u))) + log(e(u)) -  w(u) * e(u);
end

function [g, H] = Hessian(theta, G, e, w)

% This function calculates the gradiant (g) and the Hessian (H) of the
% function f(e) = (theta_1 - theta_2)/2 * log det(I + G_1 * G_1^H * e_1) +
% (theta_2 - theta_3)/2 * log det(I + G_1 * G_1^H * e_1 + G_2 * G_2^H * e_2) + ...
% (theta_{U-1} - theta_U)/2 * log det(I + G_1 * G_1^H * e_1 + ... + G_{U-1} * G_{U-1}^H * e_{U-1}) +
% theta_U/2 * log det(I + G_1 * G_1^H * e_1 + ... + G_U * G_U^H * e_U) - w^T * e + sum_{u=1}^U log(e
% theta should be in decreasing order, theta_1 >= theta_2 >= ... >=theta_U.

% the inputs are:
% 1)  theta, a U by 1 vector of weights for the rates.
% 2)  G, an Ly by U channel matrix. G(:,u) is the channel
%     vector for user u. Again each user has just one transmit antenna.
% 3)  e, a U by 1 vector containing each user's power.
% 4)  w, a U by 1 vector containing weights for each user's power

% the outputs are:
% 1)  g, U by 1 gradiant vector.
% 2)  H, U by U Hessian matrix.

[Ly,U] = size(G);

theta = 0.5 * (theta - [theta(2:U); 0]);

M = zeros(Ly,Ly,U);       % M(:,:,i) = (I + sum_{u=1}^i G_u * G_u^H * e_u)^{-1}
                          % M is computed recursively using matrix inversion lemma

M(:,:,1) = eye(Ly) - G(:,1) * G(:,1)' * e(1) / (1 + e(1) * G(:,1)' * G(:,1));

for u = 2:U
    M(:,:,u) = M(:,:,u-1) - M(:,:,u-1) * G(:,u) * G(:,u)' * M(:,:,u-1) * e(u) / (1 + e(u)
    * G(:,u)' * M(:,:,u-1) * G(:,u));
end

g = zeros(U,1);

% g_u = sum_{j=u}^U theta_j * G_u * M_j * G_u^H - w_u + 1/e_u
for u = 1:U
    for j = u:U
        g(u) = g(u) + theta(j) * G(:,u)' * M(:,:,j) * G(:,u);
    end
end

g = g + 1./ e - w;

% H_{u,l} = sum_{j = max(u,l)}^U -theta_j * tr(G_u * G_u^H * M_j * G_l * G_l^H * M_j)
%  - 1/e_u^2 * delta(u-l)

H = zeros(U,U);
for u = 1:U
    for l = 1:U
```

```
        for j = max(u,l):U
            H(u,l) = H(u,l) - theta(j) * trace(G(:,u) * G(:,u)' * M(:,:,j) * G(:,l)
              * G(:,l)' * M(:,:,j));
        end
    end
end
H = H - diag(1./(e.^2));
```

Example uses of minPMAC software

EXAMPLE 13.4.1 (simple scalar MAC) This first example returns to the simple scalar MAC with gains $h_2 = 0.8$ and $h_1 = 0.6$ of Section 12.2, Example 12.2.1. For this channel the number of tones can be set as $N = 1$, while $U = 2$ and $L_y = 1$. The input energies are $\mathcal{E}_1 = \mathcal{E}_2 = 1$, but are not used in the minPMAC software. a bit rate of 3 bits/dimension for each user is attempted. The noise variance is 0.0001, so the effective noise-whitened channel gains are 80 and 60 respectively. The matlab steps follow:

```
>> H=zeros(1,2,1)  % dimensioning a tensor
H =  0     0
>> H(1,1,1)=80;
>> H(1,2,1)=60
H=  80    60
>> b = [3
3];
>> w = [ 1
1];
>> [E, theta, B]=minPMAC(H, b, w)
count =  96

E =        0.6300
           0.0176
theta =    1.2801
           1.2957  % indicates last position in order
B =        2.9958
           3.0043

>> 0.5*log2(1+(6400*E(1))/(1+3600*E(2))) = 2.9858

>> 0.5*log2(1+(3600*E(2))) = 3.0043
```

The data rates are achieved and the energies fortunately are within the constraints, so this is a viable solution to achieve the rate pair of $\boldsymbol{b} = [3, 3]^*$.

This same example could be revisted with $N = 4$ to obtain (note that the rate vector is multipled by 4 since there are now 12 bits in 4 dimensions for each user

```
>> H=zeros(1,2,4);
>> H(1,1,:)=80 ;
>> H(1,2,:)=60 ;

>> [E, theta, B]=minPMAC(H, 4*b, w)
count = 96
E =
    0.6300    0.6300    0.6300    0.6300
```

```
    0.0175    0.0175    0.0175    0.0175
theta =
    1.1906
    1.2026
B =
    3.0000    3.0000    3.0000    3.0000
    3.0000    3.0000    3.0000    3.0000
```

In this case, not surprisingly, all the dimensions are shared equally. If the data rate were increased, then the result would be:

```
>> [E, theta, B]=minPMAC(H, 5*R, w)
count = 98
E =
    5.0918    5.0918    5.0918    5.0918
    0.0500    0.0500    0.0500    0.0500
theta =
    9.5305
    9.5645
B =
    3.7497    3.7497    3.7497    3.7497
    3.7504    3.7504    3.7504    3.7504
```

In this case, the data rate is too high, so the minimum energy violates the energy constraint. This second example illustrates that minPMAC does not accept energy constraints and only minimizes a weighted energy sum. Thus, it will always produce an "answer," and thus the designer needs more. The admMAC program of the next Section is the missing element for design.

13.5 Tracing the Rate Region

The **MAC** capacity rate region is characterized by U individual energy constraints, one for each user. Minimization of a sum of energies may have meaning in the context of radiated energy from a **MAC** into other systems, but does not necessarily lead to a solution that satisfies these U individual energy constraints. However, Mohseni's Algorithm of Section 13.4 can be applied with minor modification to find a solution that also satisfies the energy constraints. Subsection 13.5.1 describes the modified procedure, which can be used to trace $c(\boldsymbol{b})$. Subsection 13.5.3 describes a distributed implementation of these algorithms where bit-swapping is retained and active on each user independent of the others as would be necessary in any system trying to react to noise/channel changes in real time.

13.5.1 Satisfaction of an energy vector constraint

The modification of the minimum-energy-sum problem is:

$$\begin{array}{cc} \min\limits_{\left\{{}^{R}\boldsymbol{XX}^{(u,n)}\right\}} & 0 \\ ST: & 0 \preceq \boldsymbol{b}_{min} \preceq \sum_n [b_{1,n}\ b_{2,n}\ ...b_{U,n}] \\ & 0 \preceq \sum_n \mathrm{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\} \preceq [\mathcal{E}_1\ \mathcal{E}_2\ ...\ \mathcal{E}_U] = \boldsymbol{E} \\ & \boldsymbol{b}_n \in c_n(\bar{H}_n, \{R_{\boldsymbol{XX}}(u,n)\}_{u=1,...,U} \quad . \end{array} \tag{13.111}$$

The problem in (13.111) is the same as the minimum energy constrained problem with zero weights on all individual user energy terms, except that a new constraint bounding each energy has been added. The new Lagrangian becomes

$$L(\boldsymbol{R}_{\boldsymbol{XX}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{u=1}^{U} w_u \cdot \left[\sum_{n=1}^{N} \mathrm{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\} - \mathcal{E}_u\right] - \theta_u \cdot \left[\sum_{n=1}^{N} b_{u,n} - b_u\right] \quad , \tag{13.112}$$

with both $\boldsymbol{w} \succeq 0$ and $\boldsymbol{\theta} \succeq 0$ now as variables to be determined along with the best set of positive-semi-definite autocorrelation matrices. Thus, after minimization of the Lagrangian over the set of autocorrelation matrices for any given $\boldsymbol{\theta}$ and $\boldsymbol{w}$, it can be viewed as a function of both $\boldsymbol{\theta}$ and $\boldsymbol{w}$, so

$$L_{min}(\boldsymbol{w}, \boldsymbol{\theta}) = \min_{\left\{{}^{R}\boldsymbol{XX}^{(u,n)}\right\}} L(\boldsymbol{R}_{\boldsymbol{XX}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta}) \leq 0 \quad . \tag{13.113}$$

The minimized Lagrangian is only positive (by inspection) if the constraints cannot be satisfied for the given $\boldsymbol{b}$. If such a positive result occurs for any $\boldsymbol{\theta} \succeq 0$ and/or $\boldsymbol{w} \succeq 0$, then no solution exists.

Mohseni's algorithm still applies, but the "order" step now has an ellipsoid that is a function of both $\boldsymbol{\theta}$ and $\boldsymbol{w}$, which are combined into a larger vector $\tilde{\boldsymbol{\theta}} = [\boldsymbol{w}, \boldsymbol{\theta}]$. If after any Rxx step in Mohseni's algorithm, a positive $L_{min}(\boldsymbol{w}, \boldsymbol{\theta})$ occurs, then the algorithm terminates and declares that the energy constraints cannot be met. Defining

$$\Delta\boldsymbol{E} \triangleq \begin{bmatrix} \sum_n \mathrm{trace}\left\{R_{\boldsymbol{XX}}(1,n)\right\} - \mathcal{E}_1 \\ \vdots \\ \sum_n \mathrm{trace}\left\{R_{\boldsymbol{XX}}(U,n)\right\} - \mathcal{E}_U \end{bmatrix} \quad , \tag{13.114}$$

the sub-gradient vector now becomes

$$\boldsymbol{g}_k = \begin{bmatrix} \Delta\boldsymbol{E} \\ \Delta\boldsymbol{b} \end{bmatrix} \quad . \tag{13.115}$$

and the recursions now use normalized sub-gradient

$$\Delta\tilde{\boldsymbol{g}} \triangleq \frac{\Delta\boldsymbol{g}_k}{\sqrt{\Delta\boldsymbol{g}_k^* \cdot A_k^{-1} \cdot \Delta\boldsymbol{g}_k}} \quad , \tag{13.116}$$

which is used to simplify the recursions:

$$\tilde{\boldsymbol{\theta}}_{k+1} = \tilde{\boldsymbol{\theta}}_k - \frac{1}{U+1} A_k \cdot \Delta\tilde{\boldsymbol{g}}_k \tag{13.117}$$

$$A_{k+1} = \frac{2U^2}{4U^2-1}\left[A_k - \frac{2}{2U+1} A_k \cdot \Delta\tilde{\boldsymbol{g}}_k \cdot \Delta\tilde{\boldsymbol{g}}_k^* \cdot A_k\right] \quad . \tag{13.118}$$

The volume of the ellipsoid decreases as

$$\mathrm{vol}\,(O_{k+1}) \leq e^{-\frac{1}{4U}} \mathrm{vol}\,(O_k) \quad . \tag{13.119}$$

Initialization is easier in this case of the ellipse. Because the Lagrange multipliers $\boldsymbol{w}$ and $\boldsymbol{\theta}$ could be arbitrarily scaled, essentially any ellipse that satisfies the positivity constraints would contain a solution if such a solution exists. Thus an acceptable initialization overs the unit cube and would then be:

$$A_0^{-1} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{bmatrix} \quad . \tag{13.120}$$

The interior point method step remains the same for each tone with the $\boldsymbol{w}$ and $\boldsymbol{\theta}$ from previous iterations (the first iteration can set both equal to all ones). The tonal Lagrangian is computed in exactly the same manner for each tone, but the overall sum should conform to the Lagrangian in (13.112).

Tracing the Capacity Region

The capacity region is then found by gradually increasing the elements of $\boldsymbol{b}$ in a nested U-loop that calls Mohseni's algorithm and checks to see if $\boldsymbol{b}$ has a solution. If the algorithm does not abort, then the $\boldsymbol{b}$ point will be in the capacity region. If not, the point is not in the capacity region. This is sometimes called the **admission problem** tp see of $\boldsymbol{b}$ is an "admissable" data rate vector.

Mohseni's admMAC program

Mehdi Mohseni has kindly also provided a program that implements the admission problem, and it is called " **adminMAC**". This program also makes use of the minPtone, Hessian and eval_f subroutines that were used with minPMAC in Section 13.4 and are listed there. A new subroutine called dual_adm is also added and listed here.

The inputs to the adminMAC function are

- G (or $\bar{H}$) is an $L_y \times U \times N$ matrix of channel coefficients ($L_x = 1$).
- bu is a $U \times 1$ vector of non-negative bits per symbol $\boldsymbol{b}$ for each user.
- Eu is the $U \times 1$ vector $\mathcal{E}_{vec}$ of non-negative energies per symbol for each user.

The outputs from the function are

- E, a $U \times N$ matrix containing the energies $\mathcal{E}_{u,n}$. An all zero E indicates that the given rates in $\boldsymbol{b}$ are not achievable with given powers in $\mathcal{E}_{vec}$.
- bu is a $U \times N$ matrix of $b_{u,n}$. Again, all zero rates indicates that given $\boldsymbol{b}$ is not achievable with the given $\mathcal{E}_{vec}$.
- theta, the $U \times 1$ vector of weightings for user bits per symbol.
- w, the $U \times 1$ vector of weightings for user energies per symbol
- f, the actual value of the minimized Lagrangian.

```
function [E, b, theta, w, f] = admMAC(G, bu, Eu);
% This is the ellipsoid method part and the main function to call.

% the inputs are:
% 1)  G, an Ly by U by N channel matrix. Ly is the number of receiver antennas,
%     U is the total number of users and N is the total number of tones.
%     G(:,:,n) is the channel matrix for all users on tone n
%     and G(:,u,n) is the channel for user u on tone n. In this code we assume each user
%     only has single transmit antenna, thus G(:,u,n) is a column vector.
% 2)  Eu, a U by 1 vector containing power constraints for all the users.
% 3)  bu, a U by 1 vector containing the target rates for all the users.

% the outputs are:
% 1)  E, a U by N matrix containing the powers for all users and over all tones
%     that support the rate vector given in bu. E(u,:) is the power allocation for
%     user u over all tones. An all zero E indicates that the given rates
%     in bu are not achievable with given powers in Eu.
% 2)  b, a U by N matrix containing the rates of all users on all tones after convergence.
%     Again, all zero rates indicates that given bu is not achievable with
%     Eu.
% 3)  theta, the optimal U by 1 dual variable vector containing optimal weights
%     of rates. theta determines the decoding order.
% 4)  w, the optimal U by 1 dual variable vector containing optimal weights
%     of powers.
% 5) f, the dual optimal value.

bu = bu  * log(2);   %conversion from bits to nuts
err = 1e-6;
w_0 = 1;
count = 0;
[Ly, U, N] = size(G);

w = w_0 * ones(U,1);
theta = w_0 * ones(U,1);
A = eye(2 * U) * (2 * U * w_0^2);
g = w_0 * ones(2 * U,1);


while sqrt(g' * A * g) > err

    ind = find([w;theta] < zeros(2 * U,1));

    while ~isempty(ind)       % This part is to make sure that [w;theta]
        g = zeros(2 * U,1);
        g(ind(1)) = -1;
        gt = g / sqrt(g' * A * g);
        temp = [w;theta] - 1 / (2 * U + 1) * A * gt;
        w = temp(1:U);
        theta = temp(U + 1:2 * U);
        A = (2 * U)^2 / ((2 * U)^2 - 1) * (A - 2 / (2 * U + 1) * A * gt * gt' * A);
        ind = find([w;theta] < zeros(2 * U,1));
    end
    [f, b, E] = Dual_adm(G, theta, w, bu, Eu);
    if f < -err
```

```
        E = zeros(U,N);
        b = zeros(U,N);
        return
    end
    g = [Eu - sum(E,2);sum(b,2) - bu];
    gt = g / sqrt(g' * A * g);
    temp = [w;theta] - 1 / (2 * U + 1) * A * gt;
    w = temp(1:U);
    theta = temp(U + 1:2 * U);
    A = (2 * U)^2 / ((2 * U)^2 - 1) * (A - 2 / (2 * U + 1) * A * gt * gt' * A);
    count = count+1
end

b = b /log(2);      %conversion from nuts to bits
```

The function dual_adm is called and computes the Lagrangian dual function as the sum of the results from each tone.

```
function [f, b, E] = Dual_adm(G, theta, w, bu, Eu);
% this function computes the Lagrange dual function by solving the
% optimization problem (calling the function minPtone) on each tone.
% the inputs are:
% 1)  G, an Ly by U by N channel matrix. Ly is the number of receiver antennas,
%     U is the total number of users and N is the total number of tones.
%     G(:,:,n) is the channel matrix for all users on tone n
%     and G(:,u,n) is the channel for user u on tone n. In this code we assume each user
%     only has single transmit antenna, thus G(:,u,n) is a column vector.
% 2)  theta, a U by 1 vector containing the weights for the rates.
% 3)  w, a U by 1 vector containing the weights for each user's power.
% 4)  bu, a U by 1 vector containing the target rates for all the users.
% 5)  Eu, a U by 1 vector containing the power constraints for all the
%     users.

% the outputs are:
% 1)  f, the Lagrange dual function value.
% 2)  b, a U by N vector containing the rates for all users and over all tones
%     that optimizes the Lagrangian. b(u,:) is the rate allocation for user
%     u over all tones.
% 3)  E, a U by N vector containing the powers for all users and over all tones
%     that optimizes the Lagrangian. E(u,:) is the power allocation for
%     user u over all tones.


[Ly, U, N] = size(G);
f = 0;
b = zeros(U,N);
E = zeros(U,N);

% Performing optimization over all N tones,
for i = 1:N
    [temp, b(:,i), E(:,i)] = minPtone(G(:,:,i), theta, w);
    f = f + temp;
end
```

```
f = f - theta' * bu + w' * Eu;
```

13.5.2 Examples of the use of admMAC

This section returns to the example of Section 13.4.3 to find the admissability and then also the proper weight vector for use in minPMAC.

EXAMPLE 13.5.1 (scalar MAC revisited) The scalar MAC channel is revisted for the data rate of 3 bits per symbol on each of the two users.

```
>> H
H(:,:,1) = 80    60
H(:,:,2) = 80    60
H(:,:,3) = 80    60
H(:,:,4) = 80    60
>> b =
     3
     3
>> energy = [1
1];
>> [E, B, theta, w, L]=admMAC(H, b*4, 4*energy)
E =
    0.8799    0.8799    0.8799    0.8799
    0.8912    0.8912    0.8912    0.8912
B =
    0.7310    0.7310    0.7310    0.7310
    5.8240    5.8240    5.8240    5.8240
theta = 1.0e-007 *
    0.3601
    0.2771
w = 1.0e-006 *
    0.0898
    0.1082
L = 3.0497e-007

>> [Eopt, thetaopt, Bopt]=minPMAC(H, 4*b, w)
count = 104
Eopt =
    1.3389    1.3389    1.3389    1.3389
    1.1342    1.1342    1.1342    1.1342
thetaopt = 1.0e-007 *
    0.2012
    0.1211
Bopt =
    0.8157    0.8157    0.8157    0.8157
    5.9979    5.9979    5.9979    5.9979

>> [Eopt, thetaopt, Bopt]=minPMAC(H, 4*R, 1e6*w)
count = 97
Eopt =
    0.6303    0.6303    0.6303    0.6303
    0.0175    0.0175    0.0175    0.0175
thetaopt =
    0.1147
    0.1167
```

```
Bopt =
    3.0006    3.0006    3.0006    3.0006
    2.9997    2.9997    2.9997    2.9997
```

The first instance of the use of minPMAC has numerical problems, but notice the uniform scaling of the $\boldsymbol{w}$ weighting of the users' energy removes the issue on the second instance. Continuing we see that the rate pair (2,4) produces:

```
>> b=[2
4];

>> energy =[1
1];
>> [E, B, theta, w, L]=admMAC(H, 4*b, 4*energy)
count =   345
E =
    0.9121    0.9121    0.9121    0.9121
    0.6463    0.6463    0.6463    0.6463
B =
    0.9052    0.9052    0.9052    0.9052
    5.5924    5.5924    5.5924    5.5924
theta = 1.0e-007 *
    0.4236
    0.2745
w = 1.0e-006 *
    0.0882
    0.1107
L = 4.2049e-007

>> [Eopt, thetaopt, Bopt]=minPMAC(H, 4*b, 1e6*w)
count =    96
Eopt =
    0.5998    0.5998    0.5998    0.5998
    0.0708    0.0708    0.0708    0.0708
thetaopt =
    0.1131
    0.1218
Bopt =
    1.9999    1.9999    1.9999    1.9999
    4.0000    4.0000    4.0000    4.0000
```

However, increasing the rates to just outside the region produces:

```
>> [E, B, theta, w, L]=admMAC(H, 5*b, 4*energy)
E =  0     0     0     0
     0     0     0     0
B =  0     0     0     0
     0     0     0     0
theta =
    1.1029
    1.2991
w =
    0.7308
    0.8258
```

```
L =
   -0.6669
```

The negative Lagrangian value means the point is outside the capacity region and thus cannot be realized within the energy constraints.

For the inter-symbol interference example of Section 13.4, a similar investigation produces

EXAMPLE 13.5.2
```
>> H=zeros(1,2,8);
>> H(1,1,:)=abs(fft([1 .9],8));
>> H(1,2,:)=abs(fft([1 -1],8));
>> H=(1/sqrt(.181))*H;
>> b=[1
1];
>> H=(1/sqrt(.01))*H;
>> energy=[1
1];
>> [E, B, theta, w, L]=admMAC(H, 9*R, 8*energy)
count = 377
E =
    1.3818    1.3169    1.0554    0.8112    0.7529    0.8112    1.0554    1.3169
    0.6463    0.7676    1.0087    1.2261    1.2789    1.2261    1.0087    0.7676
B =
    2.4180    1.4474    0.4511    0.0686    0.0010    0.0686    0.4511    1.4474
         0    0.9004    1.8012    2.2963    2.4355    2.2963    1.8012    0.9004
theta =  1.0e-007 *
    0.6328
    0.3116
w =  1.0e-006 *
    0.1303
    0.1051
L =  1.0818e-007
[Eopt, thetaopt, Bopt]=minPMAC(H, 9*b, 1e6*w)
count =   83
Eopt =
    1.2276    1.2191    0.3660    0.0000    0.0000    0.0000    0.3660    1.2191
    0.0000    0.0000    1.0060    1.3747    1.3825    1.3747    1.0060    0.0000
thetaopt =
    3.3294
    3.0007
Bopt =
    2.3358    2.2219    1.1102    0.0000    0.0000    0.0000    1.1102    2.2219
         0    0.0000    0.8796    2.3756    2.4898    2.3756    0.8796    0.0000
>> sum(Bopt(1,:))=    9.0001
>> sum(Bopt(2,:))=    9.0003
>> sum(Eopt(1,:)) = 4.3978
>> sum(Eopt(2,:)) = 6.1440
```

13.5.3 Distributed Implementations

Figure 13.16 shows a possible implementation of a **MAC** system where a complex controller executes Mohseni's algorithm for all U users and passes the resulting bit and energy distributions to each of the users. The experienced transmission engineer will note that a noise change (or an energy/rate change of any of the users) or a channel change results in the need for the re-execution of Mohseni's algorithm. The

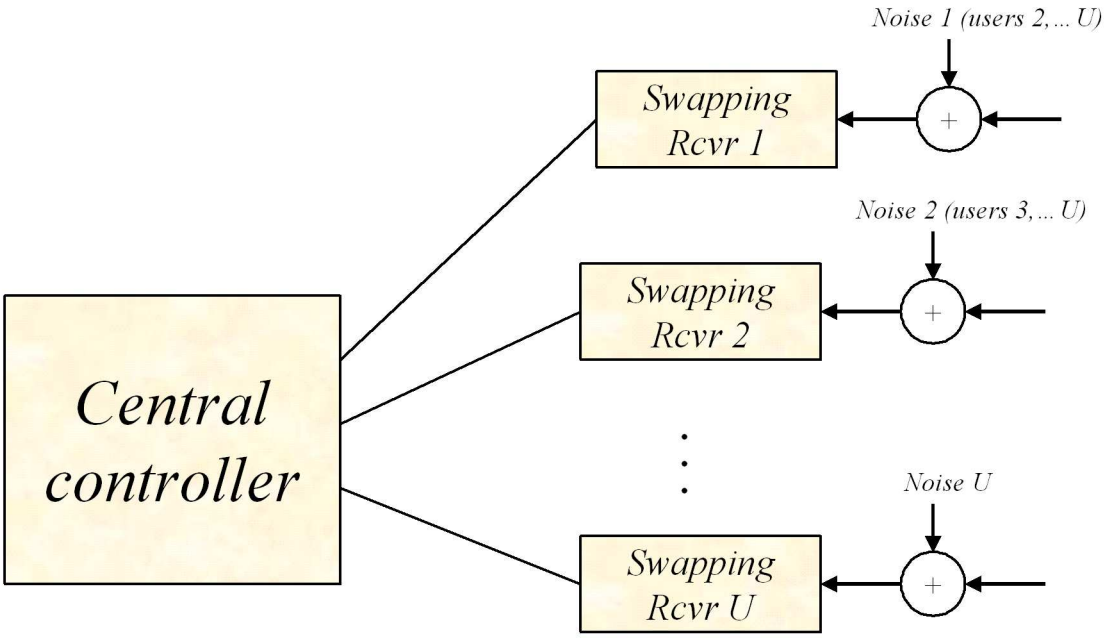

Figure 13.16: Central Control for swapping .

area of efficient incremental changes remains open to innovation although some methods may already exist.

It is possible to continue distributed "bit-swapping" (and possibly some minor gain swapping within a user's tones) to accommodate a change. First an implementation would need to ensure some margin γ_u for each of the users (it can be the same margin or different). Then an initial pass of Mohseni's algorithm is run with the following constraints:

1. reduce the energy constraints by γ_u for users if an admission problem
2. increase crosstalk by γ_u for any crosstalk component generated by user u, whether admission or min-E-sum problem
3. upon (any) completion increase the individual user energies by γ_u (recalling that the actual crosstalk in the transmission system and not in the algorithm execution is smaller by γ_u for any crosstalk generated by user u).

The results of the above single Mohseni execution the are initialized to the transceivers (or updated at sufficiently infrequent intervals to allow the controller to be able to solve the basic problem for all users). Each user then maintains an incremental energy table for its receiver and transmitter, based only upon measured noise (which includes users not yet decoded in the final optimal order from Mohseni's algorithm initial execution). A bit is swapped for any user if the difference in measured margin (or difference in next greatest and last smallest) energy exceeds 3 dB or more finely anytime the minimum margin across all tones for the user grows. The applicable PSD from infrequent uses of Mohseni's algorithm can be updated with possible change of the order. Such a PSD (and/or order) change will lead to a number of swaps but will lead to the correct bit distribution being implemented.

This procedure eliminates the need for a centralized bit-swapping procedure, which may be too complex or invoke other undesirable transmitter-synchronization effects.

Exercises - Chapter 13

13.1 *Orthogonal Codes in Space*

This problem consider a multiple-access problem in a wireless system. There is a single base station (BS) and 2 users (A, B). A, B and BS each have 2 antennas. (Figure 13.17)

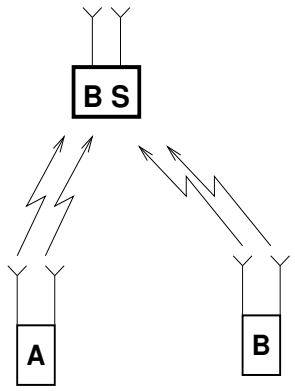

Figure 13.17: Two users with two antennas for Problem 13.1

The channel from user A to BS can be represented by 2×2 matrix $\mathbf{H_1}$, and the channel from user B to BS can be represented by 2×2 matrix $\mathbf{H_2}$. Then the received signal $\mathbf{Y}$ at the BS can be found from the following equation.

$$\mathbf{Y} = \mathbf{H_1 X_1} + \mathbf{H_2 X_2} + \mathbf{N}$$

$\mathbf{X_1}$ and $\mathbf{X_2}$ are transmitted signals from user A and B respectively, and $\mathbf{N}$ is the receiver noise ($\mathbf{N}$ is 2×1 AWGN, the power density of each noise component is -60dBm/Hz). Also, the following are known.

$$\mathbf{H_1} = \mathbf{F \Lambda_1 M^*},\ \mathbf{H_2} = \mathbf{F \Lambda_2 M^*}$$

$$\mathbf{\Lambda_1} = \begin{pmatrix} 3 & 0 \\ 0 & 0.1 \end{pmatrix}, \qquad \mathbf{\Lambda_2} = \begin{pmatrix} 0.2 & 0 \\ 0 & 4 \end{pmatrix}$$

Note that the only singular values are different from $\mathbf{H_1}$ and $\mathbf{H_2}$. (i.e $\mathbf{F}$ and $\mathbf{M}$ are same for both!) Total bandwidth of the system (=W) is 1MHz, and the maximum total power (=P) that can be used by each user at each moment is 10 dBm.

a. Find the maximum data rate of user A and B if a TDMA scheme, where only one user A transmits for half of the time, and only user B transmits for the other half of the time, is used. (Hint : What is the optimal transmission scheme if channel information is known to both transmitter and receiver?)

b. Find the maximum data rate of user A and B if a FDMA scheme, where only user A transmits in one half of the total bandwidth and user B transmits in the other half bandwidth (assume ideal rectangular filter), is used. Compare the results with TDMA results in (a). Can you explain why they are same or why they are different?

c. Both A and B want to share all of the frequencies over all time. Can you come up with an easy way of avoiding co-channel interference? What would be the codes for each user? (Hint : What's the common structure between $\mathbf{H_1}$ and $\mathbf{H_2}$?) For your chosen codes, calculate maximum data rate for each user. How much more data rate in % can we achieve compared to TDMA and FDMA cases in (a) and (b)?

   (Comment : Generally, finding a set of orthogonal space codes for given multiuser wireless channel is very hard even with perfect channel knowledge. However, in this problem, we can easily exploit the structure of channel to enhance performance.)

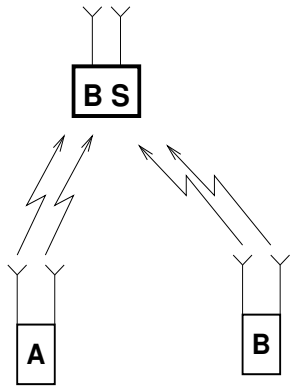

Figure 13.18: Two users with two antennas in Problem 13.2

13.2 *Chain Rule for a Matrix Channel*

Consider the 2-user case in figure 13.18 represented by

$$\mathbf{Y} = \mathbf{H_1 X_1} + \mathbf{H_2 X_2} + \mathbf{N}$$

where $\mathbf{N}$ is the receiver noise ($\mathbf{N}$ is 2×1 AWGN, the variance of each independent noise component is 0.1). The input autocorrelation matrices user 1 and user 2 are identity matrices. Also, the following are known.

$$\mathbf{H_1} = \begin{bmatrix} 4.5315 & -2.1131 \\ 2.1131 & 4.5315 \end{bmatrix}, \qquad \mathbf{H_2} = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}$$

a. First assume that each user will communicate treating the other user's signals as Gaussian interference, that is that the two receivers do not cooperate. What is the data rates that user 1 can achieve?

b. Repeat part a for user 2.

c. Instead, the designer wants to use a GDFE where coordinated joint reception is possible. What is the maximum b_1 that receiver 1 can detect reliably? While maintaining this rate for user 1, what is the maximum b_2 the receiver can achieve for user 2?

d. Reverse the order and roles of user 1 and 2 in part c and repeat.

e. What can you say about the sum rates in parts c and d?

13.3 *Rate Region for non-zero Gap*

An scalar ($L = 1$) multiple-access AWGN channel has two scalar inputs with unit eneries, gains .7 (user 2) and 1.5 (user 2) with noise variance .0001. The gap is 8.8 dB for $P_e = 10^{-6}$ on all parts of this problem.

a. Find the maximum individual rates for each user. (2 pts)

b. Find the maximum rates of the two users if the other user is treated as noise. (2 pts)

c. Find the maximum sum rate for this particular gap. (1 pt)

d. Create a formula in terms of the two independent SNRs for the two channels and the non-zero gap that has a single log term and provides the maximum sum rate. (2 pts)

e. How might other points be realized on this channel? (1 pt)

13.4 *2-user packet channel with VC*

Two users in a MAC channel with the same sampling rate have channels $H_1(D) = 1 + .8D$ and $H_2(D) = 1 - D$. The common AWGN noise power-spectral-density level is 0.1. Both users have 9 dimensional identity autocorrelation matrices for the situation of a packet size of $N = 8$ samples and a guard period of $\nu = 1$. The symbol rates of the two transmitters are synchronized at 1/9 the sampling rate.

The gap is 0 dB.

a. Find the rate sum in bits/symbol received. (4 pts)

b. Find the rate tuple for a GDFE that decodes user 1 first. (4 pts)

c. Find the rate tuple for a GDFE that decodes user 2 first. (4 pts)

d. Repeat part b if each user instead uses a vector-coding water-fill distribution for a noise that consists of the other user and background noise.

e. Repeat part c if each user instead uses a vector-coding water-fill distribution for a noise that consists of the other user and background noise.

13.5 *2-user packet channel with DMT*

Repeat Problem 13.4 for DMT. Is there a clear structure that emerges that might simplify implementation?

13.6 *Basic GDFE Program.*

Given the 2-user vector MA channel with additive white Gaussian noise (both outputs same and independent) of $\sigma^2 = .1$ and

$$P(D) = \begin{bmatrix} 1 + .9D & .8D \\ .6D & 1 + .5D \end{bmatrix} \tag{13.121}$$

Both inputs have $\bar{\mathcal{E}}_x = 1$. Vector DMT is used with $N = 512$ and a negligible cyclic prefix length of $\nu = 1$ on each of the two synchronized users. $L_x = 1$.

a. (2 pts) Determine the GDFE receiver settings and draw at frequency $n = 128$, and $n = 160$.

b. (2 pts) Determine the GDFE receiver settings at draw at frequency $n = 0$, and $n = 256$.

c. (4 pts) Write a GDFE program that automatically generates a GDFE for an input-specified tone n of a vector DMT system. The inputs should be the matrix H_n and the corresponding noise autocorrelation matrix for tone n. The nonzero input energies corresponding to each of the columns of H are also inputs. The outputs are the GDFE MS-WMF matrix and the feedback matrix G for tone n, along with the rate sum, and a vector of the bits per each dimension overall and individual SNRs.

d. (1 pt) Check your answers to part a and b with your program from part c.

13.7 *Vector IW Program.*

This problem generates an IW program. You may assume $L_x = 1$.

a. (2 pts) Write a water-filling program (or find one from EE379C) that accepts channel and noise spectra and provides energy and bits.

b. (4 pts) Modify this program to implement a IW program that treats all other users as noise

c. (2 pts) Use your program to perform vector-multiple-access IW for computation maximum rate sum on the channel of Problem 13.6 for gap= 0dB.

d. (2 pts) What are the input energies and tone rate sums for tones $n = 0$, 128, 160, and 256?

13.8 *Iterative Water-filling for MA channel.*
Refer to Problem 13.7 to begin this problem.

a. (3 pts) Modify the IW program you created in Problem 13.7 for the MA channel with a specified order so only undetected users in the order are treated as noise.

b. (3 pts) Apply your new program to a MA channel that has user 1 with $P_1(D) = 1 + .9D$, $P_2(D) = 1 - .9D$, and $\frac{\mathcal{N}_0}{2} = .181$ for the common single-dimensional output. The two inputs each have unit energy. What is the maximum sum rate? What are the rate tuples for the two possible orders?

c. (3 pts) Repeat part b except with the two channels $P_1(D) = 1 - D^2$ and $P_2 = 1 - D$.

13.9 *Rate Region for high/low.*
Using the Program of Section 13.5:

a. (3 pts) Write a program that attempts all energy pairs $\boldsymbol{E} = [i_1 \cdot 0.1 \, , i_2 \cdot 0.1]$ for $i_{max,1} = i_{max,2} = 10$ to estimate the rate region. Show your matlab code.

b. (3 pts) Find the minimum energy for the MA channel with $\frac{\mathcal{N}_0}{2} = .1$ and $P_u(D) = 1 + u \cdot .1 \cdot D$ for $U = 10$ with highest rate sum. What does this tell you?

13.10 *Sum rate of an ISI MAC with non-zero gap, courtesy S. Jagannathan*
Consider a U-user multiple access channel with ISI. When the gap is 0 dB, for any given set of PSDs for the users, the capacity region has a polymatroid structure and contains a hyperplane on its boundary at an angle of 45^o with respect to all the axes. However, when the gap is non-zero (i.e. $\Gamma > 1$ in linear units), the rate region for a given set of PSDs is a polytope that does not have a 45^o hyperplane on its boundary in general. Therefore, for a fixed PSD for each user, the sum rate is maximized at one of the U! corner points. EE479 student Joseph C. Koo has provided a dynamic programming algorithm to reduce the computational burden of enumerating all the U! points and provides the maximum sum rate and resulting spectrum allocation for the multiple access channel with non-zero gap.

In this problem, consider a 2-user ISI MAC with a single antenna at each transmitter as well as at the receiver. The results can easily be extended to U users. Let the gap for the codes of the 2 users be $\Gamma > 1$ (i.e. > 0 dB).

a. Argue that an FDM scheme maximizes the sum rate of this channel. What can you say about the ordering problem for this case? (Hint: Consider a tonal decomposition of the channel resulting in approximately flat tones. Which scheme maximizes the sum rate on each tone?).

b. Consider the MA channel that has user 1 with $H_1(D) = 1 + .9D$, user 2 with $H_2(D) = 1 - .9D$, and $\frac{N_0}{2} = .01$ for the common single-dimensional output. The gap $\Gamma = 4$ dB. The two inputs each have unit energy. Using Joseph's code, determine the maximum sum rate? Plot the spectrums for the 2 users. What does this tell you?

c. For the same channel, use the iterative water filling code of problem 13.6 with the energies reduced by Γ in linear units for each user and with gap = 0 dB. Find the resulting sum rate and plot the spectrums for the 2 users. Compare your results with part b and explain your observations.

d. Repeat parts b and c with the channels $H_1(D) = 1 - D^2$ and $H_2 = 1 - D$.

13.11 *Rate Regions.*
Using the program generated in Problem 13.7:

a. (2 pts) Estimate the rate region for the channel used in Problem 13.6 with a gap of 0 dB.

b. (2 pts) What is the maximum data rate possible for User 1?

c. (2 pts) What is the maximum data rate possible for User 2?

13.12 *FM IW for vector multiple access.*

Modify your program in Problem 13.7 to execute fixed-margin IW for any specified rate tuple. (3 pts)

a. (2 pts) Use your program to plot the spectrum use of each of the users when the two rates are equal and as large as possible with a gap of 3 dB.

b. (2 pts) Use your program to plot the spectrum use of each of the users when the rate of the first user is 1/2 the rate of the second and both are otherwise large as possible with a gap of 3 dB.

13.13 *Sum capacity of MA channel under a sum power constraint*

This problem generates a program that calculates the sum capacity of a MAC under a sum-power constraint using the parameters of Problems 13.6 and 13.7 as an example test. The following steps are taken to reach this goal:

a. (Single User Water-filling with a fixed water-level) This problem uses fixed-margin water-filling with a given fixed water-level, so that the total used power is the result. Modify the program you wrote in Problem 13.7(a) to implement FM water-filling. (2 pts)

b. (MAC water-filling with a fixed water-level) Using the program you wrote in part (a), write a second program that implements FM iterative water-filling for the MAC. That is, one user considers all other users as noise and executes FM water-filling, and other users execute the same FM water-filling iteratively. All users use the same water-level during the processes. This algorithm also converges and outputs the used energies as a result. (4 pts)

c. (Bi-section method) Suppose that the sum-power constraint is given in Problem 13.6. That is, $\sum_{u=1}^{U}\sum_{n=1}^{N}\mathcal{E}_{n,u} \leq NU\bar{\mathcal{E}}_x(=P_{tot})$ instead of the current individual power constraint of $\sum_{n=1}^{N}\mathcal{E}_{n,u} \leq N\bar{\mathcal{E}}_x$, $(u=1,\cdots,U)$. In order to determine the correct water-level $(=\lambda^*)$ for the given sum power P_{tot}, the following bi-section method is used as an outer algorithm. That is, it first fixes λ and determines the total used power by running the program obtained in part b. Then, this program keeps adjusting λ until λ^* is found. Write a program that performs the following procedures.(4 pts)
   Bi-section method *
   1. Set $\lambda_h = \lambda_{high}$ and $\lambda_l = \lambda_{low}$.
   ($\lambda_{high}/\lambda_{low}$ should be high/low enough such that $\lambda^* \in [\lambda_{low}, \lambda_{high}]$.)
   2. Set $\lambda = \frac{\lambda_h+\lambda_l}{2}$.
   3. Run the program in Problem 13.7(b) with the water-level of λ and find the sum power used by all users. Denote this sum power as P
   4. If $P < P_{tot}$, set $\lambda_l = \lambda$, else, set $\lambda_h = \lambda$.
   5. Repeat the steps from 2 to 4 until desired accuracy

d. What are the input energies and tone rates for tones n=0,128,160 and 256? (2 pts)

13.14 *Characterizing the Capacity region of a 2 user Vector DMT MAC: - M. Mohseni.*

This problem develops and uses MATLAB code to characterize the boundary points of the capacity region of a Vector DMT MAC using the provided `admMAC.m` function. This problem studies a Vector DMT MAC with $U = 2$ transmitters, each with $L_x = 1$ transmit dimension and a receiver with $L_y = 2$

receive dimensions on $N = 2$ tones. Both transmitters have unit energies, $\mathcal{E}_1 = \mathcal{E}_2 = 1$, and the whitened equivalent channel is $\bar{H}_n = R_{\mathbf{NN}}(n)^{-1/2} H_n$:

$$\bar{H}_1 = \begin{bmatrix} -0.5 & 0.1 \\ -1.7 & 0.2 \end{bmatrix} \qquad \bar{H}_2 = \begin{bmatrix} -1.2 & 1.1 \\ 1.1 & -0.1 \end{bmatrix}.$$

You can download `minPMAC.m`, `admMAC.m` and all necessary functions from class web page `http://eeclass.stanford.edu/ee479/` under the handout section.

a. Using the provided function `admMAC.m`, develop MATLAB code to find the boundary point of the capacity region of the given channel that lies on the line $b_2 = \alpha b_1$ for any given $\alpha \geq 0$. Use your code to find the boundary point that lies on the line $b_1 = b_2$. You are not required to calculate the energy distributions that achieve this point. *Hint: This problem may be solved by choosing $b_1 = b$ and $b_2 = \alpha b$ for some positive b and checking whether this rate tuple is achievable for this channel or not by using the provided function. Subsequent increase or decrease of b to find the maximum value for which the rate pair $(b, \alpha b)$ is achievable.*

b. The MATLAB function `minPMAC.m` employs the ellipsoid method to minimize a weighted-sum of powers satisfying a given set of rate constraints for each user. Closer inspection of this function reveals that the starting ellipsoid is very large, and an appropriate initialization step such as the one given in the lecture notes is missing. Modify this function to include this initialization step from the notes.

c. Using the modified `minPMAC.m`, find the minimum sum-power of two users that could achieve the boundary point you have found in part (a). Find the corresponding energy and bit distributions.

13.15 *Facets of a Gaussian Multiple-User Channel - Midterm 2008.*

Figure 13.19 illustrates a multi-user channel that uses Gaussian codes on all inputs and has additive Gaussian noise with power spectral density $\sigma^2.01$, The output is a single dimension scalar real sequence.

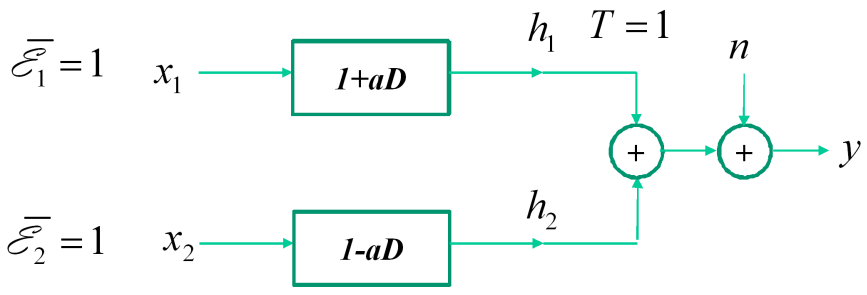

Figure 13.19: A MAC with ISI.

a. What kind of multi-user channel is this? What is U? (2 pts)

b. For parts b) to e), please let $a = 0$, $h_1 = 10$, and $h_2 = 20$. Draw the capacity region of this channel? (3 pts)

c. What is maximum rate sum? (1 pt)

d. Find frequency-division-multiplexed simultaneous water-filling spectra for the user power levels as in the figure while attaining the rate in part c and specify the corresponding rate pair? (4 pts)

e. Describe a time-sharing solution equivalent to part d . (1 pt)

f. Now please let $a = 0.8$ and $h_1 = h_2 = 1$, and you may use Vector DMT.

g. What does the frequency-division-multiplexed simultaneous water-filling spectal solution look like now? (2 pts).

h. What is the maximum rate sum? (2 pts).

i. Suppose $\bar{b}_2 = 2 \cdot \bar{b}_1$ and $\bar{b}_1 = 1$. Find an energy and bit distribution for $N = 4$ in VDMT for this data rate pair. (4 pts).

j. Find roughly the largest data rate pair of the form $(2\bar{b}_1 \ , \bar{b}_1)$ for this channel. (3 pts).

k. Show the 4 GDFEs that correspond to part i, by specifying the feedfoward sections, feedback section (or coefficient) for each of the 4 tones. (8 pts)