Generalized Decision Feedback Equalization

Generalized Decision Feedback

Generalized Decision Feedback Equalization (GDFE) is a powerful transceiver-design approach that jointly addresses space, time, and frequency dimensionality. GDFE's were introduced by this text's author and G. David Forney in [1] (1994). GDFE's fundamentally allow a common approach to many MIMO systems and to most matrix-AWGN multiuser designs. This chapter then completes this text's modulation-coding methods, while later chapters address supporting systems like timing-synchronization (Chapter 6), channel identification and adaptive tracking (Chapter 7), decoder algorithms and architecture (Chapter 8), and code designs in Chapters 9 - 10. This chapter provides a canonical general design methodology that fundamentally addresses most all linear communication systems with additive Gaussian noise.

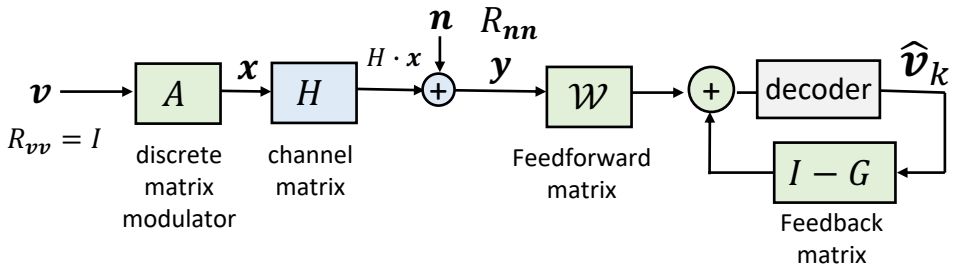


Figure 5.1: Basic Generalized Decision Feedback Equalizer (GDFE).

Figure 5.1 illustrates the basics, with a white $R_{\boldsymbol{vv}} = I$ symbol sequence $\boldsymbol{v}$ entering a discrete modulator matrix A before transmission through a matrix AWGN. The discrete receiver processes corresponding channel output vector symbol $\boldsymbol{y}$ with a linear feedforward matrix $\mathcal{W}$ before using previous decoder symbol decisions $\hat{\boldsymbol{v}}$ to reconstruct and cancel interference/crosstalk from those previous decisions' dimensions within the symbol. The dimensions can take many forms, not needing specific association with space, time, or frequency in general, although this chapter addresses each of those dimensionalities

Chapter 4 introduced symbols with large temporal dimensionality, specifically a finite $N < \infty$ is large. This chapter further develops such large symbols to a class of canonical GDFE transmission designs, of which vector coding, and its special cyclic-prefixed case DMT/C-OFDM, are special cases. This approach further expands into $\overline{N}$ sets of $\mathcal{L} = \min(\mathcal{L}_x, \mathcal{L}_y)$-dimensional GDFEs for each (temporal) index $n = 1, ..., \overline{N}$. Some of Chapter 2's MAC and BC systems are also GDFE special cases, and indeed Chapter 3's MMSE-DFE is a certain GDFE's asymptotic limiting case. This chapter then ties together the results for better understanding and extrapolates to more complete designs beyond earlier chapters' scope, particularly for Chapter 2's multiuser designs. In general, this matrix channel is $\overline{N}_y \times \overline{N}_x$, where $\overline{N}_x$ is the total number of temporal input subsymbol dimensions, and $\overline{N}_y$ is the number of output subsymbol dimensions, with then $N_x = \overline{N}_x \cdot \widetilde{N}$. Often, $\widetilde{N} = 1$ or 2.

A code may often have codewards that concatenate several subsymbols. These subsymbols become $\overline{N}_x \cdot L_x$-dimensional subsymbols when there is MIMO. An outer AEP-sense codeword (See Chapter 2, Section 3) is tacit, particularly when results with $\Gamma = 0$ dB codes appear. Tacitly, the outer codewords consist of $\overline{N}_x \cdot L_x$-dimensional subsymbols when there is MIMO. This chapter focuses on the large $\overline{N}_x \cdot \mathcal{L}_x$-dimensional subsymbol and its canonical "GDFE" processing, presuming tacitly that the outer larger concatenated code is also (typically) present, and that outer code's codewords may span many such subsymbols. Indeed, Chapter 4's separation theorem allows good AWGN-channel codes to be independently and tacitly applied outside the GDFE analysis so that any good code can be used with any good GDFE transceiver system. It will be convenient to call the $\overline{N}_x \cdot L_x$-dimensional concatenation a codeword, and consequently to call any AEP-sense longer collection of symbols also a codeword (just a longer codeword). Real (1D) or Complex (2D) elements within the symbols are the subsymbols. Collections of $\mathcal{L}_x \cdot \overline{N}_x$ dimensions are also subsymbols. When $\widetilde{N}_x \in \{1, 2$, then vector-DMT GDFEs

decompose into $\overline{N} - x$ separate smaller $\mathcal{L}_x$-dimensional GDFEs. Thus there can be very many parallel AWGNs in Figure 5.1 that may group in different ways. Chapter 4's Separation Theorem continues to apply so that good (simple scalar) AWGN codes equally well apply to linear matrix AWGN channels, although now to many GDFE-designed structures that all have the same geometric SNR and the mutual information, but may consist of different but equivalent sets.

Section 5.1's channel partitioning decomposes a general matrix-AWGN channel into canonical transmission systems, introducing the GDFE and proper single-user singularity handling. Section 5.2 then shows that existing known optimal or canonical channel-partitioning structures like Chapter 4's vector coding, discrete multi-tone, coded orthogonal-division-frequency-mutliplexed, and vectored combinations are all special GDFE cases. Section 5.2 then finds very special case of the circulant DFE that re-uses DMT/OFDM's cyclic prefix and asymptotically converges to (a minimal set of) Chapter 3's MMSE-DFE(s). Section 5.2 also investigates conditions, sometimes known as diagonally dominant or "massive MIMO," that allow linear simplification with small performance loss. Also a linear-receiver loss procedure addresses situations where the GEFE's nonlinear signal processing does not occur. Section 5.3 generalizes Chapters 3's and 4's transmit optimization for the single-user GDFE-based canonical design, finding that these often suboptimal detector structures reliably achieve capacity, or the highest data rates, and so are **canonical**.

Section 5.4 revisits Chapter 2's Gaussian MAC as a GDFE and addresses the $R_{\boldsymbol{xx}}$ optimization either to achieve a given data-rate vector $\boldsymbol{b}$ with a (possibly weighted) minimum amount of energy or to achieve a best (possibly weighted or prioritized) $\boldsymbol{b}$ with a given energy. Chapter 2's systems had either a specific $R_{\boldsymbol{xx}}$ or suggested search over all possibilities to trace a rate region - instead Section 5.4 introduces the concept of admissibility (whether a certain rate vector is possible to achieve directly without knowing the capacity region) and then how to design to achieve it with a GDFE receiver design for the given physically separated inputs. Expansion to design of a vectored DMT system with separate spatial **tonal GDFE** processing on each dimension/tone also appears in Section 5.4

Section 5.5 similarly progresses the Gaussian BC channel, with both some direct GDFE examples as well as use of a duality concept that arises naturally from GDFE concepts and achieves all Gaussian BC capacity-region points through a series of appropriately precoded GDFE's, on for each physically separated output. This dual-GDFE-set approach then circumvents need to use Chapter 2's more complex WCN-based designs. Section 5.6 then progresses to the Gaussian IC channel, again use GDFE concepts in every user connection, but also optimizes inputs across the entire set of user channels. Section 5.6 also investigates some approaches that optimize situations where implementation constraints do not permit the feedback matrix. Software analysis/design programs appear throughout this chapter to assist designers in evaluating the various design options.

Section 5.7 broadens analysis to statistically characterized channels, and remains to appear.

5.1 Generalized Channel Partitioning

This section finds and describes the many equivalent matrix-AWGN channel modulation choices that result in equivalent sets of separate simple scalar-AWGN subchannels, all with the same geometric-average SNR and bits/symbol. Chapter 4 scalar-channel sets arose from discrete Fourier transforms and/or singular value decomposition. Alternately, Chapter 3's smallest-size set of infinite-length MMSE-DFE's[1] is another equivalent set that can sometimes match Chapter 2's (and 4's) MT transmission design; and indeed both achieve the same best-coded data rate of $\overline{\mathcal{I}}(X(D);Y(D)$, for a given common power spectral density[2] $R_x(e^{-\jmath\omega T})$) and with gap $\Gamma = 0$ dB. MT systems partition the channel into a infinite number of independent scalar-AWGN subchannels that each carry a different information per subysmbol dimension, while by contrast each (of the set of) MMSE-DFE system(s)[3] partition the channel into an infinite set of AWGN subchannels that each carry the same amount of information. This later system is also the limit of Chapter 4's C-OFDM with a single good code's constellation used on the same set dimensions with the same $R_{\boldsymbol{xx}}$ and associated performance, as another instance of Chapter 4's Separation Theorem.

Discrete Matrix Channel Models: This section reviews Chapter 2's (and 4's) vector-coding (VC) partitioning for the matrix additive-Gaussian-noise channel:

$$\boldsymbol{y} = H \cdot \boldsymbol{x} + \boldsymbol{n} \quad . \tag{5.1}$$

A purely space-time MIMO channel is $L_y \times L_x$ with $\overline{N}_x = \overline{N}_y = 1$ and up to L_x and/or $L_y > 1$ dimensions. The SISO channel may also have, in frequency-time, $\overline{N}_x = \overline{N} + \nu \geq 1$ dimensions per subsymbol. The subsymbol nominal output dimensionality is $\widetilde{N} = 1$ for real, and $\widetilde{N} = 2$ for complex with Chapter 4's Vector DMT, and then $\overline{N}_x = L_x \cdot (\overline{N} + \nu)$. This chapter's GDFE developments also applies to any general H of any dimensionality. This more general $\overline{N}_y \times \overline{N}_x$ matrix H allows channel models that may expand across time/frequency and space - for instance multiple electromagnetic wave modes on a single (carrier) wavelength in waveguides (fiber or copper)[4], angular orbital momentum dimensions, or even quantum-communication dimensions. A Toeplitz H structure only applies for stationary time/frequency systems that lead to the IFFT/FFT's use for DMT/OFDM partitioning, but such Toeplitz structure does not necessarily hold more generally, particularly in space-time. One interesting non-Toeplitz H choice models time-varying convolution where the successive zeros in rows of H shift as with convolution, but each row's shifted nonzero entries vary with respect to the previous row. Such time-varying models may be useful in wireless mobile transmission systems. Other H choices model crosstalk between different copper wires in a cable, binder, or interconnection backplane. Chapter 4's VC with singular value decomposition (SVD) also did not restrict H to have any structure.

As in Chapters 2 and 4, a geometric product SNR's with singular values λ_n, and each with equal-variance $\frac{\mathcal{N}_0}{2}$ additive white Gaussian noise, is (for $\Gamma = 0$dB):

$$\text{SNR}_{vc,u} + 1 = \left[\prod_{n=1}^{\overline{N^*}} \left(1 + \frac{\mathcal{E}_n \cdot \lambda_n^2}{\frac{\mathcal{N}_0}{2}} \right) \right]^{\frac{1}{\overline{N}_x}} = \left\{ \frac{|R_{\boldsymbol{yy}}|}{|R_{\boldsymbol{nn}}|} \right\}^{\frac{1}{\overline{N}_x}} = 2^{2 \cdot \overline{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})} \quad . \tag{5.2}$$

Since Chapter 4's VC uses the optimum finite-length partitioning for any input autocorrelation matrix $R_{\boldsymbol{xx}}$, VC attains the highest reliably decodable rate bound $\overline{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})$ for this input. Any one-to-one transformation of either $\boldsymbol{x}$ or $\boldsymbol{y}$ to another random vector cannot change the mutual information as per Section 2.3's Preservation of Information Lemma. This $\mathcal{I}$-preservation admits discovery of other equivalent structures and sets of parallel channels that perform the same for the given input autocorrelation function $R_{\boldsymbol{xx}}$.

[1] See sections 3.11 and 3.12.

[2] Of the random process $X(D)$ and codes with gap $\Gamma = 0$ dB, where $X(D)$ is the discrete transform ($D \rightarrow e^{-\jmath\omega T}$ for Fourier Transform) such that $X(D) = \sum_{k=-\infty}^{\infty} x_k \cdot D^k$, with $R_x(D)$ defined in a limiting sense such that $R_x(D) = \lim_{K\to\infty} \frac{1}{2K+1} \cdot \sum_{k=-\infty}^{\infty} r_{xx,k} \cdot D^k$ where the stationary x_k has $r_{xx,k} \triangleq \mathbb{E}\left[x_k \cdot x^*_{n-k} \right]$.

[3] with correct decisions implied by $\Gamma = 0$ dB.

[4] For instance, Stanford Prof. Joseph Kahn and others have modeled intermodal crosstalk in optical fibers with such a matrix channel.

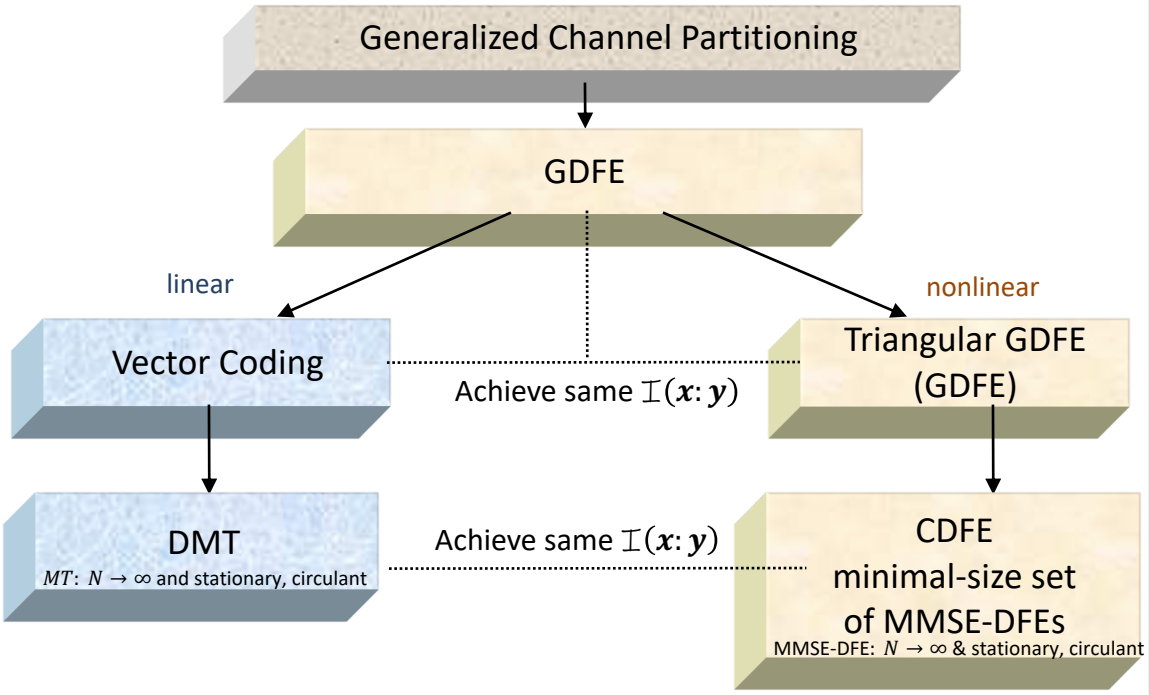


Figure 5.2: Channel-Partitioning Characterization.

Channel-Partitioning Characterization: Figure 5.2 overviews channel partitioning. There are infinitely many channel partitions that all can achieve the same aggregate data rate $b = \mathcal{I}(\boldsymbol{x};\boldsymbol{y})$ over each its corresponding set of parallel independent simple scalar-AWGN subchannels. Generalized Decision Feedback Equalization, or the "GDFE," describes and implements all these partitions. VC (and its special case DMT with cyclic prefixes) are the simplest and most natural partitions for a single-user temporal channel, but many exist. They are also linear and have optimal (ML) detectors. Nonlinear detectors use interim decisions (or modulo precoders, as in Chapter 2) and reliably achieve best data rates, despite non-ML detectors. An interesting nonlinear system for Chapter 3's ISI channels is Section 5.2's Circulant Decision Feedback, or CDFE. This CDFE, under an assumption of channel stationarity over all sampling periods, converges (with care to satisfy Appendix D's Paley Weiner Criterion or PWC) to Chapter 3's canonical minimal-size set of MMSE-DFEs. VC and DMT, under the same stationarity assumption, converge to Chapters 2 and 4's MT.

Subsection 5.1.1 describes channel-input decomposition into a channel-dependent deterministic vector space and a random input-dependent (Hilbert) space corresponding to the autocorrelation matrix $R_{\boldsymbol{xx}}$. Subsection 5.1.2 then shows that this decomposition extracts a finite-length canonical nonsingular replacement channel that is nonsingular and has the original-channel's mutual information. With this equivalent canonical channel, the GDFE then easily follows through canonical/Cholesky factorization of the information-bearing input component in Subsection 5.1.3. Subsection 5.1.4 updates Chapter 2's lossless precoders precoders, as well as those of Section 3.8, for the GDFE. Subsections 5.1.5 and 5.1.6 provide some interesting one-sided results for special channel structures that generalize those earlier found for Chapter 2's MAC and BC, respectively.

5.1.1 Discrete-Modulator Decomposition

Figure 5.3 illustrates the division of the $\overline{N}_x$-dimensional input vector space[5] $\mathbb{C}^{\overline{N}_x}$ into two mutually orthogonal subspaces: a **pass space** $\mathcal{P}_H$ containing input vectors that pass through the channel so that $H \cdot \boldsymbol{x} \neq 0$ if $\boldsymbol{x} \in \mathcal{P}_H$ and a **null space** $\mathcal{N}_H$ containing input vectors that the channel nulls so that $H \cdot \boldsymbol{x} = 0$ if $\boldsymbol{x} \in \mathcal{N}_H$.

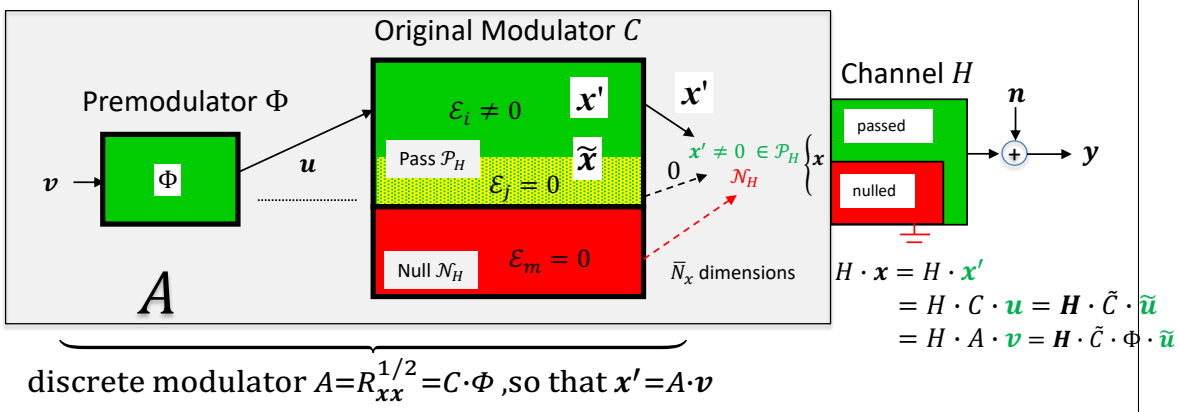


Figure 5.3: Decomposition of $\boldsymbol{x}$ into passed and nulled components.

The null space and pass space are mutually orthogonal. In general, an input vector can have nonzero components in both the pass and the null space, but those components in the null space clearly waste transmit energy because none pass to the channel output. Thus, only pass-space components carry information through the channel. A vector that has no null-space component is Figure 5.3's $\tilde{\boldsymbol{x}}$;(such a vector fails[6] Appendix D's PWC's finite-length vector equivalent if the null space has dimension 1 or larger. When this occurs, the input autocorrelation matrix is singular, $|R_{\boldsymbol{xx}}| = 0$.

Discrete-matrix modulator and pass-space projection: Figure 5.3's matrix-AWGN-channel transmitter has **discrete modulator**

$$\boldsymbol{x} = \sum_{n=1}^{\overline{N}_x} u_n \cdot \boldsymbol{c}_n = C \cdot \boldsymbol{u} \quad , \tag{5.3}$$

where the vectors $\boldsymbol{c}_n$, $n = 1, ..., \overline{N}_x$ need not be an orthonormal set, and where the random-message components multiplying these vectors, u_n, need not be independent.[7] The discrete-modulator vectors $\{\boldsymbol{c}_n\}_{n=1...\overline{N}_x}$ project into the channel's rank-ϱ_H pass space $\mathcal{P}_H$ to obtain a new set of $\varrho_H \leq \overline{N}_x$ linearly independent vectors $\{\tilde{\boldsymbol{c}}_n\}_{n=1...\varrho_H}$, that characterize $\tilde{\boldsymbol{x}} \in \mathcal{P}$ by

$$\tilde{\boldsymbol{x}} = \sum_{n=1}^{\varrho_H} \tilde{u}_n \cdot \tilde{\boldsymbol{c}}_n = \tilde{C} \cdot \tilde{\boldsymbol{u}} \quad . \tag{5.4}$$

[5] There are $2(\overline{N}_x)$ real dimensions when the input vector is complex.

[6] Said failure is not usually a problem for finite-length realizations with delay, but it's limiting case creates non-realizable filters.

[7] A special case is when the input components $u_n = X_n$ are independent and the vectors $\boldsymbol{c}_n = \boldsymbol{m}_n$ are the channel-dependent, SVD-derived, orthonormal basis of vector coding; but this need not be the case in general.

Possible $\tilde{\boldsymbol{x}}$ values span only the ϱ_H-dimensional pass subspace of the $\overline{N}_x$-dimensional complex vectors $\mathbb{C}^{\overline{N}_x}$ and are then, by construction, in one-to-one correspondence with $\tilde{\boldsymbol{u}}$ values (because of the linear independence of the vectors $\{\tilde{\boldsymbol{c}}_n\}_{n=1,\ldots,\varrho_H}$ in Equation (5.4)). Example 5.1.1.1 provides a method to compute both $\tilde{\boldsymbol{u}}$ and $\tilde{C}$ from the channel matrix H and from an arbitrary $\boldsymbol{u}$ and C. The vector $\tilde{\boldsymbol{x}}$ is the finite-length discrete-time-vector equivalent of a continuous-time modulated input signal $x(t)$ having no spectral component in the channel's dead-zones, i.e., those frequency ranges for which $H(f) = 0$ (or also those that fail Appendix D's PWC, as in Chapter 3). Figure 5.4 illustrates $\boldsymbol{x}$'s projection into $\mathcal{P}_H$.

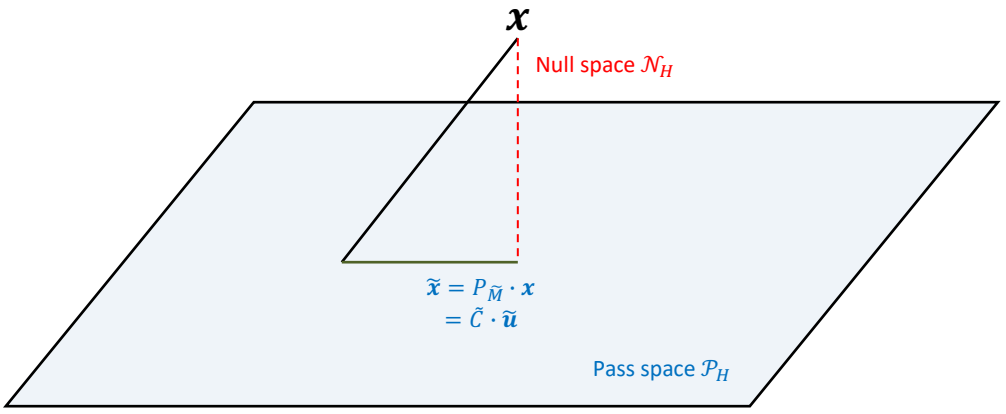


Figure 5.4: Projection illustration using singular vectors.

Mutual-Information Preservation: A general input may have nonzero null-space components, so

$$R_{\boldsymbol{x}\boldsymbol{x}} \neq R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} \quad . \tag{5.5}$$

However, any lost $\boldsymbol{x}$ components (not in $\tilde{\boldsymbol{x}}$) carry zero information through channel H. The (Gaussian) input has entropy[8]

$$\mathscr{H}_{\boldsymbol{x}} = \mathscr{H}_{\boldsymbol{x}\in\mathcal{N}} + \mathscr{H}_{\tilde{\boldsymbol{x}}} \geq \mathscr{H}_{\tilde{\boldsymbol{x}}} \quad . \tag{5.6}$$

However, basic MMSE estimation relationships with entropy (Gaussian $R_{\boldsymbol{n}\boldsymbol{n}} = I$) then also lead to[9]

$$\mathscr{H}_{\boldsymbol{x}/\boldsymbol{y}} = \mathscr{H}_{\boldsymbol{x}\in\mathcal{N}_H} + \mathscr{H}_{\tilde{\boldsymbol{x}}/\boldsymbol{y}} \geq \mathscr{H}_{\tilde{\boldsymbol{x}}/\boldsymbol{y}} \quad , \tag{5.7}$$

and thus

$$\begin{aligned} \mathcal{I}(\boldsymbol{x};\boldsymbol{y}) &= \mathscr{H}_{\boldsymbol{x}} - \mathscr{H}_{\boldsymbol{x}/\boldsymbol{y}} && (5.8)\\ &= \mathscr{H}_{\boldsymbol{x}\in\mathcal{N}_H} + \mathscr{H}_{\tilde{\boldsymbol{x}}} - \mathscr{H}_{\boldsymbol{x}\in\mathcal{N}_H} - \mathscr{H}_{\tilde{\boldsymbol{x}}/\boldsymbol{y}} && (5.9)\\ &= \mathscr{H}_{\tilde{\boldsymbol{x}}} - \mathscr{H}_{\tilde{\boldsymbol{x}}/\boldsymbol{y}} && (5.10)\\ &= \mathcal{I}(\tilde{\boldsymbol{x}};\boldsymbol{y}) && (5.11) \end{aligned}$$

so that mutual information remains the same, an intuitively gratifying result – coding/transmitting into the channel's null space does not increase the achievable data rate. Subsection 5.1.1.1 illustrates one method for determining $\mathcal{N}_H$ and $\mathcal{P}_H$ and for then finding the information-equivalent vector $\tilde{\boldsymbol{x}}$ with components $\tilde{\boldsymbol{u}}$ only in the channel's pass space. Further, since $\tilde{\boldsymbol{u}}$ is in one-to-one correspondence with possible $\tilde{\boldsymbol{x}}$ values, then (by Lemma 2.3.5's information-preservation under invertible transform)

$$\mathcal{I}(\tilde{\boldsymbol{u}},\boldsymbol{y}) = \mathcal{I}(\tilde{\boldsymbol{x}};\boldsymbol{y}) = \mathcal{I}(\boldsymbol{x};\boldsymbol{y}) \quad . \tag{5.12}$$

[8] This equation follows by viewing entropy as the sum of scalar terms based on $R_{\boldsymbol{x}\boldsymbol{x}}$'s eigendecomposition, which may have nonzero component in $\mathcal{N}_H$. Section 2.3 discusses entropy.

[9] Recall the null space does not pass to $\boldsymbol{y}$ so $H_{\boldsymbol{x}\in\mathcal{N}_H/\boldsymbol{y}} = H_{\boldsymbol{x}\in\mathcal{N}_H}$.

5.1.1.1 Discrete-Modulator Construction via channel singular vectors

Channel Singularity Elimination: Figure 5.5 illustrates an algorithm that eliminates $\boldsymbol{x}$'s components in $\mathcal{N}_H$. The noise-equivalent channel matrix[10] $\widetilde{H} = R_{\boldsymbol{nn}}^{+/2} \cdot H$ has SVD $\widetilde{H} = F \cdot \Lambda \cdot M^*$ or

$$\widetilde{H} = \sum_{n=1}^{\varrho_H} \lambda_n \cdot \boldsymbol{f}_n \cdot \boldsymbol{m}_n^* \quad \text{(step 1 in Fig. 5.5)}, \tag{5.13}$$

where $\varrho_H \leq \overline{N}_x$ is the channel rank and corresponds to the number of nonzero singular values. The singular values are non-negative real and in decreasing order $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_{\varrho_H} \geq \lambda_{\varrho_H+1} = 0$ without loss of generality. The pass space $\mathcal{P}_{\widetilde{H}}$ is then spanned by the orthonormal set of column vectors[11] of the $\overline{N}_x \times \varrho_{\widetilde{H}}$ matrix

$$\widetilde{M} = \left[\boldsymbol{m}_{\varrho_H} \; \boldsymbol{m}_2 \; ... \; \boldsymbol{m}_1\right] \quad \text{(step 1 in Fig. 5.5)}. \tag{5.14}$$

Then,

$$\tilde{\boldsymbol{x}} = P_{\widetilde{M}} \cdot \boldsymbol{x} \tag{5.15}$$

where the Hermitian matrix $P_{\widetilde{M}}$ is a "projection matrix"

$$P_{\widetilde{M}} = \widetilde{M} \cdot \left(\widetilde{M}^* \cdot \widetilde{M}\right)^{-1} \cdot \widetilde{M}^* = \widetilde{M} \cdot \widetilde{M}^* \quad \text{(step 2 in Fig. 5.5)}. \tag{5.16}$$

M's remaining column vectors similarly span the null space $\mathcal{N}_H$. The projection $\mathcal{P}_H$ of each of the $\overline{N}_x$-dimensional modulation vectors $\tilde{\boldsymbol{c}}_n$ is, see (5.3),

$$\tilde{\boldsymbol{c}}_n = P_{\widetilde{M}} \cdot \boldsymbol{c}_n \quad . \tag{5.17}$$

(5.17) may be compactly rewritten (possibly for easy matlab construction) as

$$\tilde{C}_{temp} = P_{\widetilde{M}} \cdot C \quad \text{(step 3 in Fig. 5.5)} \quad . \tag{5.18}$$

Also,

$$R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} = P_{\widetilde{M}} \cdot R_{\boldsymbol{xx}} \cdot P_{\widetilde{M}} \quad . \tag{5.19}$$

The matrix C has no more than $\overline{N}_x$, $\overline{N}_x$-dimensional, column vectors that correspond to the original modulator for $\boldsymbol{x}$. The matrix $\tilde{C}_{temp} \subseteq C$ is a rank-ϱ_H submatrix of $\overline{N}_x$-dimensional column vectors. This $\tilde{C}_{temp}$ matrix thus contains $\varrho_{\widetilde{H}}$ linearly independent columns, which reindex (the designer must record the reindexing in practice) so that

$$\tilde{C}_{temp} = [\tilde{C} \mid \tilde{C}_1] \quad \text{(step 3 in Fig. 5.5)}. \tag{5.20}$$

The $\overline{N}_x \times \varrho_H$ matrix $\tilde{C}$ corresponds to the linearly independent vectors/columns of $\tilde{C}_{temp}$, and the remaining dependent column vectors[12] are in $\tilde{C}_1$. Then, similarly reindexing the $\boldsymbol{u}$ components into the $\varrho_H \times 1$ vector $\boldsymbol{u}_2$ for $\tilde{C}$ and the remaining components[13] in vector $\boldsymbol{u}_1$ for $\tilde{C}_1$, $\tilde{\boldsymbol{x}}$ becomes

---

[10] $R_{\boldsymbol{nn}}^{+/2}$ can be any generalized inverse of any matrix square root that satisfies $R_{\boldsymbol{nn}}^{1/2} \cdot R_{\boldsymbol{nn}}^{*/2} = R_{\boldsymbol{nn}}$.

[11] Matlab places lowest index on right or bottom, while this text has that order reversed. To reverse order $F \to F \cdot J_{\overline{N}_y}$, $\Lambda \to J_{\overline{N}_y} \cdot \Lambda \cdot J_{\overline{N}_x}$, and $M \to M \cdot J_{\overline{N}_x}$.

[12] This separation is often obvious, but a general procedure follows the use of the command "qr" in matlab. By executing the command "[Q,R,J]=qr($\tilde{P}_M$ * C)," then ϱ_H is obtained by the command "rank($\tilde{P}_M$* C)," and then $\tilde{C}$ is the first ϱ_H columns of the matrix formed by Q*R*J in matlab and the last $\overline{N}_x - \varrho_H$ columns are then $\tilde{C}_1$. The rearrangement of inputs by J needs to be noted and used by the designer also. See also the Appendix G's fixmod.m and lichols.m commands.

[13] Index reversal with respect to largest at top/left here is temporary to match matlab's indexing, and this indexing is transitional only and eventually disappears.

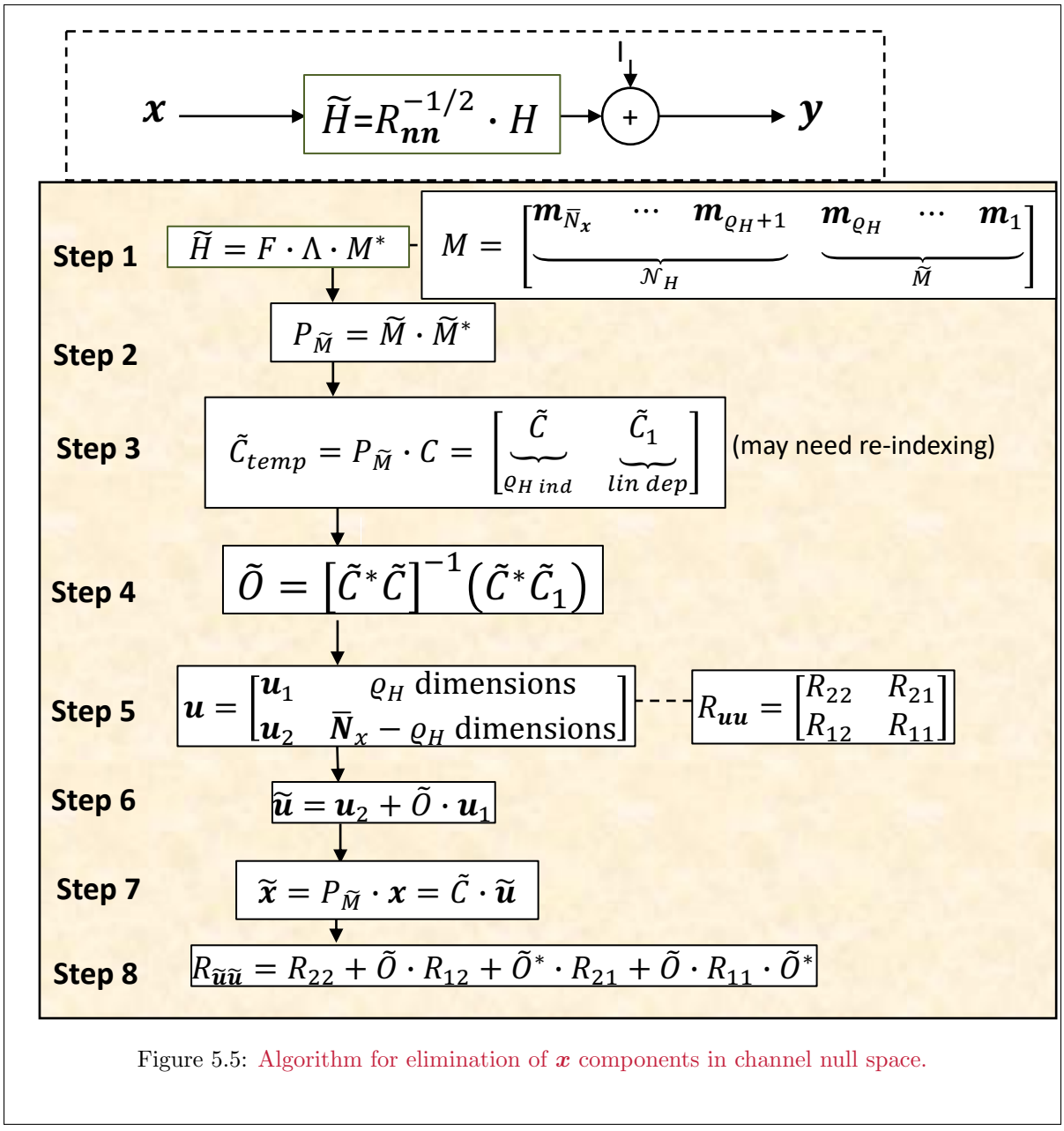


Figure 5.5: Algorithm for elimination of $\boldsymbol{x}$ components in channel null space.

$$\begin{aligned}\tilde{\boldsymbol{x}} &= P_{\widetilde{M}}\cdot\boldsymbol{x} && (5.21)\\ &= P_{\widetilde{M}}\cdot C\cdot\boldsymbol{u} && (5.22)\\ &= \tilde{C}_{temp}\cdot\boldsymbol{u} && (5.23)\\ &= [\tilde{C}\mid\tilde{C}_1]\begin{bmatrix}\boldsymbol{u}_2\\ \boldsymbol{u}_1\end{bmatrix} && (5.24)\\ &= \tilde{C}\cdot\boldsymbol{u}_2+\tilde{C}_1\cdot\boldsymbol{u}_1\quad . && (5.25)\end{aligned}$$

The new input autocorrelation matrix $R_{\boldsymbol{uu}}$ then can be partitioned according to the same labeling

$$R_{\boldsymbol{uu}}=\mathbb{E}\left\{\begin{bmatrix}\boldsymbol{u}_2\\ \boldsymbol{u}_1\end{bmatrix}[\boldsymbol{u}_2^{*}\ \ \boldsymbol{u}_1^{*}]\right\}=\begin{bmatrix}R_{22} & R_{21}\\ R_{12} & R_{11}\end{bmatrix}\quad , \qquad (5.26)$$

The remaining matrix vectors/columns $\tilde{C}_1$ are linearly dependent on $\tilde{C}$ and have projections on $\tilde{C}$ as

$$\tilde{C}_1 = \tilde{C} \cdot \left[\tilde{C}^* \cdot \tilde{C}\right]^{-1} \cdot \tilde{C}^* \cdot \tilde{C}_1 \quad . \tag{5.27}$$

(5.27) defines a matrix $\tilde{O}$ by

$$\tilde{C}_1 = \tilde{C} \cdot \left\{ \left[\tilde{C}^* \cdot \tilde{C}\right]^{-1} \cdot \tilde{C}^* \cdot \tilde{C}_1 \right\} = \tilde{C} \cdot \tilde{O} \quad \text{(step 4 in Fig. 5.5)}, \tag{5.28}$$

essentially a matrix post-multiply by $\tilde{O}$ to compute $\tilde{C}_1$ from $\tilde{C}$ in (5.20). The matrix $\tilde{O}$ is more helpful than $\tilde{C}_1$ itself as follows: Step 5 in Figure 5.5 shows explicitly the decomposition of $\boldsymbol{u}$ into the ϱ_H components $\boldsymbol{u}_1$ that multiply $\tilde{C}$ and the $\overline{N}_x - \varrho_H$ components $\boldsymbol{u}_2$ that multiply $\tilde{C}_1$. These two components combine into a single component $\tilde{\boldsymbol{u}}$ according to

$$\tilde{\boldsymbol{x}} = \tilde{C} \cdot \boldsymbol{u}_2 + \tilde{C} \cdot \tilde{O} \cdot \boldsymbol{u}_1 = \tilde{C} \cdot \underbrace{\left(\boldsymbol{u}_2 + \tilde{O} \cdot \boldsymbol{u}_1\right)}_{\text{Step 6}} = \underbrace{\tilde{C} \cdot \tilde{\boldsymbol{u}}}_{\text{Step 7}} \quad . \tag{5.29}$$

So the ϱ_H-dimensional $\tilde{\boldsymbol{u}}$ forms from $\tilde{O}$ alone by

$$\tilde{\boldsymbol{u}} \triangleq \boldsymbol{u}_2 + \tilde{O} \cdot \boldsymbol{u}_1 \quad \text{(step 6)} \, . \tag{5.30}$$

Then, the designer simply discards $\tilde{C}_{temp}$ and uses instead $\tilde{C}$. The new (more useful) input's autocorrelation matrix is

$$R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}} = R_{22} + \tilde{O} \cdot R_{12} + R_{21} \cdot \tilde{O}^* + \tilde{O} \cdot R_{11} \cdot \tilde{O}^* \quad \text{(Step 8 in Fig. 5.5)}. \tag{5.31}$$

Channel Equivalences: Figure 5.6 illustrates the original channel (with input $\boldsymbol{u}$) and with channel **null-space** components removed (with input $\tilde{\boldsymbol{u}}$). The channel outputs $\boldsymbol{y}$ are exactly the same in both cases. The lower equivalent channel captures all necessary information about the original channel while eliminating any useless transmission components that the channel singularity zeros.

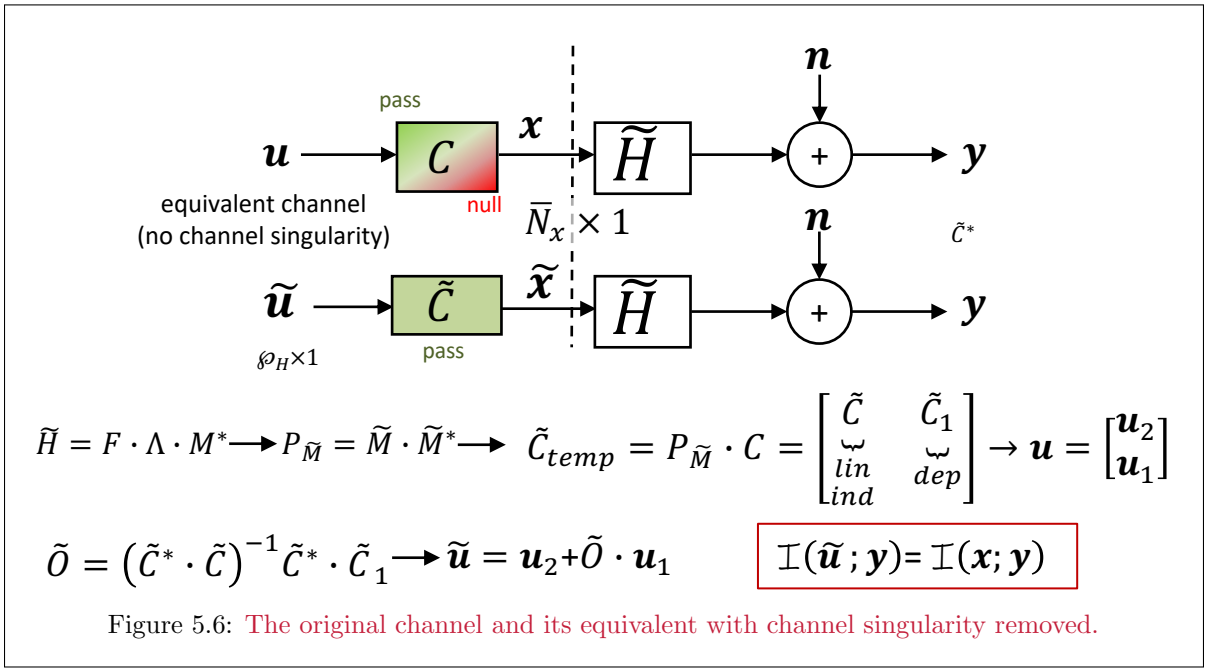


Figure 5.6: The original channel and its equivalent with channel singularity removed.

fixmod and lichols: Appendix G's matlab program fixmod.m (and its called lichols.m) execute the commands in Figure 5.5 directly[14]. The program comments are

[14]Thank you to 2021 graduate student Ethan Liang for this program.

```
 function [Ct, Ot, Ruutt] = fixmod(H_NW, Ruu, C, tol)
-------------------------------------------------------------------------
fixmod removes the part of x that lies in H_NW's nullspace
       x->xt, u->ut
 Inputs: H_NW, Ruu, C, tol
     H_NW: Noise-whitened channel
     Ruu: Autocorrelation matrix of u
     C: discrete modulator matrix
     tol: used in licols to determine rank of matrix Ctemp
 Outputs: Ct, Ot, Ruutt
     Ct: new discrete modulator with components without H_NW's nullspace
        components
     Ot: Projection matrix of C2 onto C
     Ruutt: new autocorrelation matrix for u
 This program calls licols
-------------------------------------------------------------------------
function [Xsub,idx]=licols(X,tol,mode)
 Matt J (2021). Extract linearly independent subset of matrix columns
(https://www.mathworks.com/matlabcentral/fileexchange/77437-extract-linearly-independent-subset-of-matrix-columns),
MATLAB Central File Exchange. Retrieved June 11, 2021.
 Significant modifications by Ethan Liang
Extract a linearly independent set of columns of a given matrix X
    [Xsub,idx]=licols(X)
Inputs:
  X: The given input matrix
  tol: A rank estimation tolerance. Default=1e-10
  mode: Rule for choice of columns
          'desc': choose columns based on descending values of R
          'LR': choose columns from left to right
out:
 Xsub: The extracted columns of X
 idx:  The indices (into X) of the extracted columns
```

EXAMPLE 5.1.1 ($1 + D$ Channel) An example 2×3 matrix ($R_{\boldsymbol{nn}} = I$, so $H = \widetilde{H}$)

$$H = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \tag{5.32}$$

corresponds to Chapter 3's $1 + D$ channel with $\overline{N} = 2$ and $\nu = 1$, so $\overline{N}_x = 3$; $\overline{N}_y = 2$. This channel's SVD is

$$H = F \cdot \begin{bmatrix} \sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \\ -\frac{\sqrt{2}}{\sqrt{3}} & 0 & -\frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \end{bmatrix}^* \tag{5.33}$$

and

$$\widetilde{M} = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{\sqrt{2}}{\sqrt{3}} & 0 \\ \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \quad . \tag{5.34}$$

Then

$$P_{\widetilde{M}} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & \frac{1}{3} \\ -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix} . \tag{5.35}$$

Then with a white 3-dimensional PAM input, $\boldsymbol{x} = \boldsymbol{u}$ or equivalently $C = I$ so that

$$\tilde{\boldsymbol{x}} = P_{\widetilde{M}} \cdot \boldsymbol{x} \ , \tag{5.36}$$

C_{temp} follows as

$$C_{temp} = P_{\widetilde{M}} \cdot C = P_{\widetilde{M}} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & \frac{1}{3} \\ -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix} . \tag{5.37}$$

Thus

$$\tilde{C} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{1}{3} \end{bmatrix} \quad \text{and} \quad \tilde{C}_1 = \begin{bmatrix} -\frac{1}{3} \\ \frac{1}{3} \\ \frac{2}{3} \end{bmatrix} \quad . \tag{5.38}$$

The matrix $\tilde{O}$ is then

$$\tilde{O} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad . \tag{5.39}$$

The new two-dimensional input $\tilde{\boldsymbol{u}}$ is then

$$\tilde{\boldsymbol{u}} = \begin{bmatrix} \tilde{u}_2 \\ \tilde{u}_1 \end{bmatrix} = \boldsymbol{u}_2 + \tilde{O} \cdot \boldsymbol{u}_1 = \begin{bmatrix} u_3 \\ u_2 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} u_1 = \begin{bmatrix} u_3 - u_1 \\ u_2 + u_1 \end{bmatrix} \quad . \tag{5.40}$$

The unchanged channel output is

$$\begin{aligned} \boldsymbol{y} &= \widetilde{H} \cdot P_{\widetilde{M}} \cdot C \cdot \boldsymbol{u} + \boldsymbol{n} = \widetilde{H} \cdot \tilde{\boldsymbol{x}} + \boldsymbol{n} && (5.41) \\ &= \widetilde{H} \cdot \tilde{C} \cdot \tilde{\boldsymbol{u}} + \boldsymbol{n} && (5.42) \\ &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{u}_2 \\ \tilde{u}_1 \end{bmatrix} + \boldsymbol{n} \ . && (5.43) \end{aligned}$$

This final channel-output model is, in terms of just those components of $\boldsymbol{x}$, $\tilde{u}_2 = u_3 - u_1 = x_3 - x_1$ and $\tilde{u}_1 = u_2 + u_1 = x_2 + x_1$, that "pass" through $\widetilde{H}$. Furthermore, it is clear that any input component with nonzero projection on the vector

$$\boldsymbol{m}_1 = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} \ , \tag{5.44}$$

for instance the PAM input

$$\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \ , \tag{5.45}$$

will not pass to the channel output. Thus, the two codewords +,-,+ and -,+,- that nominally look quite separated and good for coding with a minimum distance of $+2\sqrt{3}$ could not be distinguished at the channel output. Any codewords containing nonzero differences of this type simply waste energy as far as passage through the channel. For this channel, better input design than PAM would let only $\mathcal{E}_{\tilde{u}_1}$ and $\mathcal{E}_{\tilde{u}_2}$ be nonzero while avoiding any "oscillating" difference components (i.e, +,-,+ or -,+,-) between codewords.

Using the fixmod command in matlab produces:

```
>> H_NW=[1 1 0
0 1 1];
>> C=eye(3);
>> Ruu=eye(3);
>> [Ct, Ot, Ruutt] = fixmod(H_NW, Ruu, C)
Ct =
    0.6667    0.3333
    0.3333    0.6667
   -0.3333    0.3333
Ot =
   -1.0000
    1.0000
Ruutt =
    2.0000   -1.0000
   -1.0000    2.0000
```

which indeed matches (5.38) and also provides $R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}}$.

In general for noise-whitended channel $\widetilde{H} = R_{\boldsymbol{nn}}^{-1/2} \cdot H$, modulator pass-space reduction so far uses C's fixed discrete-modulator vectors and the deterministic channel $\widetilde{H}$, in particular through $\widetilde{M}$. The pass space $\mathcal{P}_{\widetilde{H}}$ and the null space $\mathcal{N}_{\widetilde{H}}$ depend only upon the channel matrix $\widetilde{H}$. However, the input vectors $\boldsymbol{x}$ are random, constructed through modulation of a random component u_n with each of the discrete-modulator vectors, $u_n \cdot \boldsymbol{c}_n$. Thus, the reduction so far to $\tilde{\boldsymbol{x}}$ depends only on input's deterministic constituents, namely $\widetilde{C}$. The projected vectors $\tilde{\boldsymbol{x}}$ themselves span a random vector ("Hilbert") space that depends (Gaussian case) on $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}$. This random vector space rarely occurs by accident, and usually occurs by design for optimum or good $R_{\boldsymbol{xx}}$ (or $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}$); such optima may have lesser rank or dimensionality such that $\varrho_x < \varrho_H$. The discrete equivalent of the PWC channel-input realization is that $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}$ and also $R_{\boldsymbol{xx}}$ be nonsingular, equivalently $\ln|R_{\boldsymbol{xx}}| < \infty$, remniscent of the PWC's integral form (see Appendix D). With $\overline{N}_x > \overline{N}_y$, the PWC is never satisfied for white inputs like PAM – so for instance, PAM is never the best-performing design on an ISI channel with finite-length symbols.

5.1.1.2 Input-Singularity Elimination

A second singularity-elimination step occurs for the input autocorrelation $R_{\boldsymbol{xx}}$. Input singularity may not relate to the channel null space, and may be intentional by design, or by accident through uniformed design. Figure 5.7 illustrates the process that this subsection subsequently describes.

The input covariance matrix $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}$ has a **eigendecomposition** (step 1 in Fig 5.7)

$$R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} = \sum_{n=1}^{\overline{N}^*} \tilde{\mathcal{E}}_{x,n} \cdot \tilde{\boldsymbol{q}}_n \cdot \tilde{\boldsymbol{q}}_n^* = \begin{bmatrix} \tilde{\boldsymbol{q}}_{\varrho_H} & \tilde{\boldsymbol{q}}_2 & \cdots & \tilde{\boldsymbol{q}}_1 \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \tilde{\mathcal{E}}_{x,\overline{N}^*} & \cdots & 0 \\ 0 & \ddots & 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & \cdots & \tilde{\mathcal{E}}_{x,1} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{q}}_{\varrho_H}^* \\ \vdots \\ \tilde{\boldsymbol{q}}_1^* \end{bmatrix} , \tag{5.46}$$

where $\tilde{\mathcal{E}}_{x,n}$ is the n^{th} eigenvalue, and $\tilde{\boldsymbol{q}}_n$ is the corresponding eigenvector[15]. In effect ϱ_x becomes $\overline{N}^*$ for the designed input; the input rank is equal to the number of used dimensions. Input-singularity removal omits the input vectors for which $\tilde{\mathcal{E}}_{x,n} = 0$ from $\tilde{\boldsymbol{x}}$'s (or $\boldsymbol{x}$'s) construction because $\tilde{\boldsymbol{x}}$ has no information-bearing component in these dimensions[16]. Define $\widetilde{Q} \triangleq [\tilde{\boldsymbol{q}}_{\overline{N}^*} \ldots \tilde{\boldsymbol{q}}_1]$ and $P_{\widetilde{Q}} \triangleq \widetilde{Q} \cdot \widetilde{Q}^*$ (step 2 in Fig. 5.7).

[15] Again Matlab often orders with largest in top left and any zeros at lower right; the permutation matrix $J_{\overline{N}_x}$ reverses this.

[16] Recalling from Chapter 2 that singular dimensions of a random vector contribute nothing to its entropy

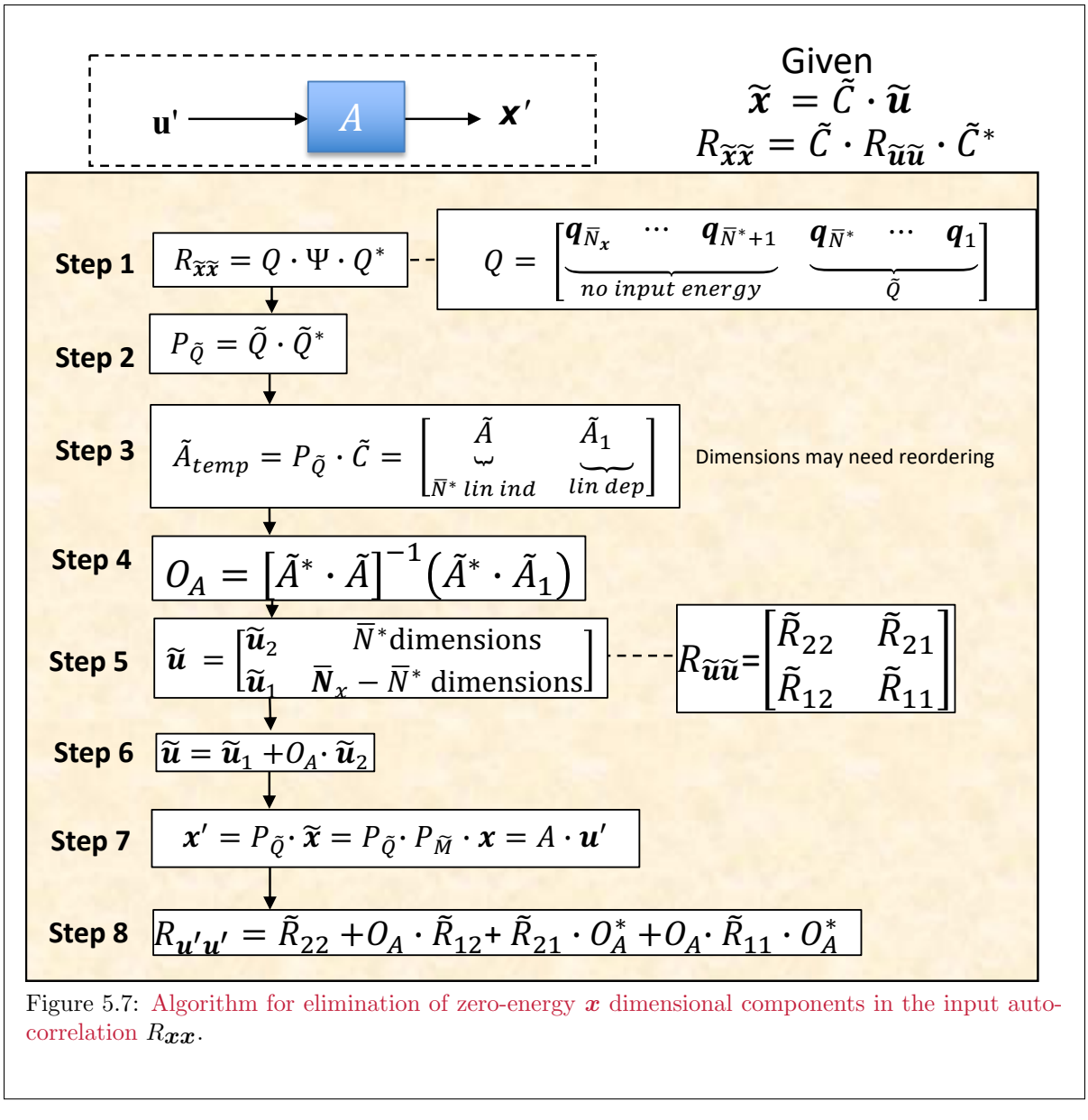


Figure 5.7: Algorithm for elimination of zero-energy $\boldsymbol{x}$ dimensional components in the input auto-correlation $R_{\boldsymbol{x}\boldsymbol{x}}$.

Figure 5.7 further suggests a second projection, now of $\tilde{\boldsymbol{x}}$ to $\boldsymbol{x}'$, into a pass-space subspace $\mathcal{P}'_H \subseteq \mathcal{P}_H$ that depends only on $R_{\boldsymbol{x}\boldsymbol{x}}$'s nonzero components/eigenmodes:

$$\begin{aligned} \boldsymbol{x}' &= P_{\widetilde{Q}} \cdot \tilde{\boldsymbol{x}} && (5.47) \\ &= P_{\widetilde{Q}} \cdot \widetilde{C} \cdot \tilde{\boldsymbol{u}} && (5.48) \\ &= \tilde{A} \cdot \boldsymbol{u}' \ , && (5.49) \end{aligned}$$

where $\tilde{A}$ and $\boldsymbol{u}'$ remain to be determined:

Without loss of generality, eigenvalue indexing satisfies $\mathcal{E}_{x,n} = 0$ for $n > \bar{N}^*$. This second "input" projection uses the $\bar{N}^* \leq \varrho_H$ column vectors of the $\bar{N}_x \times \bar{N}^*$ matrix $\tilde{A}_{temp}$

$$\tilde{A}_{temp} = P_{\widetilde{Q}} \cdot \widetilde{C} = \left[\tilde{A} \mid \tilde{A}_1\right] \quad \text{(step 3 in Fig. 5.7)} \ , \tag{5.50}$$

Then[17]

$$\tilde{A}_1 = \tilde{A} \cdot O_A \tag{5.51}$$

where $O_A = (\tilde{A}^* \cdot \tilde{A})^{-1} \cdot \tilde{A}^* \cdot \tilde{A}_1$ (step 4 in Fig. 5.7). Further,

$$\tilde{\boldsymbol{u}} = \begin{bmatrix} \tilde{\boldsymbol{u}}_2 \\ \tilde{\boldsymbol{u}}_1 \end{bmatrix} \quad \text{(step 5 in Fig. 5.7)}, \tag{5.52}$$

and

$$\boldsymbol{u}' = \tilde{\boldsymbol{u}}_2 + O_A \cdot \tilde{\boldsymbol{u}}_1 \quad \text{(step 6 in Fig. 5.7)}. \tag{5.53}$$

When $N^* = \varrho_H$, then the matrix $\widetilde{Q}$ is a square unitary matrix ($\widetilde{Q} \cdot \widetilde{Q}^* = \widetilde{Q}^* \cdot \widetilde{Q} = I$) and then $\tilde{A} = \tilde{C}$.

The entropy of $\boldsymbol{x}'$ remains the same as $\tilde{\boldsymbol{x}}$, $\mathscr{H}_{\boldsymbol{x}'} = \mathscr{H}_{\tilde{\boldsymbol{x}}}$, because removing zeroed eigenvalues from the Gaussian autocorrelation matrix does not change entropy nor the matrix itself. Thus, the mutual information remains the same as the original channel, or

$$\mathcal{I}(\boldsymbol{x}'; \boldsymbol{y}) = \mathcal{I}(\tilde{\boldsymbol{x}}; \boldsymbol{y}) = \mathcal{I}(\boldsymbol{x}; \boldsymbol{y}) \quad . \tag{5.54}$$

The final input decomposition into its nonsingular component is

$$\boldsymbol{x}' = P_{\widetilde{Q}} \cdot P_{\widetilde{M}} \cdot \boldsymbol{x} = \tilde{A} \cdot \boldsymbol{u}' \quad \text{(step 7 in Fig. 5.7)}. \tag{5.55}$$

The new input autocorrelation matrix satisfies

$$R_{\boldsymbol{x}'\boldsymbol{x}'} = \tilde{A} \cdot R_{\boldsymbol{u}'\boldsymbol{u}'} \cdot \tilde{A}^* = \tilde{A} \cdot \left(\widetilde{R}_{22} + O_A \cdot \widetilde{R}_{12} + \widetilde{R}_{21} \cdot O_A^* + O_A \cdot \widetilde{R}_{11} \cdot O_A^* \right) \cdot A^* \quad \text{(step 8, Fig. 5.7)}, \tag{5.56}$$

where $R_{\boldsymbol{u}'\boldsymbol{u}'}$ is necessarily nonsingular with rank $\varrho_{x'} = \overline{N}^*$ and of course the matrix entries $\widetilde{R}_{11}$, $\widetilde{R}_{12}$, $\widetilde{R}_{21}$ and $\widetilde{R}_{22}$ correspond to $\tilde{\boldsymbol{u}}$, and hence then also derive from $\boldsymbol{u}$. For every value that the random vector $\boldsymbol{x}'$ takes with nonzero probability, there is a unique $\boldsymbol{u}'$ value that corresponds to it, so these two are in one-to-one correspondence, and thus

$$\mathcal{I}(\boldsymbol{u}'; \boldsymbol{y}) = \mathcal{I}(\boldsymbol{x}'; \boldsymbol{y}) = \mathcal{I}(\boldsymbol{x}; \boldsymbol{y}) \quad . \tag{5.57}$$

Essentially, $\boldsymbol{u}'$ generates $\boldsymbol{x}'$ and (5.57) is thus the only mutual information of interest on channel $\widetilde{H}$.

The input's two-stage reduction to $\boldsymbol{u}'$, is the finite-length equivalent to Chapter 3's finding only that transmit band over which there is nonzero channel-output energy (or which satisfies Appendix D's PWC) and Chapter 3's focusing attention on this band only for the infinite-length MMSE-DFE. Since the mutual information is the same as the original system, and indeed also the same as the known optimum VC for the input $R_{\boldsymbol{x}\boldsymbol{x}}$, a nonsingular equivalent system now exists.

fixin.m: Appendix G's program "fixin.m" removes the input singularity[18].

```
>> help fixin
   function [At, OA, Ruupp] = fixin(Ruutt, Ct, tol)
 ------------------------------------------------------------------------
  xt->xp, ut->up (Reducing ranking of ut->up, possible b/c Rxxtt singlar)
  Inputs: Ruutt, Ct, tol
    Ruutt: autocorrelation matrix of utilde
    Ct: "nullspace of H"-adjusted discrete modulator matrix
    tol: used in licols to determine rank of matrix Ctemp (default is 1e-10)
  Outputs: At, OA, Ruupp
    At: "nullspace of H & Rxx singularity"-adjusted discrete modulator matrix
    OA: projection matrix of A2 onto A
    Ruupp: autocorrelation matrix of uprime
 ------------------------------------------------------------------------
```

[17] This separation is often obvious, but a general procedure follows the use of the command "qr" in matlab. By executing the command "[Q,R,J]=qr($\tilde{P}_{\widetilde{Q}}$ * $\tilde{C}$)," then N^* is obtained by the command "rank($\tilde{P}_{\widetilde{Q}}$ * $\tilde{C}$)," and then A is the first N^* colums of the matrix formed by Q*R*J in matlab and the last $\varrho_H - N^*$ columns are then A_2. The rearrangement of inputs by J needs to be noted and used by the designer also.

[18] Thanks again to 2021 Student Ethan Liang for this program.

[reducing a singular input for the $1+D$ channel] This example returns to Example 5.1.1's $1+D$ AWGN channel with $\overline{N}+\nu=3$ ($N_x=4$): This channel's input originally had $\varrho_x=3$ with $R_{\boldsymbol{xx}}=I$, the elimination of channel-null-space input components reduced the input to $\varrho_{\tilde{u}}=2$; therefore there are no singular input components remaining.

```
>> Rxxtt=Ct*Ruutt*Ct'            (input in pass space)
    0.6667    0.3333   -0.3333
    0.3333    0.6667    0.3333
   -0.3333    0.3333    0.6667
>> rank(Rxxtt) =     2
>> rank(Ruutt) =     2
>> rank(Ct)  =     2
>> [At, OA, Ruupp] = fixin(Ruutt, Ct)
At =      (same as Ct)
    0.6667    0.3333
    0.3333    0.6667
   -0.3333    0.3333
OA =  2 x 0 empty double matrix (happens: no more singularity)
Ruupp =  (looks same as Ruutt)
    2.0000   -1.0000
   -1.0000    2.0000
```

Thus, instead the example proceeds with the alternative input autocorrelation matrix and modulator:

$$R_{\boldsymbol{xx}}=\begin{bmatrix}5/6 & -1/3 & 5/6\\ -1/3 & 4/3 & -1/3\\ 5/6 & -1/3 & 5/6\end{bmatrix}\quad, \tag{5.58}$$

which forms from some $\boldsymbol{u}$ as $\boldsymbol{x}=C\cdot\boldsymbol{u}$, where the input $\boldsymbol{u}$ has 3×3 autocorrelation matrix

$$R_{\boldsymbol{uu}}=\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 2\end{bmatrix}\quad, \tag{5.59}$$

and[19]

$$C=\begin{bmatrix}\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}}\\ \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}}\\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}}\end{bmatrix}\quad. \tag{5.60}$$

The following matlab commands eliminate the channel singularity and then also eliminate channel input singularity for the above example.

```
>> [F,L,M]=svd(H_NW);
M =
   -0.4082    0.7071    0.5774
   -0.8165   -0.0000   -0.5774
   -0.4082   -0.7071    0.5774
 >>M = -M;  F=-F; \% for appearance
```

Inspecting the values in this matrix shows that indeed the C matrix above is the same as this M:

```
 >> C=[1/sqrt(6) -1/sqrt(2) -1/sqrt(3)
sqrt(2/3) 0 1/sqrt(3)
1/sqrt(6) 1/sqrt(2) -1/sqrt(3) ]
    0.4082   -0.7071   -0.5774
    0.8165    0.0000    0.5774
    0.4082    0.7071   -0.5774
```

Since the last column corresponds to channel singularity, for illustration, 2 units of energy are placed upon it, while one of the channel pass-space modes is zeroed:

[19] This C was found by zeroing one of the two pass-space energies and putting its energy in the null space. There are many other ways to construct a modulator and $R_{\boldsymbol{uu}}$ to start this example; this is just one way. The matrix M from the channel's SVD was used for C in this case.

```
>> Ruu=[1 0 0
0 0 0
0 0 2];
>> Rxx=[ 5/6 -1/3 5/6
-1/3 4/3 -1/3
5/6 -1/3 5/6]
    0.8333   -0.3333    0.8333
   -0.3333    1.3333   -0.3333
    0.8333   -0.3333    0.8333
>> C*Ruu*C \% checks
    0.8333   -0.3333    0.8333
   -0.3333    1.3333   -0.3333
    0.8333   -0.3333    0.8333
>> [Ct, Ot, Ruutt] = fixmod(H_NW, Ruu, M)
Ct =
    0.4082   -0.7071
    0.8165    0.0000
    0.4082    0.7071
Ot =   1.0e-15 *  (Ot is zero)
    0.1473
    0.0196
Ruutt =
    1.0000    0.0000
    0.0000    0.0000
>> [At, OA, Ruupp] = fixin(Ruutt, Ct)
At =
    0.4082
    0.8165
    0.4082
OA =   9.9148e-16  (zero)
Ruupp =   1.0000
-------------------- see new input to channel with null-space and zero energy removed --------------
>> Rxxtt=At*At' =
    0.1667    0.3333    0.1667
    0.3333    0.6667    0.3333
    0.1667    0.3333    0.1667
>> trace(Rxxtt) =    1.0000
>> trace(Rxx) =     3
```

First from $R_{\boldsymbol{uu}}$, it is clear that input dimension 2 has zero energy so the $\tilde{C}$ is for inputs u_1 and u_3 (which were the linearly independent columns that were maintained):

$$\tilde{x}_3 = \frac{u_3}{\sqrt{6}} - \frac{u_1}{\sqrt{2}} \tag{5.61}$$

$$\tilde{x}_2 = \frac{\sqrt{2} \cdot u_3}{\sqrt{3}} \tag{5.62}$$

$$\tilde{x}_1 = \frac{u_3}{\sqrt{6}} + \frac{u_1}{\sqrt{2}} \ . \tag{5.63}$$

The input autocorrelation matrix $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}$ is singular but the zero energy on one of the pass-space dimensions is addressed above by the fixin.m command leaving

$$x'_3 = \frac{u_3}{\sqrt{6}} \tag{5.64}$$

$$x'_2 = \frac{\sqrt{2} \cdot u_3}{\sqrt{3}} \tag{5.65}$$

$$x'_1 = \frac{u_3}{\sqrt{6}} \ , \tag{5.66}$$

which is a one-dimensional input after removal of all singularity. The last matlab steps above illustrate $R_{\boldsymbol{x}'\boldsymbol{x}'}$ and its energy (trace) being less by 2 units of energy that were lost into null space from the original $R_{\boldsymbol{xx}}$ that had only 1 unit of energy on a channel input singular vector that passed into the channel.

Finall the overall channel is

```
>> H_NW=[1 1 0
0 1 1];
>> H_NW*At  =
```

```
   1.2247
   1.2247
>> At'*H_NW'*H_NW*At =    3.0000
```

so the matched filter output is 3 times the input u_3. Equivalently, this 3×4 channel for the given input $R_{\boldsymbol{xx}}$ is equivalent to a simple single scalar AWGN.

5.1.2 Designing A, the GDFE's Discrete Modulator

Given an input $\boldsymbol{u}'$ after Subsection 5.1.1's two-step singularity removal, this subsection and all ensuing chapters and sections reset this input's name simply to $\boldsymbol{u}$ to avoid the use of primes and tildes. (The primes and tildes are necessary in singularity elimination for notational bookkeeping purposes, but their use need no longer continue.) The modulator input $\boldsymbol{u}$ with corresponding possibly non-diagonal $R_{\boldsymbol{uu}}$ is a better design start point than $\boldsymbol{x}$ but does not yet easily accommodate independent message inputs. This subsection progresses the discrete modulator to such a design-friendly input that allows separation of independent-symbol encoding from modulation.

Canonical Forward Channel: Again, the GDFE development generalizes to Gaussian noise vectors $\boldsymbol{n}$ that do not necessarily have a diagonal autocorrelation matrix through

$$R_{\boldsymbol{nn}} = R_{\boldsymbol{nn}}^{1/2} \cdot R_{\boldsymbol{nn}}^{*/2} \quad , \tag{5.67}$$

where $R_{\boldsymbol{nn}}^{1/2}$ is any (matrix square roots[20] are not usually unique) square root of $R_{\boldsymbol{nn}}$. The equivalent "white-noise" channel $\widetilde{H}$ forms through noise whitening, $\widetilde{H} = R_{\boldsymbol{nn}}^{-1/2} \cdot H$. An additional step is possible because $R_{\boldsymbol{uu}}$ is now nonsingular, through any $R_{\boldsymbol{uu}}$ square root Φ such that

$$R_{\boldsymbol{uu}} = \Phi \cdot \Phi^* \quad , \tag{5.68}$$

where the input $\boldsymbol{u}$ now has relationship

$$\boldsymbol{u} = \Phi \cdot \boldsymbol{v} \tag{5.69}$$

with some white input $\boldsymbol{v}$ where $R_{\boldsymbol{vv}} = I$. Then the channel becomes

$$\tilde{\boldsymbol{y}} = R_{\boldsymbol{nn}}^{-1/2} \cdot \boldsymbol{y} = R_{\boldsymbol{nn}}^{-1/2} \cdot (H \cdot \underbrace{\tilde{A} \cdot \Phi}_{A} \cdot \boldsymbol{v} + \boldsymbol{n}) = \widetilde{H} \cdot \boldsymbol{v} + \tilde{\boldsymbol{n}} \quad , \tag{5.70}$$

where $\widetilde{H} = R_{\boldsymbol{nn}}^{-1/2} \cdot H \cdot \tilde{A} \cdot \Phi$, and $\tilde{\boldsymbol{n}}$ is white noise with unit variance per (possibly complex) dimension. Figure 5.5 illustrates this noise whitening, and the consequent noise-whitened channel matrix $\widetilde{H}$ can undergo all singularity elimination (channel and input).

Definition 5.1.1 [The GDFE discrete modulator] *A GDFE discrete modulator satisfies*

$$A = \tilde{A} \cdot \Phi \quad , \tag{5.71}$$

where $\tilde{A}$ is any set of linearly independent columns of any valid symmetric decomposition of $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} = \tilde{A} \cdot R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}} \tilde{A}$, and Φ is any valid decomposition of the resultant $R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}} = \Phi \cdot \Phi^*$.

The input to the GDFE discrete modulator is where designers input codeword subsymbols from a good AWGN-code encoder. The filtered noise $\tilde{\boldsymbol{n}}$ remains independent of the input $\boldsymbol{u}$. By the sufficiency of matched filtering[21], the receiver may process the received signal by a set of matched-filter vectors

[20] The matlab command sqrtm finds a symmetric-matrix square root such that $R^{1/2} \cdot R^{1/2} = R$. The matlab chol command finds a triangular square root such that $R^{1/2} \cdot R^{*/2} = R$.

[21] This matched-filter matrix is a one-to-one mapping between the signal components of $\boldsymbol{x}'$ and $\boldsymbol{z}$. The noise after filtering is white and so any components on other dimensions are irrelevent by Chapter 1's Theorems on Irrevelance and Reversibility..

(the rows of $\widetilde{H}^*$) without information loss to obtain Appendix D's MMSE **canonical forward channel model**:

$$\boldsymbol{z} = \widetilde{H}^* \cdot \widetilde{H} \cdot \boldsymbol{v} + \widetilde{H}^* \cdot \tilde{\boldsymbol{n}} = R_f \cdot \boldsymbol{v} + \boldsymbol{n}' \quad , \tag{5.72}$$

where R_f is the canonical forward-channel matrix $R_f = \widetilde{H}^* \cdot \widetilde{H}$ and $R_{\boldsymbol{n}'\boldsymbol{n}'} = R_f$. This forward channel is canonical (as per Appendix D) because the channel shaping R_f is the same as the noise autocorrelation or shaping, also R_f. Because the noise vector $\boldsymbol{n}$ is uncorrelated with the input vector $\boldsymbol{u}$, a right triangle can be drawn to illustrate Equation (5.72) as in Figure 5.8 (on the left). This triangle satisfies a "Pythagorean Theorem"

$$R_{\boldsymbol{z}\boldsymbol{z}} = R_f \cdot R_{\boldsymbol{v}\boldsymbol{v}} \cdot R_f + R_{\boldsymbol{n}'\boldsymbol{n}'} = R_f^2 + R_f \quad , \tag{5.73}$$

observing that $R_f^* = R_f$. Also, because canonical channel noise $\boldsymbol{n}'$ is uncorrelated with the channel input $\boldsymbol{u}$, then (via Appendix D's Orthogonality Principle)

$$\hat{\boldsymbol{z}} = R_f \cdot \boldsymbol{v} \quad , \tag{5.74}$$

is the MMSE vector estimate of $\boldsymbol{z}$ given $\boldsymbol{v}$. Thus, another expression for R_f directly invoking Appendix D's Orthogonality Principle, is

$$R_f = R_{\boldsymbol{z}\boldsymbol{v}} \cdot R_{\boldsymbol{v}\boldsymbol{v}}^{-1} = R_{\boldsymbol{z}\boldsymbol{v}} \quad . \tag{5.75}$$

The MMSE matrix for this forward-channel estimation problem is then clearly $R_{\boldsymbol{n}'\boldsymbol{n}'} = R_f$. (This also implies that $R_{\boldsymbol{z}\boldsymbol{v}} = R_{\boldsymbol{v}\boldsymbol{z}}^*$, or $R_{\boldsymbol{z}\boldsymbol{v}}$ is conjugate symmetric.)

Canonical Backward Channel: Good design desires a canonical MMSE estimate of $\boldsymbol{v}$ from $\boldsymbol{z}$, where $\boldsymbol{u} = \Phi \cdot \boldsymbol{v}$; this MMSE estimate arises through the the **canonical backward channel model:**

$$\boldsymbol{v} = R_b \cdot \boldsymbol{z} + \boldsymbol{e} \quad , \tag{5.76}$$

where R_b is the MMSE matrix estimator (equalizer)

$$R_b = R_{\boldsymbol{v}\boldsymbol{z}} \cdot R_{\boldsymbol{z}\boldsymbol{z}}^{-1} \quad . \tag{5.77}$$

Singularity removal insures that R_b^{-1} and R_f^{-1} both exist ($|R_b| > 0$ and $|R_f > 0|$)[22]. The vector $\boldsymbol{e}$ is the MMSE error vector for the input $\boldsymbol{v}$'s MMSE estimate. given $\boldsymbol{z}$. This backward channel is a matrix linear equalizer. The dual Pythagorean relationship on Figure 5.8's right is

$$R_{\boldsymbol{e}\boldsymbol{e}} = R_{\boldsymbol{v}\boldsymbol{v}} - R_{\boldsymbol{v}\boldsymbol{z}} \cdot R_{\boldsymbol{z}\boldsymbol{z}}^{-1} \cdot R_{\boldsymbol{z}\boldsymbol{v}} = I - R_b \cdot R_{\boldsymbol{z}\boldsymbol{z}} \cdot R_b \quad . \tag{5.78}$$

A related error vector is the MMSE estimate of $\boldsymbol{u}$ or $\boldsymbol{e}' \triangleq \Phi \cdot \boldsymbol{e}$ by linearity of MMSE and

$$R_{\boldsymbol{e}'\boldsymbol{e}'} = \Phi \cdot R_{\boldsymbol{e}\boldsymbol{e}} \cdot \Phi^* \quad . \tag{5.79}$$

[22] However a simpler insurance is simply $|R_{\boldsymbol{n}\boldsymbol{n}}| > 0$, but that may not always be true for some special GDFE cases that include, for instance, Section 2.8's worst-case noise that can be singular.

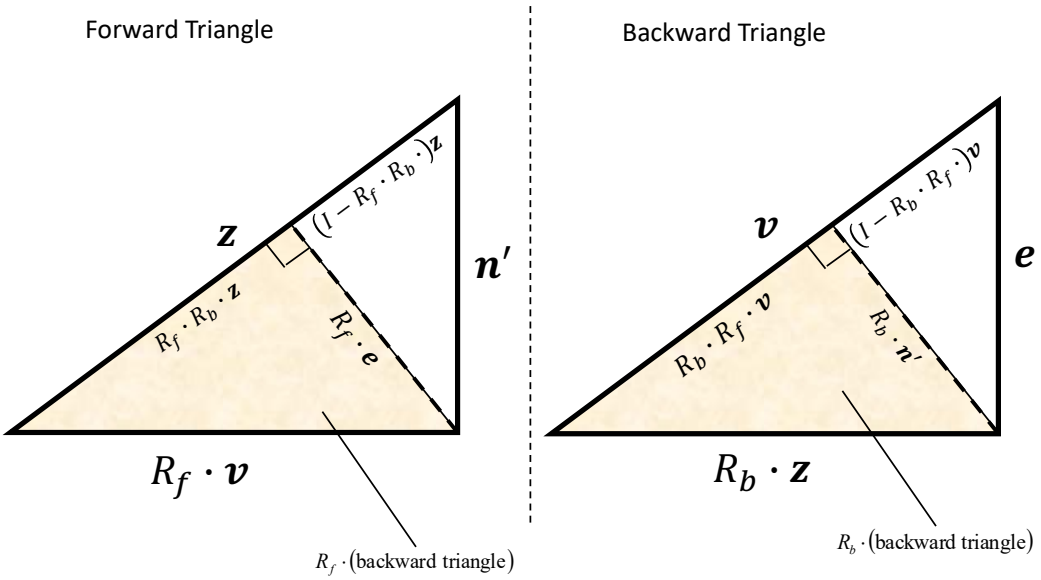


Figure 5.8: Canonical triangles.

A very useful alternative expression for R_b is (recalling that $R_{\boldsymbol{vz}}$ is square nonsingular because of the earlier singularity removal).

$$\begin{aligned} R_b &= R_{\boldsymbol{vz}} \cdot R_{\boldsymbol{zz}}^{-1} && (5.80)\\ R_b^{-1} &= R_{\boldsymbol{zz}} \cdot R_{\boldsymbol{zv}}^{-*} && (5.81)\\ R_b^{-1} \cdot R_{\boldsymbol{vv}} &= R_{\boldsymbol{zz}} \cdot R_{\boldsymbol{zv}}^{-*} \cdot R_{\boldsymbol{vv}} && (5.82)\\ R_b^{-1} &= R_{\boldsymbol{zz}} \cdot R_f^{-1} && (5.83)\\ &= \left(R_f^2 + R_f\right) \cdot R_f^{-1} && (5.84)\\ R_b^{-1} &= R_f + I && (5.85) \end{aligned}$$

which allows computation of R_b from the forward channel $R_f = \widetilde{H}^* \cdot \widetilde{H}$ and the given transmit autocorrelation matrix $R_{\boldsymbol{vv}} = I$. Equivalently,

$$\begin{aligned} I &= R_b \cdot R_f + R_b = R_f \cdot R_b + \cdot R_b && (5.86)\\ R_b &= I - R_f \cdot R_b && (5.87)\\ &= I - R_b \cdot R_f \ , && (5.88) \end{aligned}$$

or equivalently R_b and R_f commute, since I is symmetric. These MMSE-based R_b and R_f are the same as Chapter 2's versions for the MAC and BC user inputs because those were also the MMSE estimators.

Canonical Triangles: Using this convenient result (5.88) in Equation 5.78

$$\begin{aligned} R_{\boldsymbol{ee}} &= R_{\boldsymbol{vv}} - R_{\boldsymbol{vz}} \cdot R_{\boldsymbol{zz}}^{-1} \cdot R_{\boldsymbol{zv}} = I - R_{\boldsymbol{vz}} \cdot R_{\boldsymbol{zz}}^{-1} \cdot R_{\boldsymbol{zv}} && (5.89)\\ &= I - R_{\boldsymbol{vv}}^{-1} \cdot R_{\boldsymbol{vz}} \cdot R_{\boldsymbol{zz}}^{-1} \cdot R_{\boldsymbol{zv}} && (5.90)\\ &= I - R_b \cdot R_f && (5.91)\\ &= R_b \text{ (uses Equation (5.88))} && (5.92)\\ \frac{|\, R_{\boldsymbol{ee}} \,|}{|\, R_{\boldsymbol{vv}} \,|} &= |I - R_b \cdot R_f| = |R_{\boldsymbol{ee}}| \ , && (5.93) \end{aligned}$$

an inverse signal to-MMSE ratio. The use of the two triangles allows derivation of any backward relation from any forward relation by swapping R_f and R_b as well as replacing $R_{\boldsymbol{ee}}$ by $R_{\boldsymbol{n'n'}}$ (essentially then $\boldsymbol{z} \leftrightarrow \boldsymbol{u}$, $\boldsymbol{n'} \rightarrow \boldsymbol{e}$). The following relations use Figure 5.8's forward and backward channels' duality: (**with** $R_{\boldsymbol{vv}} = I$ **ONLY**)

$$\begin{aligned}
R_b^{-1} &= R_f + R_{\boldsymbol{vv}}^{-1} = R_f + I && (5.94)\\
R_f^{-1} &= R_b + R_{\boldsymbol{zz}}^{-1} && (5.95)\\
R_{\boldsymbol{ee}} &= R_b && (5.96)\\
R_{\boldsymbol{n'n'}} &= R_f && (5.97)\\
R_{\boldsymbol{zz}} &= R_f \cdot R_{\boldsymbol{vv}} \cdot R_f + R_f = R_f \cdot R_b^{-1} && (5.98)\\
R_{\boldsymbol{vv}} &= R_b \cdot R_{\boldsymbol{zz}} \cdot R_b + R_b = R_b \cdot R_{\boldsymbol{zz}} \cdot R_f^{-1} && (5.99)\\
R_{\boldsymbol{zv}} &= R_f \cdot R_{\boldsymbol{vv}} = R_f && (5.100)\\
R_{\boldsymbol{vz}} &= R_b \cdot R_{\boldsymbol{zz}} \quad . && (5.101)
\end{aligned}$$

Because the signal part of $\boldsymbol{z}$ is in one-to-one correspondence with $\boldsymbol{v}$ as a result of the careful elimination of unnecessary dimensionality through the matrix A,

$$\mathcal{I}(\boldsymbol{v};\boldsymbol{z}) = \mathcal{I}(\boldsymbol{v};\boldsymbol{y}) = \mathcal{I}(\boldsymbol{x};\boldsymbol{y}) \quad , \tag{5.102}$$

so that the new channel between $\boldsymbol{v}$ and $\boldsymbol{z}$ carries all the relevant information between channel input and output. The backward and forward canoncial channels both have the same mutual information. Transmission from $\boldsymbol{v}$ to $\boldsymbol{z}$ has the same performance as $\boldsymbol{z}$ to $\boldsymbol{v}$, and as from $\boldsymbol{x}$ to $\boldsymbol{y}$. However, the backward channel next leads to a canonical and simpler receiver for $b \leq \mathcal{I}(\boldsymbol{x};\boldsymbol{y})$, the GDFE.

5.1.3 Generalized Decision Feedback Development

The GDFE exploits the backward canonical model's structure. The backward canonical model is

$$\boldsymbol{v} = R_b \cdot \boldsymbol{z} + \boldsymbol{e} \quad , \tag{5.103}$$

where R_b is now nonsingular. The inverse MMSE R_b^{-1} has an lower-diagonal-upper Cholesky factorization[23]

$$R_b^{-1} = G^* \cdot \boldsymbol{S}_0 \cdot G \quad , \tag{5.104}$$

where G is upper triangular and monic (ones along the diagnonal), and $\boldsymbol{S}_0$ is a diagonal matrix of positive-real Cholesky factors. Subsections 5.1.1.1 and 5.1.1.2's channel and input singularity removal ensure that R_f and R_b are nonsingular and generally advisable. However only $R_b \succeq 0$ must be nonsingular lo obtain $[G \;\; S_0]$; this creates a design shortcut when $R_{\boldsymbol{nn}} \succeq 0$. So, the singularithy removal becomes superfluous with nonsingular noise. Chapter 2's BC may have "worst-case" noise, which is an example that will require input singularity removal for an appropriate GDFE to exist. Then factoring $R_{\boldsymbol{ee}} = R_b$:

$$R_b = G^{-1} \cdot \boldsymbol{S}_0^{-1} \cdot G^{-*} \quad . \tag{5.105}$$

Premultiplication of $\boldsymbol{v}$ in (5.103) by G produces

$$\boldsymbol{v}' = G \cdot \boldsymbol{v} = \boldsymbol{S}_0^{-1} \cdot G^{-*} \cdot \boldsymbol{z} + \boldsymbol{e}' = \boldsymbol{z}' + \boldsymbol{e}' \quad , \tag{5.106}$$

where $\boldsymbol{e}' = G \cdot \boldsymbol{e}$ has diagonal autocorrelation matrix $\boldsymbol{S}_0$, and $\boldsymbol{z}'$ is the matrix feedforward-filter output $\boldsymbol{S}_0^{-1} \cdot G^{-*} \cdot \boldsymbol{z}$. Again, mutual information remains the same because the triangular matrices G and S_0 are one-to-one, so

$$\mathcal{I}(\boldsymbol{v};\boldsymbol{z}') = \mathcal{I}(\boldsymbol{x};\boldsymbol{y}) \quad . \tag{5.107}$$

[23] See Appendix D; Cholesky Facorization of an autocorrelation matrix $R_{\boldsymbol{xx}}$ is the matrix or finite-length-packet equivalent of spectral factorization of the infinite-length autocorrelation function $R_x(D)$.

Back Substitution: The receiver recovers the channel input vector $\boldsymbol{v}$ from

$$\boldsymbol{z}' = G \cdot \boldsymbol{v} - \boldsymbol{e}' \tag{5.108}$$

by a process known as back substitution, as also used by Chapter 2's MAC: A detailed matrix description of (5.108) is

$$\begin{bmatrix} z'_{\overline{N}^*-1} \\ z'_{\overline{N}^*-2} \\ \vdots \\ z'_0 \end{bmatrix} = \begin{bmatrix} 1 & g_{\overline{N}^*-1,\overline{N}^*-2} & \cdots & g_{\overline{N}^*-1,0} \\ 0 & 1 & \cdots & g_{\overline{N}^*-2,0} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{\overline{N}^*-1} \\ v_{\overline{N}^*-2} \\ \vdots \\ v_0 \end{bmatrix} - \begin{bmatrix} e'_{\overline{N}^*-1} \\ e'_{\overline{N}^*-2} \\ \vdots \\ e'_0 \end{bmatrix} , \tag{5.109}$$

where each symbol's first time dimension is at the bottom, following Chapters 2 and 4[24]. By starting at the bottom of (5.109) and ascending, back-substitution solution of (5.109) obtains the decision-feedback equations:

$$\begin{aligned} \hat{v}_0 &= \text{dec}_0(z'_0) && (5.110)\\ \hat{v}_1 &= \text{dec}_1(z'_1 - g_{1,0} \cdot \hat{v}_0) && (5.111)\\ \hat{v}_2 &= \text{dec}_2(z'_2 - g_{2,1} \cdot \hat{v}_1 - g_{2,0} \cdot \hat{v}_0) && (5.112)\\ \vdots &= \vdots && (5.113)\\ \hat{v}_{\overline{N}^*-1} &= \text{dec}_{\overline{N}^*-1}(z'_{\overline{N}^*-1} - \sum_{i=0}^{\overline{N}^*-2} g_{\overline{N}^*-1,i} \cdot \hat{v}_i) && (5.114) \end{aligned}$$

The back-substitution solution is not a maximum-likelihood detector: but because $R_{\boldsymbol{vv}} = I$ is diagonal, it can reliably detect outputs at $b = \mathcal{I}(\boldsymbol{x};\boldsymbol{y})$ as long as error propagation for an actual systems dimensional detectors is negligible. With $\Gamma = 0$ dB gap, it may be that many (infinite) successive symbols for dimension $n = 0$ must decode ("dec") first before any of the results can be used for dimension $n = 1$, and so on. This can introduce several subsymbol delays in decoding. In practice, Chapter 4's separation theorem with the same code and constellation on all dimensions reduces this delay.

ComputeGDFE Program: Appendix G's computeGDFE.m program can be used, but computes a SNR according to the size of the input square-root matrix A, unless a last optional input appears that is an alternative symbol size.

```
function [snrGDFEu, GU, WU, SO, MSWMFU, b, bbar, snrGLEu] = computeGDFE(H, A, cb, Lx)
--------------------------------------------------------------------------
 Inputs
 H: noise-whitened channel, Ly x Lx (one tone)
 A: any Lx x Lx square root of input autocorrelation matrix
        This can be generalized non-square square-root
 cb: =1 if H is complex baseband; =2 if H is real baseband
 Lx: optional input of Lx when not equal to size of A
     this is used to compute bits/dimension properly
 Outputs
 GU: unbiased feedback matrix
 WU: unbiased feedforward linear equalizer
 SO: sub-channel channel gains
 MSWMFU: unbiased mean-squared whitened matched filter
 b: bit distribution vector
 bbar: number of bits/dimension (real if cb=2, complex if cb=1)
 snrGDFEu - unbiased SNR in dB; assumes size of R_sqrt input
    the user should recompute SNR if there is a cyclic prefix
 b = bit distribution over symbol dimensions (real cb=2; complex cb=1)
 bbar is sum(b)/Lx so total bits/(real cb=2; cplx cb=1) dimension
 snrGLEu is the linear GLE SNR
 Note: The R_sqrt need not be square non-singular, as long as
 R_sqrt*R_sqrt' = input autocorrelation matrix Rxx, but SNRLEu is off
Thanks to Ethan Liang, corrected/updated J. Cioffi
```

[24] This should not be confused with the singularity-elimination process that simply described blocks of vectors within a symbol

See upcoming Example 5.1.2 for example use. The last optional argument snrGLEu is for a MMSE LE (so no feedback allowed) best SNR. It is never larger than the GDFE SNR and usually smaller, often significantly so.

Suboptimal Detection: The "dec_n" operation can be simple symbol-by-symbol "uncoded" detection for the constellation C_n on the n^{th} transmitted subsymbol $u_n, n = 0, ..., N^* - 1$ (in which case $\Gamma > 0$ dB). Any error made likely causes other subsequent $\text{dec}_{i>n}$ errors, but the error propagation arrests when n attains $\overline{N}^*$ (unlike Chapter 3's MMSE-DFE potential infinite-length error bursts). The feedback "filters" are a function of the dimensional index n and represent a periodic dimension-variable[25] feedback section. The period is the symbol period. Similarly, the feedforward section is also periodically dimension-varying because it is a triangular matrix multiply for which the rows are not simple shifts of each other. This receiver is called the "Generalized DFE" or GDFE, and a GDFE block diagram appears in Figure 5.9. The n^{th} tone's MSE is

$$\mathcal{E}[|e'_n|^2] = S_{0,n}^{-1} = \frac{1}{S_{0,n}} \quad . \tag{5.115}$$

The GDFE creates a set of parallel AWGNs with (biased) SNR's equal to

$$\text{SNR}_{bias,n} = 1 + \text{SNR}_{v,n} = \frac{\mathcal{E}[|v_n|^2]}{\mathcal{E}[|e'_n|^2]} = 1 \cdot S_{0,n} \tag{5.116}$$

where this expression is similar to Chapter 3's expression $\text{SNR}_{\text{MMSE-DFE}} = \gamma_0 \cdot \frac{\bar{\mathcal{E}}_{\boldsymbol{x}} \cdot \|h\|^2}{\frac{N_0}{2}}$ with a dimensionally variant $S_{0,n}$ replacing the infinite-length MMSE-DFE's dimensionally invariant $\gamma_0 \cdot \frac{\|h\|^2}{\frac{N_0}{2}}$, which derives from Chapter 3's canonical factorization when the channel, noise, and input processes are stationary. The product of these biased SNR's is then

$$\text{SNR}_{\text{GDFE}} = \left(\prod_{n=1}^{\bar{N}_x^*} \text{SNR}_{bias,n} \right)^{\frac{1}{\bar{N}_x}} = \frac{|R_{\boldsymbol{vv}}|^{1/\bar{N}_x}}{|R_{\boldsymbol{e'e'}}|^{1/\bar{N}_x}} = |R_{\boldsymbol{ee}}|^{-\frac{1}{\bar{N}_x}} = 2^{2 \cdot \bar{\mathcal{I}}(\boldsymbol{v}, \boldsymbol{z}')} = 2^{2 \cdot \bar{\mathcal{I}}(\boldsymbol{x}, \boldsymbol{y})} \quad , \tag{5.117}$$

(5.117) is a finite-length CDEF result (see Chapter 3 for the infinite-length version). A suboptimal detector, namely the GDFE can exhibit an SNR on an equivalent set of independent AWGN channels that is equal to the best achievable for such a set for independent input-dimensional components. This surprising result occurs only for the backward canonical channel model. For this symbol-length, the same good codes that achieve capacity on the ISI-free channel can also do so reliably with the GDFE system; further the same-gap codes can nearly achieve the same gap-reduced data rate with the GDFE. If $R_{\boldsymbol{x'x'}}$ is the same as that generated by water-filling on the $\overline{N}^*$ discrete subchannels of $\widetilde{H}$ that vector coding would use, then the GDFE's reliably decoded data rate approaches $\mathcal{C}$, just as closely as with the Chapter 4's VC, even though the detector is suboptimum. Only a "white input" (diagonal $R_{\boldsymbol{vv}} = I$) attains this canonical performance level with the suboptimal detector. Furthermore, following Chapter 2's arguments on shaping gain, GDFE use on good codes built from square constellations (like PAM or QAM subsymbols) so that $\Gamma \to \Gamma - \gamma_s = \Gamma - 1.53$ dB will retain their goodness and just lose the 1.53 dB of shaping that is possible with best codes, and square subsymbol constellations.

For Matrix ISI Channel Models $\widetilde{H}$: The GDFE replaces Chapter 3's MMSE-DFE's linear filters by matrix multiplies, indeed triangular matrix multiplies $(I - G)$ and $S_0 \cdot G^{-*}$ correspond to a causal feedback section and a non-causal feed-forward section, respectively, when H is Toeplitz (stationary channel). The matched filter is the matrix $(R_{\boldsymbol{nn}}^{-1/2} \cdot H \cdot A)^*$. These matrix multiplies can be interpreted as periodically dimensionally varying filters - a different set of coefficients for each index $n = 1, ..., \widetilde{N}^*$ within the packet. The GDFE filters are thus fundamentally different from Chapter 3's finite-length filters in general.

[25] Or more generally, periodically "index-varying." The period is the symbol period.

The GDFE's diagonal-matrix SNR is

$$\boldsymbol{SNR}_{\text{GDFE}} \triangleq R_{\boldsymbol{vv}} \cdot R^{-1}_{\boldsymbol{e'e'}} = R^{-1}_{\boldsymbol{e'e'}} = \boldsymbol{SNR}_{\text{GDFE,unb}} + I \quad , \tag{5.118}$$

and its determinant is $\text{SNR}_{\text{GDFE}} = |\boldsymbol{SNR}_{\text{GDFE}}|$. Thus

$$|\boldsymbol{SNR}_{\text{GDFE}}| = |\boldsymbol{SNR}_{\text{GDFE,unb}} + I| = \prod_{i=0}^{\bar{N}^*-1} (1 + \text{SNR}_{U,i}) \quad . \quad . \tag{5.119}$$

Figure 5.9 illustrates the system.

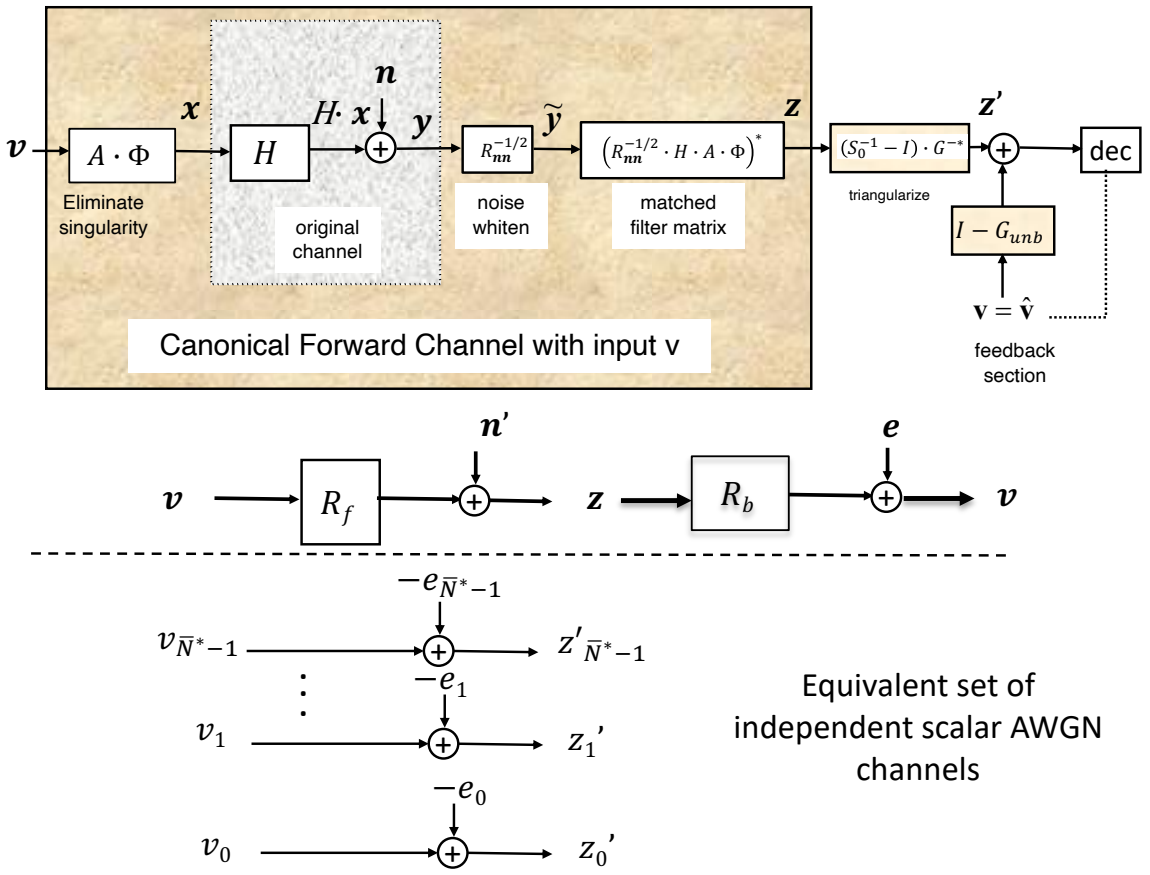


Figure 5.9: The Generalized DFE with white input $\boldsymbol{v}$

Bias: The GDFE has bias like all MMSE estimators, as per Appendix D. Each GDFE output dimension has a MMSE estimate with $S_{0,n} = \text{SNR}_n + 1$. The matrix $\boldsymbol{SNR}_{\text{GDFE}}$ corresponds to a biased estimate. That is

$$\mathbb{E}[\hat{v}_n / v_n] = \left(\frac{S_{0,n} - 1}{S_{0,n}}\right) \cdot v_n \neq v_n \quad . \tag{5.120}$$

Bias can be removed individually after feedback on the subchannel dimensions or by scaling each feedforward-matrix output dimension by $\frac{S_{0,n}}{S_{0,n}-1}$. Also

$$G_{unb} = I + S_0 \cdot (S_0 - I)^{-1} \cdot [G - I] \tag{5.121}$$

$$W_{unb} = (S_0 - I)^{-1} \cdot G^{-*} \tag{5.122}$$

$$MSWMF_{unb} = (S_0 - I)^{-1} \cdot G^{-*} \cdot A^* \cdot H^* \cdot R^{-1/2}_{\boldsymbol{nn}} \quad . \tag{5.123}$$

Reconstructing $\hat{\boldsymbol{u}}$: After the GDFE estimates $\hat{\boldsymbol{v}}$, then the corresponding MMSE estimate of $\boldsymbol{u}$ is $\hat{\boldsymbol{u}} = \Phi \cdot \hat{\boldsymbol{v}}$, and in turn $\hat{\boldsymbol{x}}' = A \cdot \hat{\boldsymbol{v}}$,because MMSE estimates are linear. Any part of $\boldsymbol{x}$ not contained in $\boldsymbol{x}'$ is lost in transmission. When $\boldsymbol{x} = \boldsymbol{x}'$, there is no such loss and $\boldsymbol{u}$ and $\boldsymbol{v}$ are in 1-to-1 correspondence (as long as the GDFE symbol-by-symbol detectors function without significant error in estimation). Thus $A = \tilde{A} \cdot \Phi$ is a canonical matrix transmit filter or discrete modulator. Chapter 4's separation theorem still applies if $\Gamma = 0$ dB good codes use a (large) constellation of constant size, and loading simplifies to determining constellation size and code rate (with no error propagation since $\Gamma = 0$ dB).

Section 5.3 investigates two specific structures for the matrix Φ and finds that a triangular factorization can lead to a MMSE-DFE finite-length equivalent. The other structure, an eigen-decomposition, leads to VC. There can be many such transformations and each corresponds to a different GDFE, but all attain the same performance. The following example illustrates one such case:

EXAMPLE 5.1.2 *[GDFE for $1 + .9 \cdot D^{-1}$ Channel]* This example revisits Chapters 3's and 4's $1 + .9 \cdot D^{-1}$ channel with input energy $\bar{\mathcal{E}}_{\boldsymbol{x}} = 1$ ($R_{\boldsymbol{xx}} = I = R_{\boldsymbol{uu}}$) and $\sigma^2 = .181$. This current-example instance investigates a GDFE for this channel with $N = 2$ and $\nu = 1$, so $N_x = 3$. The matrix $R_{\boldsymbol{nn}}^{-1/2} \cdot H$ for this channel is

$$\frac{1}{\sigma} \cdot H = \frac{1}{\sqrt{.181}} \cdot \begin{bmatrix} .9 & 1 & 0 \\ 0 & .9 & 1 \end{bmatrix} = \begin{bmatrix} 2.1155 & 2.3505 & 0 \\ 0 & 2.1155 & 2.3505 \end{bmatrix} \quad , \tag{5.124}$$

with singular value decomposition:

$$\frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 3.8694 & 0 & 0 \\ 0 & 2.2422 & 0 \end{bmatrix} \begin{bmatrix} .3866 & .6671 & .6368 \\ .8161 & .0741 & -.5731 \\ .4295 & -.7412 & .5158 \end{bmatrix}^* \quad . \tag{5.125}$$

The first two columns of M span $\widetilde{H}$'s pass space $\mathcal{P}_{\widetilde{H}}$. The original PAM modulator corresponds to $C = I$, so

$$\tilde{C}_{temp} = P_{\widetilde{M}} \cdot I = \begin{bmatrix} .5945 & .3649 & -.3285 \\ .3649 & .6715 & .2956 \\ -.3285 & .2956 & .7340 \end{bmatrix} \quad , \tag{5.126}$$

Then $\tilde{C}$ is the leftmost 2 columns of $\tilde{C}_{temp}$. This special case of white PAM modulation on $\boldsymbol{x}$ corresponds to a rank-2

$$R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} = P_{\widetilde{M}} \cdot I \cdot P_{\widetilde{M}} \quad . \tag{5.127}$$

Since $\varrho_x = 2$, then $\varrho_x = \varrho_H$ and no further singularity elimination is necessary. However, $R_{\boldsymbol{uu}}$ is not diagonal. By using the steps in Figure 5.5,

$$R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}} = \begin{bmatrix} 2.5242 & -1.3717 \\ -1.3717 & 2.2346 \end{bmatrix} \quad , \tag{5.128}$$

and

$$\tilde{u}_1 = x_1 - 1.235 \cdot x_3 \tag{5.129}$$

$$\tilde{u}_2 = x_2 + 1.111 \cdot x_3 \tag{5.130}$$

In matlab, the commands are:
ELIMINATE CHANNEL SINGULARITY:

```
>> H=(1/sqrt(.181))*[.9 1 0
0 .9 1];
>> [Ct, Ot, Ruutt] = fixmod(H, eye(3), eye(3))
Ct =
    0.5945    0.3649
    0.3649    0.6715
   -0.3285    0.2956
Ot =
   -1.2346
    1.1111
Ruutt =
    2.5242   -1.3717
   -1.3717    2.2346
```

The $R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}}$ is not white. To find a white input, the designer can use several square root forms for the 2×2 matrix $R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}}$. This $R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}}$ is already nonsingular, the fixin.m program essentially does not further improve, but we show it anyway here:

```
>> [A, OA, Ruupp] = fixin(Ruutt, Ct)
A =
    0.5945    0.3649
    0.3649    0.6715
   -0.3285    0.2956
OA =
  2 x 0 empty double matrix
Ruupp =
    2.5242   -1.3717
   -1.3717    2.2346
```

Specifically, $R_{\boldsymbol{x}'\boldsymbol{x}'} = R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}$ and $\tilde{C} = A$ in this situation. Design can proceed with a choice of square root for $R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}}$, so a first example is Cholesky Factorization of $R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}}$

```
>> Gubar=lohc(Ruupp);
>> Gu=Gubar*inv(diag(diag(Gubar))) ;
>> Xmit=A*Gubar;
>> cb=2;
>> Lx=3;
[snrGDFEu, GU, WU, S0, MSWMFU, b, bbar, ~] = computeGDFE(H, Xmit,cb,Lx)
snrGDFEu =    5.5427 dB
>> GU =
    1.0000    0.5731
         0    1.0000
>> WU =
    0.1328         0
   -0.0492    0.0972
>> S0 =
    8.5275         0
         0   11.2899
>> MSWMFU =
    0.3645    0.0000
    0.0179    0.3073
b =                                  bits/symbol
    1.5461
    1.7485
bbar =    1.0982     bits/dimension (real for this example because cb = 2)
```

The SNR is invariant at 5.5427 dB.

Instead, a symmetric square root might be of interest:

```
>> Gxbar=sqrtm(Ruutt)
>> Gxbar =
    1.5186   -0.4668
   -0.4668    1.4201
>> Xmit=A*Gxbar;
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar,~] = computeGDFE(H, Xmit,cb,Lx);
>> snrGDFEu  =    5.5427 dB
>> GU =
    1.0000    0.3680
         0    1.0000
>> MSWMFU  =
    0.3881   -0.1812
    0.1215    0.2378
>> b' =    1.3447    1.9499
>> bbar =    1.0982
```

The performance is the same, as is the total bits/symbol, but the bit distribution and the implementation are different. Yet another version uses the eigendecomposition of $R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}}$ directly, starting with the A from the fixin program's outputs A and Ruupp.

```
>> [V,D]=eig(Ruutt);
>> Gxbar=V*sqrt(D) =
   -0.6690   -1.4411
   -0.7433    1.2970
>> Xmit=A*Gxbar;
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar] = computeGDFE(H, Xmit,cb,Lx);
>> snrGDFEu =    5.5427 dB
>> GU =
```

```
    1.0000   -0.3459
         0    1.0000
>> MSWMFU =
   -0.2535   -0.1261
   -0.1648    0.3645
>> b' =    1.8760    1.4186
>> bbar =    1.0982
```

A GDFE based on estimating $\tilde{u}_1$ and $\tilde{u}_2$ will therefore not achieve canonical performance because $R_{\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}} \neq I$. However, any of the above $R_{\boldsymbol{vv}} = I$ choices generates a valid GDFE with the same performance.

The discrete modulator generates 3 subsymbol inputs for each symbol on this GDFE. Because $R_{\boldsymbol{nn}} \succeq 0$, $R_b \succeq 0$, and so R_b is directly invertible and "'factorizable." A GDFE based on a direct 3-dimensional singular square-root of the input $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} = G_{\boldsymbol{x}} \cdot G_{\boldsymbol{x}}^*$ (using matlab's symmetric square root, which works for singular positive semidefinite matrices) also achieves canonical performance over 3 dimensions:

```
>> [Ct, Ot, Ruutt] = fixmod(H, eye(3), eye(3));
>> [A, OA, Ruupp] = fixin(Ruutt, Ct);
>> Rxxtt=A*Ruupp*A';
>> Gxtilde=sqrtm(Rxxtt) =
    0.5945    0.3649   -0.3285
    0.3649    0.6715    0.2956
   -0.3285    0.2956    0.7340
>> [snrGDFEu, GU, WU, SO, MSWMFU, b, bbar,~] = computeGDFE(H, Gxtilde,cb,Lx);
>> snrGDFEu =    5.5427 dB
>> GU =
    1.0000       1.1111    0.0000
         0       1.0000    0.9067
         0       0         1.0000
>> MSWMFU =
    0.4727    0.0000
    0.0783    0.3857
   -0.1923    0.4254
>> b' =    1.2264    1.3485    0.7196
>> bbar =    1.0982
```

There are now 3 dimensions carrying data, but the sum of data rates over these 3, or equivalently the subsymbol, are indeed the same as is $\bar{b} \approx 1.1$.

Finally a simple white input or the input $R_{\boldsymbol{xx}}$ also wastes energy (1 unit of 3) and has $trace\{R_{\boldsymbol{xx}}\} = 3\}$ and has the same bit distribution across 3 dimensions. The corresponding GDFE now is:

```
>> [snrGDFEu, GU, WU, SO, MSWMFU, b, bbar,~] = computeGDFE(H, eye(3),cb,Lx)
S>> nrGDFEu =  5.5427 dB
>> GU =
    1.0000    1.1111         0
         0    1.0000    0.9067
         0         0    1.0000
>> MSWMFU =
    0.4727         0
    0.0783    0.3857
   -0.1923    0.4254
>> b' =    1.2264    1.3485    0.7196
>> bbar =    1.0982
```

or if the $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}$ energy is reset to the original 3 units, but now in the channel pass space, so basically increase input energy by $1.5x$ or any equivalent input

$$R_{\boldsymbol{xx}} = \Phi \cdot \Phi^* \quad , \tag{5.131}$$

or $A \rightarrow \sqrt{1.5} \cdot A$. This is the same as creating an input with 3 units of energy on the pass space and none in the null space.

```
>> [snrGDFEu, GU, WU, SO, MSWMFU, b, bbar,~] = computeGDFE(H, sqrt(1.5)*eye(3),cb,Lx)
snrGDFEu =    6.8589 dB
GU =
    1.0000    1.1111         0
```

```
         0    1.0000    0.9578
         0         0    1.0000
WU =
    0.1490         0         0
   -0.1242    0.1284         0
    0.4195   -0.4338    0.5111
S0 =
    7.7127         0         0
         0    8.7872         0
         0         0    2.9565
MSWMFU =
    0.3860         0
    0.0479    0.3327
   -0.1619    0.3474
b =
    1.4736
    1.5677
    0.7819
bbar =    1.2744
---------- or use the old Rxxtt that has null space energy removed ----------
>> trace(Rxxtt) =    2.0000
>> Rxxtt =
    0.5945    0.3649   -0.3285
    0.3649    0.6715    0.2956
   -0.3285    0.2956    0.7340
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar] = computeGDFE(H, sqrt(1.5)*Rxxtt,cb,Lx);
has all same outputs as wwhen input is diagonal.
```

The SNR increases to 6.9dB and the feedforward filter and the combination of feedforward-matched filter complete the matlab commands above. The complexity remains at 1.667/dimension for the minimum 2-dimensional GDFEs above (it increases unnecessarily for the 3-dimensional versions, but they would be less desirable than 2D). From Chapter 3, the finite-length MMSE-DFE required $N_f = 2$ and $N_b = 1$ to get this same performance, leading to a complexity of 3, almost twice that of the GDFE with altered input. Such is often the benefit of transmit optimization – not only does it work better, the receiver simplifies. The two-dimensional energy choice choice approximates water-filling. As $\overline{N}$ becomes large, the ultimate SNR is 8.4 dB for an i.i.d. input. Unfortunately, as $\overline{N}$ grows, the complexity for this GDFE choice will grow as $\overline{N}^2$, which is unnecessarily well above the complexity of DMT (which Chapter 4 shows can achieve 8.8 dB when $\overline{N} = 16$ and complexity is 3.8/dimension). If the design uses a water-fill input, then the large $\overline{N}$ GDFE SNR can also approach 8.8 dB.

Chain rule relationships and order: Chapter 2's mutual-information chain rule

$$\mathcal{I}(\boldsymbol{x}\ \boldsymbol{y}) = \mathcal{I}(x_0; \boldsymbol{y}) + \mathcal{I}(x_1\ \boldsymbol{y}/x_0) + ... + \mathcal{I}(x_n\ \boldsymbol{y}/[x_1 ... x_{n-1}]) + ... + \mathcal{I}(x_{N-1}\ \boldsymbol{y}/[x_1 ... x_{N-2}]) \tag{5.132}$$

relates directly to all GDFE's. Each term in (5.132) corresponds to a MMSE-DFE estimate of x_n given the entire channel symbol output $\boldsymbol{y}$ and the previous values of x_1, ... x_n. This corresponds to the GDFE's series of MMSE estimates that use the channel output and past decisions – when the input sequence has diagonal autocorrelation $R_{\boldsymbol{vv}} = I$; an equivalent chain rule is then

$$\mathcal{I}(\boldsymbol{x}\ \boldsymbol{y}) = \mathcal{I}(v_0; \boldsymbol{y}) + \mathcal{I}(v_1\ \boldsymbol{y}/v_0) + ... + \mathcal{I}(v_n\ \boldsymbol{y}/[v_1 ... v_{n-1}]) + ... + \mathcal{I}(v_{N-1}\ \boldsymbol{y}/[v_1 ... v_{N-2}])\ . \tag{5.133}$$

There are an infinite number of such possible transformations, leading to an infinite number of GDFEs, all with the same $\mathcal{I}$.

The gap applies to MSE, not just to noise. This MSE varies with dimension, so there can be variation when $\Gamma > 0$ dB when the gap-dependent SNR is

$$SNR_{GDFE}(\Gamma) + 1 \triangleq \left\{ \prod_{n=1}^{\bar{N}^*} \left(1 + \frac{SNR_{bias,n} - 1}{\Gamma} \right) \right\}^{1/\bar{N}_x} . \tag{5.134}$$

The product is exactly equal to the original geometric SNR when $\Gamma = 0$ dB; otherwise

$$\log_2 \left(1 + \frac{SNR_{GDFE}(\Gamma)}{\Gamma} \right) \leq \log_2 \left(1 + \frac{SNR_{GDFE}}{\Gamma} \right) , \tag{5.135}$$

so non-zero gap amplifies inaccuracy of energy distribution. This is analogous to Chapter 2's situation where rate-sum became order dependent if gap increased above 0 dB.

Zero-Forcing GDFE's: The zero-forcing GDFE sets zero noise in the MMSE solution and determines G and S_0. There may be a multitude of equivalent MMSE structures that all have the same MMSE performance; however, caution needs to be exercised in assuming all ZF-GDFEs have the same performance with nonzero noise.

For any given GDFE, the ZF-equivalent uses the forward channel model to derive a ZF-GDFE. The forward channel output $\boldsymbol{z}$ has lower-diagonal-upper Cholesky factorization of $R_f = G_f^* \cdot S_f \cdot G_f$; this corresponds to $R_b = R_f$ if $R_{\boldsymbol{nn}} = 0$. Then $\boldsymbol{z}$ is

$$\boldsymbol{z} = G_f^* \cdot S_f \cdot G_f \cdot \boldsymbol{v} + \boldsymbol{n}' \quad . \tag{5.136}$$

The triangular G_f also follows directly from QR factorization of the original channel, $\widetilde{H} = S_f^{1/2} \cdot G_f \cdot Q$ (ignore zeros to the left or below G_f to extract triangular part from the QR factorization if $\widetilde{H}$ is not square and non-singular). The receiver can process $\boldsymbol{z}$ with $S_f^{-1} \cdot G_f^{-*}$ to obtain

$$\boldsymbol{z}' = S_f^{-1} \cdot G_f^{-*} \cdot \boldsymbol{z} = G_f \cdot \boldsymbol{v} + \boldsymbol{n}'' \quad . \tag{5.137}$$

Ignoring the noise $\boldsymbol{n}''$ as if it were zero produces a triangular set of equations (since G_f is upper triangular) and letting "*dec*" represent a decoder for the code on each dimension[26].

$$\begin{aligned}
\hat{v}_0 &= \mathrm{dec}_0(z'_0) && (5.138)\\
\hat{v}_1 &= \mathrm{dec}_1(z'_1 - G_{f,1,0} \cdot \hat{v}_0) && (5.139)\\
\hat{v}_2 &= \mathrm{dec}_2(z'_2 - G_{f,2,1} \cdot \hat{v}_1 - G_{f,2,0} \cdot \hat{v}_0) && (5.140)\\
\vdots &= \vdots && (5.141)\\
\hat{v}_{\bar{N}^*-1} &= \mathrm{dec}_{\bar{N}^*-1}(z'_{\bar{N}^*-1} - \sum_{i=0}^{\bar{N}^*-2} G_{f,N^*-1,i} \cdot \hat{v}_i) \quad . && (5.142)
\end{aligned}$$

Equations (5.138) - (5.142) estimate the transmitted sequence, but the overall SNR (which is unbiased for ZF) is less than or equal to $2^{2\bar{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})} - 1$. This corresponding SNR product is (with 0 dB gap)

$$\mathrm{SNR}_{zf-gdfe} = \left[\prod_{n=1}^{\bar{N}_x} (1 + S_{f,n}) \right]^{1/\bar{N}_x} - 1 \quad . \tag{5.143}$$

Equality to MMSE-GDFE holds under various conditions like the matrix R_f is diagonal (that is $G_f = I$ and there is no feedback matrix) or certain types of noise (worst-case), as will be seen later in Section 5.3. Often, the ZF-GDFE, can have performance very close to the MMSE-GDFE, which is canonical and highest SNR level. The ZF design complexity is less because essentially just one Cholesky or QR factorization is necessary. An ML detector for the ZF-GDFE would achieve the full $\mathcal{I}$, but would be very complex in most situations and is not necessarily equal to (5.138) - (5.142)'s back-substitution process, nor is ZF necessarily canonical.

5.1.3.1 The Generalized Linear Equalizer - GLE

GDFE implementation may impose additional constraints. One oft-encountered constraint is no feedback section (nor nonlinear precoder as in the upcoming Subsection 5.1.4), here called the **Generalized Linear Equalizer (GLE)**. Such a linear-only discrete-modulation and/or discrete demodulation system retains canonical (and optimum) performance if R_b is diagonal, or equivalently $G = I$. One example of such a system is Chapter 4's vector coding, which the next Section 5.2 directly details further.

[26] If uncoded, then an instantaneous decision occurs for each subsymbol/dimension.

More generally, GLE performance will depend on the choice of A and will not be the same for all the different square root choices of $R_{\boldsymbol{xx}}$. Different A choices for the same $R_{\boldsymbol{xx}}$ can provide different GLE performance, unlike the GDFE where the performance is insensitive to the particular square-root choice of $A = R_{\boldsymbol{xx}}^{1/2}$. The linear MMSE estimate of $\boldsymbol{v}$ remains

$$\hat{\boldsymbol{v}} = R_b \cdot \boldsymbol{z} \quad . \tag{5.144}$$

Indeed the MMSE matrix remains also $R_{\boldsymbol{ee}} = R_b$ as in the canonical backward model. An ML detector acting on $R_b \cdot \boldsymbol{z}$ can obtain optimum performance and highest data rate $\mathcal{I}$, but can also be very complex. The GLE cannot use the GDFE's feedback processing, which simplifies implementation. However the feedfoward matrix W minimizes MSE; further because this matrix has a separate row for each dimension output, it does indeed minimize also each dimensional MSE

$$MMSE_n \triangleq \mathbb{E}\left[|e_n|^2\right] \quad . \tag{5.145}$$

The need for ML decoder complexity arises when R_b is not diagonal. A decoder that ignores the correlation between error values on different dimensions is not optimum, nor is it canonical with non-diagonal R_b. However, it does have an SNR that applies to decoding if the error is (incorrectly) treated as AWGN:

$$SNR_{GLE,n} = \frac{1}{R_{b,n}} \quad , \tag{5.146}$$

where $R_{b,n}$ is the n^{th} diagonal term of R_b, which is only equal to $S_{0,n}$ if R_b is diagonal[27]. It is also biased, so analysis should subtract 1 from each diagonal element. A GLE SNR forms as

$$SNR_{GLE,U} = \left[\prod_{n=1}^{\bar{N}^*} SNR_{GLE,n}\right]^{1/\bar{N}_x} - 1 \quad . \tag{5.147}$$

Clearly $SNR_{GLE} \leq SNR_{GDFE}$ with equality when R_b is diagonal. There is thus a GLE performance loss

$$0 \leq \gamma_{GLE} = \frac{SNR_{GLE} - 1}{SNR_{GDFE} - 1} \leq 1 \tag{5.148}$$

with respect to using the feedback (or precoded) GDFE. The loss magnitude will depend on how close R_b is to diagonal, which in turn will depend on the square-root choice A. Some choices lead to little or no loss, which Section 5.2 develops further.

An example use returns to the Example 5.1.2 and the 3×3 $R_{\boldsymbol{xx}} = I$ input that a designer might try to implement:

```
>> [snrGDFEu, GU, WU, SO, MSWMFU, b, bbar, snrLEU] = computeGDFE(H, sqrt(1.5)*Rxxtt,cb,Lx);
snrGDFEu =  6.8589 dB
snrLEU =    2.0742 dB
```

Thus the nonlinear feedback gains roughly 4.8 dB in this example, even with the input roughly optimized.

5.1.4 The Precoded GDFE

Precoding averts error propagation by essentially moving the feedback section to the transmitter, where previous subsymbols are available without error. Section 3.8 introduced several types of suboptimal simple precoders, such as the Tomlinson and/or Laroia precoders. Similarly Chapter 2's theoretical lossless precoders are the ideal limit of all good precoders. Practical precoder implementation can increase the transmit power slightly as in Section 3.8. The GDFE needs some straightforward modification for precoding.

Transmission design may use precoding to eliminate even error propagation within a GDFE symbol. The feedback section's rows vary with subchannel index n and

$$\boldsymbol{g}_n \triangleq [1 \;\; g_{n,n-1} \;\; g_{n,n-2} \;\; \ldots \;\; g_{n,0}] \;\; \forall \; n = 0, \ldots, \bar{N}^* - 1 \quad . \tag{5.149}$$

[27] Caution - this is not the n^{th} diagonal element of R_b^{-1}.

The precoder again requires an unbiased feedback section:

$$\boldsymbol{g}_{unb,n} = \left[1 \; \frac{\text{SNR}_n + 1}{\text{SNR}_n} \cdot g_{n,n-1} \cdots \frac{\text{SNR}_n + 1}{\text{SNR}_n} \cdot g_{n,0}\right] \quad . \tag{5.150}$$

Each used dimension $n = 0, ..., \overline{N}^* - 1$ has an associated $\bar{b}_n = \frac{1}{2} \cdot \log_2 (1 + \mathcal{E}_n \cdot S_n)$. The lossless precoder then becomes periodic in the symbol period with operation

$$v'_n = \left(v_n - \sum_{i=0}^{n-1} g_{U,n,i} \cdot v'_i \right)_{\mathcal{E}_n} \quad , \tag{5.151}$$

as in Chapter 2's BC. Each of the $\overline{N}^*$ dimensions of the feedforward matrix output should have a modulo operator prior to the decision element, except the first.

Section 3.8's Tomlinson and Laroia precoders also easily follow with the periodic $\boldsymbol{g}_{U,n}$ vectors being used in the usual manner for each corresponding dimension; however, the Laroia precoder requires a cross-dimensional decoder processing (as in Section 3.8) after a first receiver decision element. So it would not be applicable, for instance, to Chapter 2's broadcast channel).

5.1.5 Single-sided Receiver GDFEs

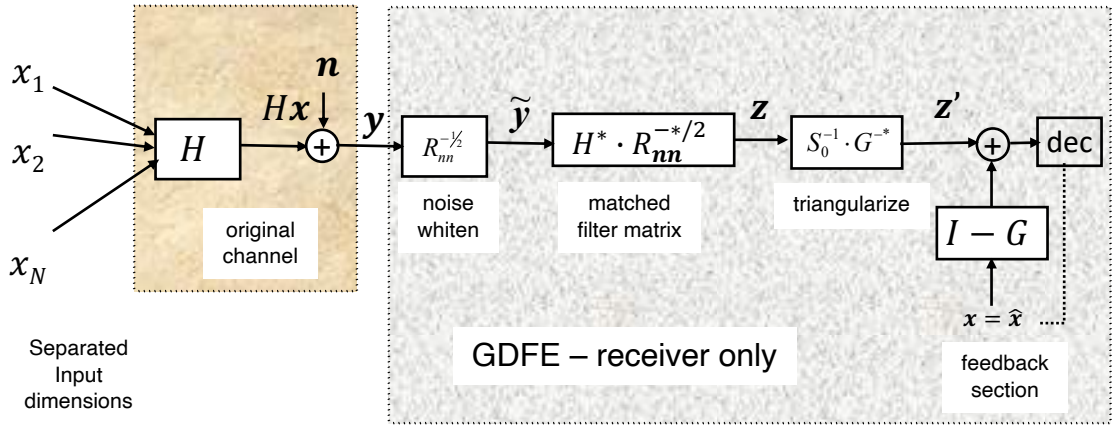


Figure 5.10: The single-sided receiver GDFE.

When the matrix H has rank ϱ_H equal to the number of channel input dimensions, Figure 5.10's special one-sided GDFE can occur. Such a H does not occur in the simple convolution case. However, examples of H that have such rank are:

[**Vector DMT**] The cyclic-block matrix H that implements a common symbol clock $1/T$ and $\bar{b}$ and a common cyclic-prefix ν in several parallel processors, each with their own IDFT discrete modulator with **zero energy on all the other processors' assigned dimensions** - that is Chapter 4's Vector DMT/OFDM with the additional constraint.

[**Multiple-Access Channel**] For instance temporarily viewing $\overline{N} = U$ with $L_{x,u} = 1$, Chapter 2's MAC has a matrix AWGN that corresponds to $\overline{N}$ dimensions independently exciting a common channel (for instance a MIMO transmission system with $\overline{N}$ independent transmit antennas and $\overline{N}$ or more receive antennae). Equivalently $R_{\boldsymbol{xx}}$ is diagonal.

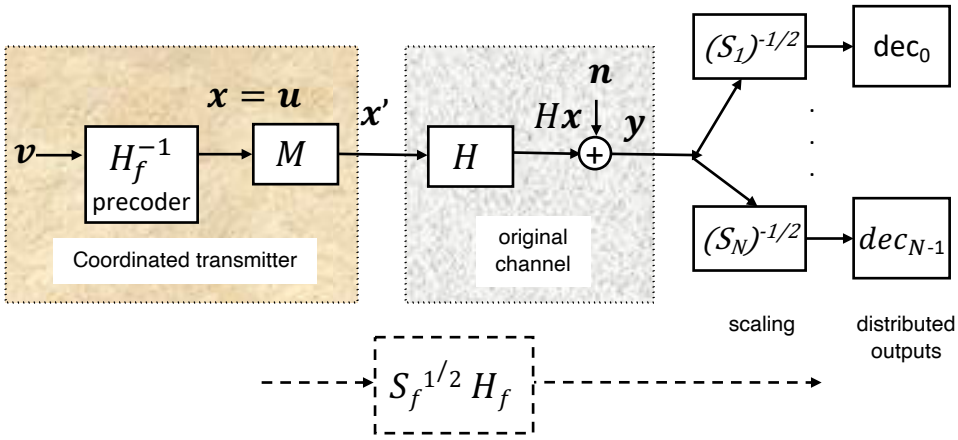


Figure 5.11: The single-sided transmitter GDFE.

The absence of an ability to coordinate the input dimensions with an A or C matrix modulator might correspond to physically distinct locations in which the transmitted signals entering the channel, like Chapter 2's MAC. If the H matrix has rank greater than or equal to the number of inputs, then a diagonal design choice is $A = C = I$, and Figure 5.10's structure remains canonical and one-sided. Sections 5.4 and 5.5 further address the low-rank situation on multi-user channels.

5.1.6 Single-sided Transmitter GDFE

A dual of Subsection 5.1.5's single-sided situation has channel-output dimensions separately implemented, and thus matrix receiver processing is not physically possible, as for instance with Chapter 2's BC. The channel input dimensions share a common processor. If the matrix H is square and non-singular, this corresponds to Chapter 2's BC having all primary-component users.

Figure 5.11 illustrates the single-sided-transmitter GDFE There, the matrix $\widetilde{H}$ has QR-factorization $\widetilde{H} = S_f^{1/2} \cdot G_f \cdot M^*$ where S_f is the diagonal matrix from the ZF-GDFE, G_f is the monic upper triangular matrix, and M is a Hermitian matrix ($MM^* = M^*M = I$). Then the transmit signal is $\boldsymbol{x}' = M \cdot \boldsymbol{v}$ and the receive signal undergoes $\boldsymbol{z}' = S_f^{-1/2} \cdot \boldsymbol{z}$, leaving

$$\boldsymbol{z}' = G_f \cdot \boldsymbol{v} \tag{5.152}$$

A lossless precoder using G_f to create the ZF-GDFE's set of parallel independent subchannels. This choice leads to the overall SNR being

$$\text{SNR}_{zf-gdfe} = \prod_{n=0}^{\bar{N}_x - 1} S_{f,n} \quad . \tag{5.153}$$

When the noise is not white, then there may be additional performance loss with respect to GDFE levels, unless as first presented in Chapter 2, the noise has worst-case mutual-information-minimizing autocorrelation. If the actual noise is worst-case, the single-sided transmitter GDFE achieves canonical performance.

5.2 Special GDFEs: VC, Triangular DFE, and Massive MIMO

Figure 5.9's GDFE for a matrix additive Gaussian-noise (AGN) channel $(H, R_{\boldsymbol{nn}})$ has performance, for Gaussian code with $\Gamma = 0$ dB, that is a function of the input autocorrelation matrix $R_{\boldsymbol{xx}}$ with maximum data rate $\mathcal{I}(\boldsymbol{x};\boldsymbol{y}) = \mathcal{I}(\boldsymbol{v};\boldsymbol{y})$. Best design considers the GDFE input vector to be $\boldsymbol{v}$, with $R_{\boldsymbol{vv}} = I$; the channel matrix $\widetilde{H}$ absorbs any energy scaling through the discrete modulator A so $\widetilde{H} = R_{\boldsymbol{nn}}^{-1/2} \cdot H \cdot A$. The GDFE receiver then directly detects the encoded symbol-vector components v_n, $n = 0, ..., N^* - 1$ in succession to achieve this cannonical performance maximum of $\boldsymbol{SNR} = 2^{2 \cdot \bar{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})}$, for any $R_{\boldsymbol{xx}} \rightarrow R_{\boldsymbol{x'x'}} \rightarrow R_{\boldsymbol{uu}} \rightarrow R_{\boldsymbol{vv}}$ as per Section 5.1. The following lemma essentially summarizes then Section 5.1.

Lemma 5.2.1 *[Finite-Length CDEF Equivalent] Any matrix-AGN has a GDFE system with receiver input $\boldsymbol{y}$ and transmitter input $\boldsymbol{v}$ and input autocorrelation $R_{\boldsymbol{vv}} = I$ such that*

$$SNR_{GDFE,U} = 2^{2 \cdot \bar{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})} - 1 \quad . \tag{5.154}$$

The system can reliably transmit at $\bar{b} \leq \bar{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})$ with any good (scalar) AWGN code with sufficiently small gap Γ on all energized/passed GDFE dimensions/subchannels $\boldsymbol{v}$ corresponding to the given input autocorrelation $R_{\boldsymbol{xx}}$ and retains this same gap to best data rate for that $R_{\boldsymbol{xx}}$ on matrix channel H.

Proof: *the development in Section 5.1.* **QED.**

Sufficiently small gap merits attention: The gap applies to the MMSE on each GDFE (scalar) output dimension, and when $\Gamma > 0$ dB, there is an inaccuracy as per (5.135) and a dependency on GDFE dimensional order. No such inaccuracy nor dependency exists when $\Gamma = 0$ dB. However, the useful result is that with a reasonably good code, this slight deviation is tolerable and the canonical GDFE transmission system is often substantially less complex than an ML detector for this matrix channel. Effectively, the GDFE decouples the demodulation optimization from the good scalar code's decoding. Good AWGN codes also apply equally well when the subsymbol constellation boundary is a Voronoi region like a square. This square limitation (with average energy remaining the same) loses 1.53 dB of shaping gain. Indeed good codes exist that capture the minimum distance properties without regard to the subsymbol constellation boundary. The same codes have Chapter 2's good fundamental-gain γ_f. These good-γ_f codes can reliably achieve $P_e \rightarrow 0$ for any rate less than $\log_2(1 + \frac{SNR}{\gamma_{s,max}})$. The $\gamma_{s,max} = 10^{.153}$ when the constellation boundary is a square. Thus GDFE results also are canonical for such systems, or really any system where the loss is γ_s for the non-hyperspherical boundary. Thus, good code expands to mean any code that would have $\Gamma = 0$ dB with external good shapring code, or more explicitly any code that with probability one satisfies the AEP constraint of only one codeword in each conditional typical set within the given shaprng-constellation boundary.

Imperfect Codes with any boundary: More specifically, the GDFE generates a set of parallel AWGN subchannels that each have mutual information $\mathcal{I}_n$. These subchannels or dimensions have an energy allocation $\mathcal{E}_{v,n}$ that the matrix A absorbs. The dimensional codes use constant or variable constellations with corresponding appropriate decoder, as per Chapter 4's Separation Theorem, with $b_n \approx \mathcal{I}_n$. There are several input energy distributions that correspond to different GDFE dimensional orders, each with different individual-dimensional mutual information, where the differences are among the dimensionally indexed

$$\bar{\mathcal{I}}_n = \frac{1}{2} \cdot \log_2 (S_{0,n}) \quad . \tag{5.155}$$

However, these add to the same total $\bar{\mathcal{I}} = \sum_n \bar{\mathcal{I}}_n$ for all these energy distributions; each such energy distribution corresponds to a different discrete modulator A. With nonzero gap code the loaded data

rate (bits/dimension) is[28]

$$\bar{b}_n = \frac{1}{2}\log_2\left(1 + \frac{S_{0,n} - 1}{\Gamma}\right) \quad . \tag{5.156}$$

Equation (5.156) loses accuracy as Γ increases, similar to Chapter 4's Separation Theorem accuracy loss because the code is not near capacity. Coding and modulation separate best with good codes.

When the input $\boldsymbol{x}$'s dimensions, have a water-filling energy allocation and a good code, they thereby create an $R_{\boldsymbol{xx}}$ that maximizes $I(\boldsymbol{x};\boldsymbol{y})$, and thereby achieve the highest possible data rate, or capacity. So, nothing is lost by reusing a good-scalar-AWGN-designed code on this GDFE-processed matrix-AWGN channel. This section finds ways to construct such a best $R_{\boldsymbol{xx}}$ input and a corresponding GDFE. Similarly, as Γ increases, Chapter 4's gap-dependent water-fill energy (recall $K = \mathcal{E}_n + \Gamma/g_n$) narrows the water-fill band (which can reduce the number of energized dimensions).

GDFE basic types: There exist many $R_{\boldsymbol{vv}} = I$ inputs that achieve the canonical level, corresponding to different feedback sections (and feedforward sections), but the same $\text{SNR}_{\text{GDFE,U}}$. This section investigates two specific forms:

1. **Vector Coding** - which corresponds to the matrix $A = \Phi$ resulting from eigendecomposition of $R_{\boldsymbol{uu}}$ to obtain Φ as the eigenvector matrix that scales by the diagonal square-root-eigenvalue matrix.

2. **Triangular GDFE** - which corresponds to $R_{\boldsymbol{uu}}$'s Cholesky factorization $R_{\boldsymbol{vv}} = \Phi \cdot \Phi^*$ to obtain triangular (causal) Φ that scales by the diagonal square-root matrix of Cholesky factors, and includes the important special case of cyclic GDFE or the CDFE when H (or $\widetilde{H}$) is cyclic, only the latter of which corresponds to Chapter 3's original MMSE-DFE.

As $\overline{N} \to \infty$ (or equivalently as $\overline{N}_x \to \infty$ with $L_x = L_y = 1$ for each spatial transfer) for stationary (Toeplitz) $R_{\boldsymbol{uu}}$, then VC $\to$ MT and the GDFE $\to$ MMSE-DFE set. This holds if the $R_{\boldsymbol{xx}}$ is the same (and each method uses scalar AWGN codes with the same gap on all dimensions), even if that $R_{\boldsymbol{xx}}$ is not a water-fill solution. If the water-fill $R_{\boldsymbol{xx}}$ is used in all, then all attain the highest performance level for the given gap. Section 5.3 considers the construction of the optimum water-filling or any other desired $R_{\boldsymbol{xx}}$ for the CDFE and GDFE to avoid violation of the PWC (which will effectively result in a minimum number of parallel CDFE's that correspond to Chapter 3's minimum-size set of canonical MMSE-DFEs).

5.2.1 Vector-Coding as the GDFE without feedback

Vector coding is a special GDFE case. First, noise-whitening replaces H by $R_{\boldsymbol{nn}}^{-1/2} \cdot H$ and applies SVD

$$R_{\boldsymbol{nn}}^{-1/2} \cdot H = F \cdot \Lambda \cdot M^* \quad . \tag{5.157}$$

Vector coding's input vectors are this channel's right-singular-vectors, M, with M' as those singular vectors that associate with nonzero singular values,

$$\boldsymbol{x} = \sum_{n=1}^{\overline{N}_x} X_n \cdot \boldsymbol{m}_n = M' \cdot \boldsymbol{X}' \quad . \tag{5.158}$$

The choice $R_{\boldsymbol{uu}} = M \cdot R_{\boldsymbol{XX}} \cdot M^*$ thus characterizes vector-coding – any other $R_{\boldsymbol{uu}}$ is not vector coding, but could correspond to other GDFEs. Channel-pass-space reduction removes the column vectors $\boldsymbol{m}_n$, $n > \varrho_{\widetilde{H}}$ from the M (the singular values are ordered from largest to smallest) so to obtain $\widetilde{M}$

$$\boldsymbol{x}' = \sum_{n=1}^{\varrho_{\widetilde{H}}} X_n \cdot \boldsymbol{m}_n = \widetilde{M} \cdot \tilde{\boldsymbol{X}} \quad . \tag{5.159}$$

[28] recall the unbiased SNR is $SNR = SNR_{bias} - 1$ for any gap

Vector coding with water-filling may also zero some pass-space dimensions, which then creates a singular $R_{\boldsymbol{xx}}$. The corresponding input autocorrellation matrix, $R_{\boldsymbol{x}'\boldsymbol{x}'}$, is then (with M'' as the first ϱ_x columns of M')

$$\boldsymbol{x}'' = \sum_{n=1}^{N^*=\varrho_x} X_n \cdot \boldsymbol{m}_n = M'' \cdot \boldsymbol{X}'' \quad . \tag{5.160}$$

and finally then

$$v_n = \frac{X_n''}{\underbrace{\sqrt{\mathbb{E}\left[X_n'' \cdot X_n''^*\right]}}_{\sqrt{\mathcal{E}_n}}} \quad n = 1, ..., \varrho_x \quad , \tag{5.161}$$

with then $A = M'' \cdot \text{Diag}\{\sqrt{\mathcal{E}_n}\}_{n=1,...,\varrho_x}$. The forward canonical matrix R_f is then diagonal

$$R_f = \tilde{H}^* \cdot \tilde{H} = \text{diag}\left\{\mathcal{E}_{\varrho_x} \cdot \lambda_{\varrho_x}^2 \quad , \quad \mathcal{E}_{\varrho_x - 1} \cdot \lambda_{\varrho_x - 1}^2 \quad , ..., \quad \mathcal{E}_1 \cdot \lambda_1^2\right\} \quad , \tag{5.162}$$

and thus $R_b^{-1} = R_f + I$ is also diagonal. Consequently, $G = I$. The feedback section thus simplifies to no feedback at all (which avoids the error-propagation concern with suboptimal dimension-by-dimension detection), and the feedforward section simply becomes $S_0^{-1} = R_b$, an independent (diagonal) scaling of each received matched-filter-output dimension.

Lemma 5.2.2 [Vector Coding as the Optimum and Canonical Transmission System] *Vector coding is both canonical and optimum as a transmission method for all* $\Gamma \geq 0$ *dB.*

Proof: Since the GDFE provides $\text{SNR}_{dgfe} = 2^{2 \cdot \bar{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})}$, VC is canonical. Since there is zero GDFE feedback section, $G = I$, there is no concern for suboptimal detection, and an independent ML decoder on each subchannel (which is a simple slicer for uncoded transmission, but would be the ML decoder for the code applied to that dimension more generally) is optimum. **QED.**

The VC-based GDFE's joint optimality and canonical nature is a special discrete modulator with modulating vectors equal to the channel's right singular vectors. With bias removed, VC further has a MMSE equal to the (scaled) noise, so the gap only multiplies such noise. While the effect in (5.135) remains, the amplification of other dimensions' signal components is also absent, so any gap-approximation inaccuracies are less than with a nonzero-feedback GDFE. The detector in this one VC-GDFE case is indeed ML. No other[29] structure produces a diagonal channel with the consequence of $G = I$. Any other $G \neq I$ may have canonical SNR ($2^{2 \cdot \bar{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})}$) but assumes correct decisions and thus is not ML nor optimum as a detector. Thus, VC is truly the best performing GDFE structure for transmission on the channel $(H, R_{\boldsymbol{nn}})$. A receiver could implement an ML detector for $G \neq I$, but such a full ML detector with $G \neq I$ is likely much more complex than VC and would degrade more with $\Gamma > 0$ dB.

5.2.2 The triangular GDFE

The GDFE can also use Cholesky factorization to relate a nonsingular input autocorrelation matrix $R_{\boldsymbol{uu}}$ to a diagonal input autocorrelation matrix $R_{\boldsymbol{vv}}$:

$$R_{\boldsymbol{uu}} = \Phi \cdot \Phi^* = G_\phi \cdot S_u \cdot G_\phi^* \quad , \tag{5.163}$$

[29] A part from trivial phase differences; a factorization that allows complex entries for Λ is essentially the same as one with a positive real Λ.

where $\Phi = G_\phi \cdot S_u^{1/2}$ is upper-triangular.[30]

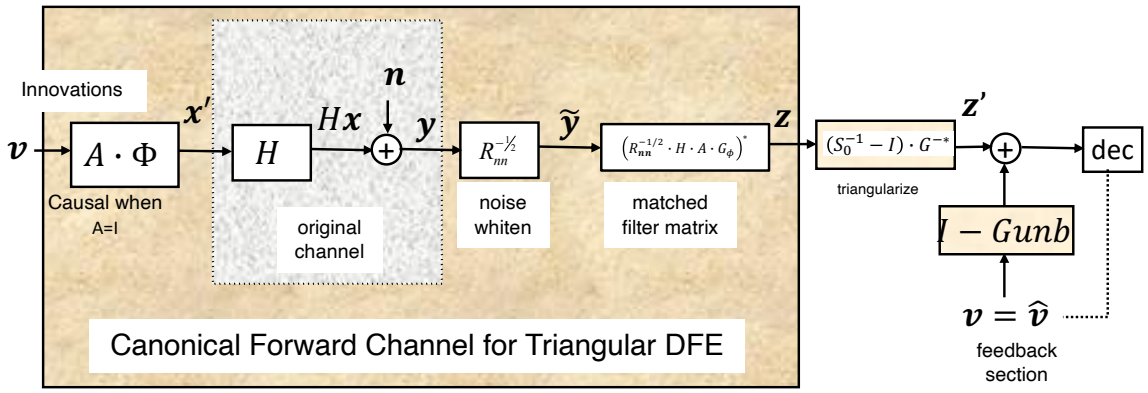


Figure 5.12: Triangular DFE with white input $\boldsymbol{v}$.

Furthermore,

$$\boldsymbol{u} = \Phi \cdot \boldsymbol{v} \quad . \tag{5.164}$$

The data-message's subsymbols are the $\overline{N}^*$ components of $\boldsymbol{v}$, and

$$\boldsymbol{x}' = \tilde{A} \cdot \boldsymbol{u} = \underbrace{(\tilde{A} \cdot \Phi)}_{A} \cdot \boldsymbol{v} \quad . \tag{5.165}$$

The corresponding GDFE has design for

$$\widetilde{H} = R_{\boldsymbol{nn}}^{-1/2} \cdot H \cdot \tilde{A} \cdot \Phi = R_{\boldsymbol{nn}}^{-1/2} \cdot H \cdot A \tag{5.166}$$

and otherwise follows exactly as in Section 5.1 with input v_n, $n = 0, ..., \overline{N}^* - 1$ as input data and also as the GDFE's feedback-section input. This structure appears in Figure 5.12.

The GDFE's feedforward matrix and feedback matrices remain nonzero triangular (and not necessarily diagonal), leading to a higher implementation complexity and suboptimal detection; however, because the input has diagonal $R_{\boldsymbol{vv}} = I$ by construction, this GDFE is still canonical. The SNR_{gdfe} performance still approaches capacity just like the VC system. This GDFE's most interesting temporal-channel quality may be the triangular transmission filter, which suggests a causally implemented filter when the input $R_{\boldsymbol{xx}}$ can be made nonsingular as $\overline{N} \to \infty$. Causal filters satisfy the PWC. Upper-triangular filters can be implemented essentially without delay in this special case. In general, however, the discrete modulator $\tilde{A}$ can cause $A = \tilde{A} \cdot \Phi$ to be non triangular and there is then no causal-like implementation.

Even though the GDFE input $\boldsymbol{v}$ is not the same as VC's modal input $\boldsymbol{v}$ and the decision-based detector is not optimum, the system-input vector $\boldsymbol{v}$ carries the same information that leads to the same GDFE SNR in the absence of any previous-decision errors. The parallel-subchannel set is also different, even though the overall equivalent AWGN has the same SNR. Both can operate reliably at capacity. Figure 5.13 illustrates the linear prediction relationships for the triangular GDFE transmitter. With respect to Chapter 3's infinite-length MMSE-DFE, Φ replaces $G_x(D)$ in the finite-length symbol case.

[30]The upper triangular matrix correctly appears first here, unlike the factorization of R_b^{-1} where the upper triangular matrix appeared last. The upper-lower matlab Cholesky should use the chol($JR_{\boldsymbol{uu}}J$) command where J has all ones down the anti-diagonal and zeros elsewhere, and then **pre- and post-**multiply the resultant matrix by J, and then take congugate transpose to obtain Gxbar, or in matlab Gxbar=(J*chol(J*rxx*J)*J)', and then finally remove diagonal terms by post multiplying as Gx=Gxbar*inv(diag(diag(Gxbar))), to obtain G_x. $R_{\boldsymbol{vv}}$ is diag(diag(Gxbar))*diag(diag(Gxbar). For those with access to Appendix E's programs, the matlab program lohc.m directly computes this same Gx=lohc(rxx).

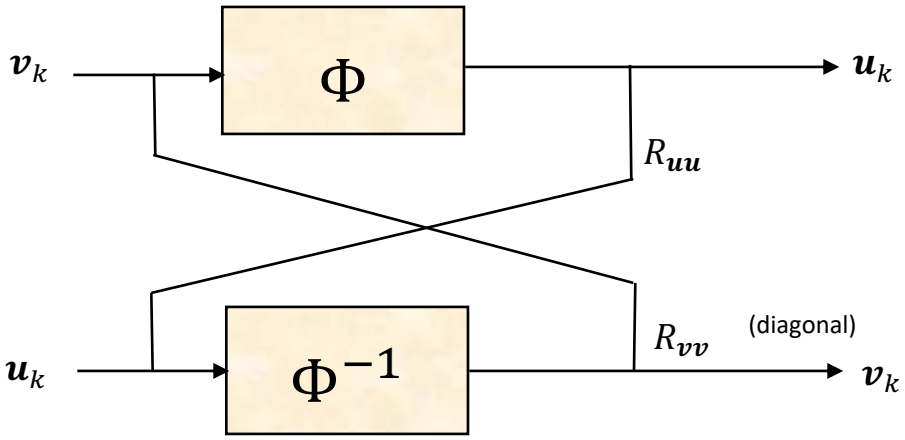


Figure 5.13: Innovations and input.

EXAMPLE 5.2.1 *[1+.9·D^{-1} Triangular GDFE]* This example revisits Example 5.1.2, but this time uses the vector-coded modulation vectors directly. The following matlab commands should be self explanatory and work the complete process from channel to final design for the triangular GDFE on this channel with $\overline{N} = 2$ and $\nu = 1$ ($\overline{N}_x = 3$).

```
Recalling
H =
    2.1155    2.3505         0
         0    2.1155    2.3505
>> [Ct, Ot, Ruutt] = fixmod(H, eye(3), eye(3));
>> [A, OA, Ruupp] = fixin(Ruutt, Ct);
% previous square-root forms as
>> Gxbar=lohc(Ruupp);
>> Gx=Gxbar*inv(diag(diag(Gxbar))) ;
>> Xmit=A*Gxbar;
cb=2;
Lx=3;
```

Continuing now with the square-root matrix Xmit that includes both A and the Cholesky square-root of $R_{\boldsymbol{u}'\boldsymbol{u}'}$

```
[snrGDFEu, GU, WU, S0, MSWMFU, b, bbar,snrLEu] = computeGDFE(H, Xmit,cb,Lx);
snrGDFEu =    5.5427 dB
>> GU =
    1.0000    0.5731
         0    1.0000
>> WU =
    0.1328         0
   -0.0492    0.0972
>> S0 =
    8.5275         0
         0   11.2899
>> MSWMFU =
    0.3645    0.0000
    0.0179    0.3073
>> b' =    1.5461    1.7485
>> bbar =    1.0982
snrLEu =    6.1761 dB
```

The receiver design needs the above G_U and also the forward combined filter MSWMFU. The transmitter implements Xmit. Further there are two used (real) dimensions with the triangular GDFE's bit distributions shown on each.

5.2.3 The Circulant DFE (CDFE)

The $\overline{N} \times (\overline{N} + \nu)$, or $\overline{N}_y \times \overline{N}_x$ with $L_x = L_y = 1$, temporal convolution matrix H can never have rank greater than $\overline{N}$. Also, the square $\overline{N} \times \overline{N}$ matrix H that arises from cyclic-prefix use (the same cyclic prefix used for DMT) always has rank $\varrho_H \leq \overline{N}$. The case $\varrho_x = \overline{N}^* = \varrho_H = \overline{N}$ is occurs when the input autocorrelation matrix $R_{\boldsymbol{xx}}$ corresponds to a sampling rate where loading energizes all input dimensions. One situation where $\varrho_H < \overline{N}$ occurs when the channel has notches (zero gain) at the exact frequencies that would be considered "DMT subcarrier" frequencies[31]. Best input design always zeroes energy on any such notch frequencies, which leads to multiple discontiguous bands as Section 5.3 addresses. When only a single band (subset) of carriers have nonzero energy, the CDFE sometimes has the name "Single-Carrier OFDM."

In the simpler nonsingular-input case,

$$R_{\boldsymbol{uu}} = R_{\boldsymbol{xx}} \quad , \tag{5.167}$$

the input factorization trivially becomes Cholesky factorization, and

$$\boldsymbol{x} = \Phi \cdot \boldsymbol{v} \tag{5.168}$$

with Φ a upper-triangular $\overline{N} \times \overline{N}$ matrix. This situation can correspond to the **circulant DFE** or **CDFE**. The CDFE case (with cyclic prefix) has a stationary channel with a circulant Toeplitz H and finite fixed ν. As $\overline{N} \to \infty$, this CDFE converges to Chapter 3's MMSE-DFE as in Section 5.3. With input singularity corresponding to input design zeroing certain non-zero-measure frequency bands, or equivalently failing to satisfy the PWC, then the situation has instead multiple CDFEs each converging to its own corresponding MMSE-DFE in its own PWC-satisfying band (see Section 3.11).

Convergence: For finite large $\overline{N}$, the transmit filter corresponds to any of G_ϕ's middle rows, while the MMSE-DFE's feedback section corresponds to the middle rows of the CDFE's G.[32] The CDFE's matched-filter matrix converges to a matched filter, and any feedforward matrix middle row converges to the MMSE-DFE feedforward filter. The CDFE exists only when $R_{\boldsymbol{xx}}$ is nonsingular, which may not occur, especially with water-filling's typical dimensional zeroing. Section 5.3.4 provides an adjustment for multiple disjoint bands, which is called "generalized Cholesky factorization." Under the cyclic-channel-matrix restriction, DMT is clearly an optimum transmission method with best SNR and ML detection achieved because $G = I$, and there is no error propagation. The CDFE when its input exists and matches DMT, canonically obtains the same SNR and reliable data rate, but uses a suboptimal detector. An ML detector for this CDFE can be very complex, but would match canonically the simpler DMT performance. Chapter 4's Separation Theorem now takes additional support in that basically the C-OFDM system and CDFE have the same performance if the simple AWGN code has $\Gamma = 0$ dB. So C-OFDM is also canonical if it uses the same $R_{\boldsymbol{xx}}$ and a very low Γ code. Chapter 4's Separation-Theorem discussion notes that use of large-gap codes can erode the C-OFDM/DMT equivalence rapidly. Again, the good code that ignores only shaping gain exhibits all the same canonical performance, for an SNR reduced by the shaping loss.

EXAMPLE 5.2.2 *[$1 + .9 \cdot D^{-1}$ CDFE with PAM input]* Returning to this text's familiar $1 + .9 \cdot D^{-1}$ example, the AWGN noise variance is $\frac{\mathcal{N}_0}{2} = .181$ and $\bar{\mathcal{E}}_{\boldsymbol{x}} = 1$. This cyclic prefix is length $\nu = 1$ on a block size of $\overline{N} = N = 8$ ($N_x = 9$). Thus, $R_{\boldsymbol{vv}} = R_{\boldsymbol{uu}} = I$. For this case, $A = I$ also because $R_{\boldsymbol{vv}}$ is diagonal. Rather than repeat the analysis of previous developments mathematically here, this example simply inserts some descriptive matlab text with results:

```
>> H=toeplitz([.9 zeros(1,7)]',[.9 1 zeros(1,6)]);
>> H(8,1)=1
H = 0.9000    1.0000         0         0         0         0         0         0
         0    0.9000    1.0000         0         0         0         0         0
```

[31] Equivalently $H(D) = \prod_{n=1}^{\overline{N}} (1 + aD)$ has at least one n for which $|a_n| = 1$.

[32] All these middle rows will have the nonzero coefficient values tending to the same fixed filter when $\overline{N}$ is long enough.

```
         0         0    0.9000    1.0000         0         0         0         0
         0         0         0    0.9000    1.0000         0         0         0
         0         0         0         0    0.9000    1.0000         0         0
         0         0         0         0         0    0.9000    1.0000         0
         0         0         0         0         0         0    0.9000    1.0000
    1.0000         0         0         0         0         0         0    0.9000
>> [snrGDFEu, GU, WU, SO, MSWMFU, b, bbar,snrLEu] = computeGDFE(1/sqrt(.181)*H, eye(8), cb, 9)
>> snrGDFEu =    7.1666 dB
>> GU =
    1.0000    0.4972         0         0         0         0         0    0.4972
         0    1.0000    0.6414         0         0         0         0   -0.2899
         0         0    1.0000    0.6930         0         0         0    0.1780
         0         0         0    1.0000    0.7128         0         0   -0.1113
         0         0         0         0    1.0000    0.7206         0    0.0702
         0         0         0         0         0    1.0000    0.7237   -0.0444
         0         0         0         0         0         0    1.0000    0.7531
         0         0         0         0         0         0         0    1.0000
>> MSWMFU =
    0.2115         0         0         0         0         0         0    0.2351
    0.1798    0.2729         0         0         0         0         0   -0.1371
   -0.1104    0.1601    0.2948         0         0         0         0    0.0841
    0.0691   -0.1002    0.1525    0.3033         0         0         0   -0.0526
   -0.0435    0.0631   -0.0961    0.1495    0.3066         0         0    0.0332
    0.0275   -0.0399    0.0608   -0.0945    0.1483    0.3079         0   -0.0210
   -0.0174    0.0253   -0.0385    0.0598   -0.0939    0.1478    0.3084    0.0133
   -0.0933    0.0418    0.0010   -0.0439    0.0961   -0.1687    0.2772    0.1647
>> b' =
    1.7297    1.5648    1.5156    1.4978    1.4909    1.4882    1.4871    1.0792
>> bbar =    1.3170
>> snrLEu =    1.5191
```

The important G matrix clearly converges to what is known (from Chapter 3's instance of this same example in multiple places) to be the (unbaised) MMSE-DFE feedback section of $1 + .75 \cdot D$ in the middle rows of G_u. This behavior would happen for any sufficiently large $\overline{N}$ on this example and illustrates that the MMSE-DFE is simply the limit of the CDFE as $\overline{N}$ becomes large. That is, no symbol-blocking is necessary if the transmitted symbol sequence is just long enough. However, at the beginning of transmission, especially on this maximum phase channel, the filter is different during a transient. An MMSE-DFE designed according to Chapter 3 for finite or infinite-length filters would actually only achieve its 8.4 dB SNR asymptotically, because the assumptions made in that derivation were stationarity for all time, when in fact this example's more realistic situation illustrates that transmission begins at one point in time and continues. Thus, the CDFE then processes the initial dimensions/samples better than any MMSE-DFE in Chapter 3. Indeed the 7.16 dB SNR here increases to 7.75 dB at $N = 16$ (real baseband $N_x = N + 1$), 8.26 dB at $N = 100$ and 8.3 dB at $N = 200$, which all illustrate the fastest or best convergence to the eventual 8.4 dB. Thus, use of a stationary-designed MMSE-DFE over finite block lengths does not attain the asymptotic performance. Best performance instead would use the finite-length CDFE with N equal to the packet size.

This example's feedforward filter also converges, but clearly has significant terms for all dimensions, at least for $N = 8$. The convergence is evident in the middle rows to [0.22 0.36 0.58 −.92 0.145]. The non-zero "end" dimension here is related to $\nu = 1$ and eventually just implies an implementation with Chapter 3's finite-equalizer-causality delay parameter $\Delta > 0$ to handle the maximum-phase channel.

It is interesting to compare the CDFE with a conventional GDFE as in the design of Example 5.1.2 except that N is increased from 2 there to 8 here to produce the results with now energy 9/8 fed into each of the dimensions avoiding the channel singularity on one dimension and transporting its energy instead on the other dimensions.

```
>> H=(1/sqrt(.181))*toeplitz([.9 zeros(1,7)]',[.9 1 zeros(1,7)]);
H=(1/sqrt(.181))*toeplitz([.9 zeros(1,7)]',[.9 1 zeros(1,7)]);
>> [Ct, Ot, Ruutt] = fixmod(H, eye(9), eye(9));
>> Ct
    0.7764    0.2012   -0.1811    0.1630   -0.1467    0.1320   -0.1188    0.1069
    0.2012    0.8189    0.1630   -0.1467    0.1320   -0.1188    0.1069   -0.0962
   -0.1811    0.1630    0.8533    0.1320   -0.1188    0.1069   -0.0962    0.0866
    0.1630   -0.1467    0.1320    0.8812    0.1069   -0.0962    0.0866   -0.0779
```

```
   -0.1467    0.1320   -0.1188    0.1069    0.9038    0.0866   -0.0779    0.0702
    0.1320   -0.1188    0.1069   -0.0962    0.0866    0.9221    0.0702   -0.0631
   -0.1188    0.1069   -0.0962    0.0866   -0.0779    0.0702    0.9369    0.0568
    0.1069   -0.0962    0.0866   -0.0779    0.0702   -0.0631    0.0568    0.9489
   -0.0962    0.0866   -0.0779    0.0702   -0.0631    0.0568   -0.0511    0.0460
   >> [A, OA, Ruupp] = fixin(Ruutt, Ct)
>> A
    0.7764    0.2012   -0.1811    0.1630   -0.1467    0.1320   -0.1188    0.1069
    0.2012    0.8189    0.1630   -0.1467    0.1320   -0.1188    0.1069   -0.0962
   -0.1811    0.1630    0.8533    0.1320   -0.1188    0.1069   -0.0962    0.0866
    0.1630   -0.1467    0.1320    0.8812    0.1069   -0.0962    0.0866   -0.0779
   -0.1467    0.1320   -0.1188    0.1069    0.9038    0.0866   -0.0779    0.0702
    0.1320   -0.1188    0.1069   -0.0962    0.0866    0.9221    0.0702   -0.0631
   -0.1188    0.1069   -0.0962    0.0866   -0.0779    0.0702    0.9369    0.0568
    0.1069   -0.0962    0.0866   -0.0779    0.0702   -0.0631    0.0568    0.9489
   -0.0962    0.0866   -0.0779    0.0702   -0.0631    0.0568   -0.0511    0.0460
>> Ruupp
    6.3966   -4.8569    4.3712   -3.9341    3.5407   -3.1866    2.8680   -2.5812
   -4.8569    5.3712   -3.9341    3.5407   -3.1866    2.8680   -2.5812    2.3231
    4.3712   -3.9341    4.5407   -3.1866    2.8680   -2.5812    2.3231   -2.0908
   -3.9341    3.5407   -3.1866    3.8680   -2.5812    2.3231   -2.0908    1.8817
    3.5407   -3.1866    2.8680   -2.5812    3.3231   -2.0908    1.8817   -1.6935
   -3.1866    2.8680   -2.5812    2.3231   -2.0908    2.8817   -1.6935    1.5242
    2.8680   -2.5812    2.3231   -2.0908    1.8817   -1.6935    2.5242   -1.3717
   -2.5812    2.3231   -2.0908    1.8817   -1.6935    1.5242   -1.3717    2.2346
>> Gxbar=lohc(Ruupp);
Gx=Gxbar*inv(diag(diag(Gxbar))) ;
Xmit=A*Gxbar
>> Xmit =
    0.8812   -0.0000    0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000
    0.2283    0.8757    0.0000    0.0000   -0.0000   -0.0000   -0.0000    0.0000
   -0.2055    0.2397    0.8681   -0.0000    0.0000    0.0000    0.0000    0.0000
    0.1850   -0.2157    0.2554    0.8574   -0.0000    0.0000    0.0000    0.0000
   -0.1665    0.1942   -0.2299    0.2779    0.8416    0.0000    0.0000   -0.0000
    0.1498   -0.1747    0.2069   -0.2501    0.3120    0.8163    0.0000    0.0000
   -0.1348    0.1573   -0.1862    0.2251   -0.2808    0.3678    0.7710   -0.0000
    0.1213   -0.1415    0.1676   -0.2026    0.2527   -0.3310    0.4733    0.6690
   -0.1092    0.1274   -0.1508    0.1823   -0.2274    0.2979   -0.4260    0.7433
```

Continuing with this non-cyclic alternative transmit-filter matrix

```
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar] = computeGDFE(H, Xmit,2,9)
snrGDFEu = 7.4896 dB
GU =
    1.0000    0.8573    0.0000    0.0000   -0.0000   -0.0000   -0.0000   -0.0000
         0    1.0000    0.7628    0.0000   -0.0000    0.0000    0.0000    0.0000
         0         0    1.0000    0.7174   -0.0000    0.0000    0.0000    0.0000
         0         0         0    1.0000    0.6899    0.0000    0.0000   -0.0000
         0         0         0         0    1.0000    0.6624    0.0000    0.0000
         0         0         0         0         0    1.0000    0.6189    0.0000
         0         0         0         0         0         0    1.0000    0.5233
         0         0         0         0         0         0         0    1.0000
MSWMFU =
    0.4165    0.0000    0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000
    0.0471    0.3738    0.0000   -0.0000    0.0000    0.0000    0.0000    0.0000
   -0.0294    0.0650    0.3560   -0.0000   -0.0000   -0.0000    0.0000    0.0000
    0.0178   -0.0394    0.0693    0.3487    0.0000    0.0000   -0.0000   -0.0000
   -0.0104    0.0231   -0.0407    0.0669    0.3452   -0.0000    0.0000    0.0000
    0.0058   -0.0129    0.0227   -0.0374    0.0599    0.3415    0.0000    0.0000
   -0.0029    0.0065   -0.0114    0.0188   -0.0302    0.0479    0.3328    0.0000
    0.0011   -0.0024    0.0042   -0.0069    0.0111   -0.0176    0.0279    0.3023
b' =
    1.3789    1.4498    1.4859    1.5067    1.5249    1.5512    1.6042    1.7592
bbar =    1.3623
```

Because there is no cyclic-prefix energy loss, this solution is slight better, however the filters are less stationary in this case and may not converge as quickly[33], but the performance is better (because no energy is wasted in a cyclic prefix or in the channel's null space), and the SNR is 7.5 dB. Vector Coding with equal-energy transmission on the 8 best singular-value dimensions would achieve this same performance level. This nonstationary triangular GDFE more rapidly converges in SNR with increasing N to a point intermediate to the 8.4 dB of the MMSE-DFE (which effectively is the CDFE and inserts energy into a useless dimension) and

[33] Any infinite-length extension will deviate by a vanishingly small amount from cyclic as $N \to \infty$.

less than the 8.8 dB maximum that only occurs with water-filling. ($N_y = N$, $N_x = N + \nu$)

$$8.4 \text{ dB} < \lim_{N \to \infty} \text{SNR}_{N_x/N_y, GDFE, u} < 8.8 \text{ dB} \quad . \tag{5.169}$$

5.2.4 Diagonal Dominance and Massive MIMO Linear GDFE Simplification

Chapter 2 defines **massive MIMO** as $L_y >> U$ for the multiuser MAC and $L_x >> U$ for the multiuser BC, where U is the number of users. In a more general context, $U << L$ simply means that H matrix is very "tall" or very "fat," respectively; this leads to the term "massive." GDFEs have a useful situation for a wide array of commonly encountered tall/fat H channel matrices that have low correlation between the linearly independent U columns or rows respectively; essentially the GDFE simplifies to a linear, but still canonical, structure where $G \to I$, i.e,. no feedback nor precoder is necessary.

MU-MIMO, Massive-MIMO, Multiple Access Channel: Typically convolution matrices do not have this structure, and usually are not tall/fat for reasonable frequency-time decompositions (closer to square if $\nu << \overline{N}$, or more generally $\overline{N}_x \approx \overline{N}$). With MAC channel elements $\widetilde{H} = [h_{i,k}]_{i=0,\ldots,L_y;k=0,\ldots,U}$, the correlation requirement uses (for each tone in vector DMT)

$$\langle \tilde{h}_{i,\ell} \cdot \tilde{h}^*_{k,\ell} \rangle = \frac{1}{L_y} \cdot \tilde{\boldsymbol{h}}^*_i \cdot \tilde{\boldsymbol{h}}_k = \frac{1}{L_y} \cdot \sum_{\ell=0}^{L_y - 1} \tilde{h}^*_{i,\ell} \cdot \tilde{h}_{k,\ell} \approx |\tilde{h}|^2 \cdot \delta_{i,k} \; . \tag{5.170}$$

This means the massive MIMO $\widetilde{H}^* \cdot \widetilde{H}$ tends toward diagonal, or is **diagonally dominant**. When the system is massive MIMO (usually wireless), this means the spatial correlation between the different users' signals arriving at the common receiver is low, and indeed the random-vector columns corresponding to the different users received signals are independent. A convolution matrix rarely has this structure. Crosstalking cables of wires also have this structure also for some frequency bands. This sometimes also has the name "rich scattering" in that the different users' MIMO spatial channels robustly differ from one another.

The GDFE has a crucial step of factoring the $U \times U$ (really $U^o \times U^o$ but all users are primary in the massive MIMO case)

$$R_b^{-1} = R_f + I = G \cdot S_0 \cdot G^* \quad . \tag{5.171}$$

The MAC spatial inputs will have a diagonal $R_{\boldsymbol{xx}}$ so a diagonal R_b^{-1}, equivalently diagonal R_f, which implies that $G = I$ and no feedback is necessary. This system retains canonical GDFE performance. The MAC also has $A = I$, so then

$$R_f = \widetilde{H}^* \cdot \widetilde{H} \quad . \tag{5.172}$$

While R_b and R_f have the lower dimensionality U, the Massive MIMO matrix $\widetilde{H}$ has tall columns, so that (5.170) will be true, particularly in the Law of Large Numbers (LLN, see Chapter 2) sense. Equivalently R_b/L_y or R_f/L_y will have very small or negligible off-diagonal terms, particularly when $L_y >> U$. Indeed in many wireless applications just $L_y \geq U$ is sufficient, so often linear feedforward-only receiver matrix filtering (basically the GDFE's $W_{unb} \cdot \widetilde{H}^*$) is sufficient for canonical performance (and indeed no error propagation either). Thus, much literature actually only teaches the linear matrix filter for the MAC as sufficient. This is not true in general, even with arbitrary tall matrices $\widetilde{H}$, but is true when the low spatial correlation in (5.170) holds. When this situation does hold, the MAC (or channel more generally) is diagonally dominant, and linear processing is nearly canonical.

MU-MIMO, Massive-MIMO, Broadcast Channel: Again for each $\widetilde{H}_n$, the BC is slightly more complex to analyze, but exhibits the same effect when $L_x >> U$. In this case, with any energy scaling from $R_{\boldsymbol{vv}}$'s normalization absorbed into the special square-root transmitter matrix $A = R_{\boldsymbol{xx}}^{1/2}$

$$R_b^{-1} = A^* \cdot H^* \cdot R_{wcn}^{-1} \cdot H \cdot A + I \tag{5.173}$$

where R_{wcn} is the $U \times U$ worst-case noise autocorrelation, whose presumed existence limits the BC achievable data rates. In this case the fat matrix is H (equivalently also $\widetilde{H} = R_{wcn}^{-1/2} \cdot H$ is fat) and A is tall. If the BC's rows - instead of columns – (including worst-case noise adjustment) satisfy the equivalent of (5.170), then any A matrix that will attempt to match-filter to the BC to increase energy transfer would have column $\boldsymbol{a}_u$ such that

$$\langle \tilde{h}_{\ell,i} \cdot a^*_{\ell,k} \rangle = \frac{1}{L_x} \cdot \sum_{\ell=0}^{L_x-1} \tilde{h}^*_{\ell,i} \cdot a_{\ell,k} \approx \delta_{i,k} \; . \tag{5.174}$$

Again by the LLN, this same effect of diagonal dominance causes $G = I$ when $L_x >> U$. Not all BC's satisfy (5.174), even with large L_x. However in practice, $L_x \geq U$ is often sufficient for R_b^{-1} to be diagonally dominant and the feedback (lossless precoder for the BC) becomes unnecessary to approach the GDFE's canonical performance level.

5.2.4.1 Generalized Cholesky Algorithm

Fig 5.14's Generalized Cholesky Algorithm follows and operates on the reordered input that occurs during the creation of a permutation matrix J_A, Traditional Cholesky[34] acts upon the nonsingular reordered sub-autocorrelation matrix (see Subsection 5.3.3.1)

$$R_{\boldsymbol{x}\boldsymbol{x}}(\overline{N}^* - 1) = \Phi \cdot S_x \cdot \Phi^* \quad . \tag{5.175}$$

Generalized Cholesky exploits this interpretation. Cholesky factorization produces the MMSE linear-prediction estimate of x_n from $\{x_{N-1} ... x_0\}$ through Φ^{-1} through the successively computed errors v_n (which are temporary to this section and not elements of a normalized $\boldsymbol{v}$), or equivalently an MMSE estimate direction from the values of v_n with Φ:

$$x_n = v_n + \underbrace{g_{g,n-1} \cdot v_{n-1} + ... + g_{n,0} \cdot v_0}_{\hat{x}_n} \quad . \tag{5.176}$$

Continuing with the MMSE interpretation, the rightmost $\overline{N}^*$ positions in $R_{\boldsymbol{x}\boldsymbol{x}}$'s upper $\overline{N}_x - \overline{N}^*$ rows are $R_{\boldsymbol{x}_{\bar{\nu}}\boldsymbol{x}_\nu}$. These rightmost positions essentially define a $\boldsymbol{x}_{\bar{\nu}}$ that depends on $\boldsymbol{x}_\nu$ through a linear combination that Generalized Cholesky finds. This positions constitute the $(\overline{N}_x - \overline{N}^*) \times \overline{N}^*$ cross-correlation matrix between $\boldsymbol{x}_{\bar{\nu}}$ and $\boldsymbol{x}_\nu$. This cross-correlation matrix helps construct MMSE estimates, namely $\hat{\boldsymbol{x}}_{\bar{\nu}} = R_{[\boldsymbol{x}_{\bar{\nu}}\boldsymbol{\nu}]} \cdot S_x^{-1} \cdot \boldsymbol{\nu}$. Then, the MMSE estimator (and discrete modulator) for the nonsingular case is

$$A = \begin{bmatrix} R_{\boldsymbol{x}_{\bar{\nu}}\boldsymbol{x}_\nu} \cdot \Phi^{-*} \cdot S_x^{-1} \\ \Phi \end{bmatrix} = \begin{bmatrix} \tilde{\Phi}_x \\ G_x \end{bmatrix} \quad , \tag{5.177}$$

which is generalized triangular, so $\boldsymbol{x} = A \cdot \boldsymbol{\nu}$ and

$$R_{\boldsymbol{x}\boldsymbol{x}} = A \cdot S_x \cdot A^* \quad . \tag{5.178}$$

The discrete modulator can generate $\boldsymbol{x}$ "causally" with the last $\overline{N}_x - \overline{N}^*$ dimensions depending exclusively only on previous inputs $\boldsymbol{\nu}$ as in Figure 5.14. The diagonal factor $S_x^{1/2}$ can subsequently be absorbed into $\widetilde{H}$ upon return to GDFE design.

[34] The correct upper triangular matrix can be computed using matla's lichols command.

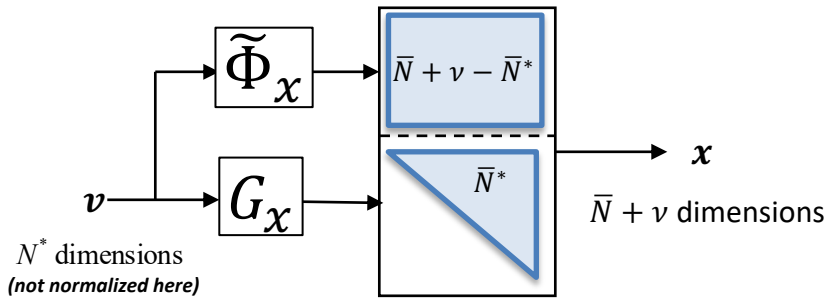


Figure 5.14: Generalized Cholesky's interpolation to full set of dimensions.

Figure 5.14 and this process presumes re-ordering has already occurred prior to defining time for causality. A recursive implementation is always possible even if non-causality is imposed by the re-ordering. The following example illustrates the basic procedure.

EXAMPLE 5.2.3 **[Generalized Cholesky and GDFE for $1+.9\cdot D^{-1}$ Channel.]** The singular $R_{\boldsymbol{x}'\boldsymbol{x}'}$ for the channel's pass space with white input was previously found in Example 5.2.2 as

$$R_{\boldsymbol{x}'\boldsymbol{x}'} = \begin{bmatrix} .5945 & .3649 & -.3285 \\ .3649 & .6715 & .2956 \\ -.3285 & .2956 & .7340 \end{bmatrix} \quad . \tag{5.179}$$

This matrix becomes the desired input autocorrelation. Then Generalized Cholesky proceeds as follows:

```
>> H=(1/sqrt(.181))*[.9 1 0
0 .9 1] =   2.1155    2.3505         0
                 0    2.1155    2.3505
>> [F,L,M]=svd(H)
F=   -0.7071   -0.7071     L= 3.8694         0         0   M= -0.3866   -0.6671    0.6368
     -0.7071    0.7071           0    2.2422         0      -0.8161   -0.0741   -0.5731
                                                             -0.4295    0.7412    0.5158
>> Mt=M(1:3,1:2)  =
   -0.3866    0.6671
   -0.8161    0.0741
   -0.4295   -0.741
>> rxxtt=Mt*Mt' =
    0.5945    0.3649   -0.3285
    0.3649    0.6715    0.2956
   -0.3285    0.2956    0.7340
>> Phibar=lohc(rxxtt(2:3,2:3)) =
    0.7433    0.3451
         0    0.8567
> Phi=Phibar*inv(diag(diag(Phibar))) =
    1.0000    0.4028
         0    1.0000
>> Sx=diag(diag(Phibar))^2 =
    0.5525         0
         0    0.7340
>> A= [rxxtt(1,2:3)*inv(Phi')*inv(Sx)
Phi] =
    0.9000   -0.4475
    1.0000    0.4028
         0    1.0000
>> A*Sx*A'  (another check)   0.5945    0.3649   -0.3285
                              0.3649    0.6715    0.2956
                             -0.3285    0.2956    0.7340
>> A=A*sqrtm(Sx) =
    0.6690   -0.3834
    0.7433    0.3451
         0    0.8567
>> [snrGDFEu, GU, WU, S0, MSWMFU, b,bbar,~] = computeGDFE(H, A, 2, 3)
```

```
snrGDFEu =    5.5427 dB
GU =
    1.0000    0.3459
         0    1.0000
WU =
    0.0802         0
   -0.0521    0.1627
SO =
   13.4725         0
         0    7.1461
MSWMFU =
    0.2535    0.1261
   -0.1648    0.3645
b' =   1.8760    1.4186
bbar =    1.0982
```

This is the same value as obtained previously. The input matrix A is generalized triangular. The GLE output should be ignored when the R_b, which is 2×2 with this Generalized Cholesky example and is not $L_x \times L_x$ (or 3×3) as the calculations internal to computeGDFE assume. Instead, it can be computed recognizing that the R_b^{-1} is 2×2 and the R_f that constructs it can only excite 2 of the 3 dimensions with one unit of energy each.

```
>> Rf=(H*A*(2/3)*A'*H') =
    6.6667    3.3149
    3.3149    6.6667
>> Rbinv=Rf+eye(2) =
    7.6667    3.3149
    3.3149    7.6667
>> sGLEO=diag(diag(Rbinv)) =
    7.6667         0
         0    7.6667
>> snrGLEu=10*log10(det(sGLEO)^(1/(Lx))-1) =    4.6061
```

so roughly 2 dB loss.

5.2.5 The Tonal GDFE

Section 4.7's Vectored DMT/OFDM becomes $\overline{N}$ L_x-dimensional GDFEs on each tone's L_x "spatial" dimensions, again presuming all GDFEs' full synchronization to the same symbol and sampling clocks. Section 4.7 found that each (possibly complex) Vector-DMT/Output channel-output dimension has vector form

$$\boldsymbol{Y}_n = \underbrace{\widetilde{H}_n}_{L_y \times L_x} \cdot \boldsymbol{X}_n + \boldsymbol{N}_n \quad , \; n = 0, ..., \overline{N} - 1 \; . \tag{5.180}$$

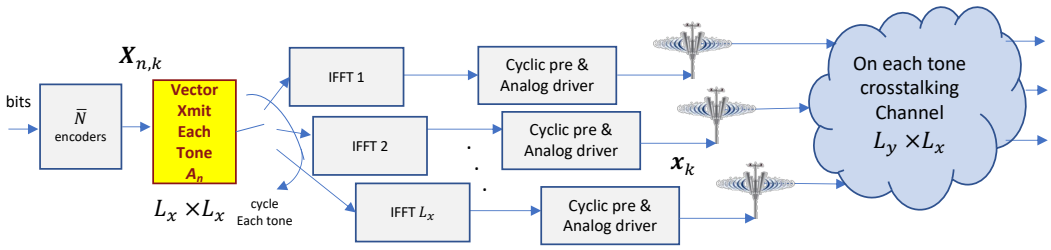


Figure 5.15: Tonal GDFE Transmitter.

The transmission system decomposes into Figures' 5.15 (transmitter) and 5.16's (receiver) "tonal GDFE's." H_n is not in general cyclic, nor nonsingular, but the general GDFE theory applies to it, independently for each n and corresponding $R_{\boldsymbol{xx}}(n)$.

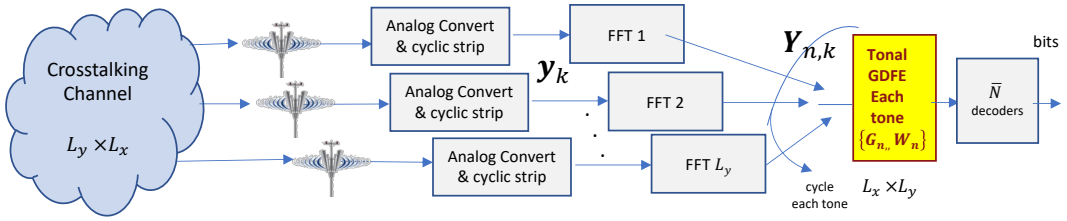


Figure 5.16: Tonal GDFE Receiver.

Singularity can often be avoided by design (each tone's exclusion of input dimensions in the channel model that don't pass to the tone's channel output, i.e. that are in the null-space $\mathcal{N}_n$ for that tone n). Thus, a tonal GDFE is a set of up to $\bar{N}$ separate GDFE's, one for each tone. Tonal GDFE's dominant complexity is L_x^2, while the FFT's used are $\bar{N} \cdot \log_2(\bar{N})$ for each user. Thus total complexity per symbol is the order of $\left(\bar{N} \cdot L_x^2 + L_x \cdot \bar{N} \log_2(\bar{N})\right) / \bar{N} = L_x^2 + \log_2(N)$, which remains low for up to say $L_x \approx log_2(\bar{N})$ spatial dimensions or less. This is usually a large complexity reduction from the general case of $(\bar{N} \cdot L_x)^2$. Such tonal GDFE systems appear in "Massive MIMO" in wireless, as in Sections 2.7 and 2.8. Tonal GDFE's also find use in wireline cables of crosstalking transmission lines.

For each tone, the designer has the choice of using either $G = I$ (vector coding) or $G \neq I$ (triangular GDFE). The triangular GDFE may be necessary because vector coding (with $G = I$) requires coordination (i.e., the vector-coding matrix M_n) on the transmit side, which may not be physically possible (for instance, Chapter 2's MAC). Thus a triangular GDFE for the square H_n may be the only feasible canonical implementation.

5.2.5.1 Tonal GDFE Calculation: using the correct noise-whitened channel equivlent

The Gaussian MAC often has sampled-time-domain specification $\boldsymbol{H}(D) = \boldsymbol{h}_0 + \boldsymbol{h}_1 \cdot D + \boldsymbol{h}_2 \cdot D^2 + ... + \boldsymbol{h}_\nu \cdot D^\nu$. Negative powers on D mean "non-causal" (realized with delay), so no generality is lost by starting at time 0. The $\boldsymbol{h}_k$ are the $L_y \times L_x$ transfer responses in sampled time, indexed by k. The following two matlab commands synthesize the tonal channels

```
h= cat(3, h0 , h1, ..., hnu);
H=fft(h, N, 3)
```

The output H is a $L_y \times L_x \times \bar{N}$ tensor where Matlab's H(:,:,n) would correspond to $\tilde{H}_n$ in (5.180). A GDFE exists for each such tonal channel.

Why is there not a $\sqrt{1/N}$ factor in front of matlab's unnormalized FFT command? The $1/\sqrt{(N)}$ factor that would cause Matlab's FFT command to correspond to a unitary (squared magnitude preserving) transform doe not appear in the above matlab equations. This is correct for a noise-whitened channel, although the reason may not be immediately evident: With fixed sampling rate, $1/T'$, the symbol period increases with $\bar{N}$ as $T = \bar{N} \cdot T'$. Thus the signal and noise energy both increase by a factor of $\bar{N}$ over the longer time period. A random process that has energy 1 per symbol will instead have energy increased by the factor of increase in $\bar{N}$, so if the original system at sampling rate had unit energy for the process, the longer-symbol system now has energy $\bar{N}$. Program inputs should therefore increase input energy/symbol by $\sqrt{N}$ to be correct (energy per sample inputs remain the same).

However, with noise-whitening, there is no explicit noise-energy input. There is a tacit discrete-time convolution of the sampled channel and the noise whitening filter. The product of normalized FFT's then overnormalizes the channel and thus the output must be scaled up by $\sqrt{\overline{N}}$. Matlab's FFT does not need this scaling, but again it is not unitary. The unitary transform is useful to communication designers who use it to preserve input energy/power before/after the modulator processing. This is why the Matlab FFT above is used without scaling because it tacitly includes convolution that corresponds to the FFT's multiplied in succession that represents the noise-normalized/whitening. The following example helps illustrate this effect.

EXAMPLE 5.2.4 *[Computation of a Tonal GDFE Set]* A complex baseband-channel matrix AWGN channel with $\sigma^2 = .01$ has

$$H(D) = \begin{bmatrix} 1+D & -.5-.4D \\ .9-.3D & 1-.9D \end{bmatrix}$$

has

$$\boldsymbol{h}_0 = \begin{bmatrix} 1 & -.5 \\ .9 & 1 \end{bmatrix} \text{ and } \boldsymbol{h}_1 = \begin{bmatrix} 1 & -.4 \\ -.3 & -.9 \end{bmatrix} . \tag{5.181}$$

With 8 tones, the matlab sequence would be

```
>> h0=[ 1 -.5
.9 1];
>> h1=[1 -.4
-.3 -.9];
>> h= cat(3, h0 , h1)
>> h
h(:,:,1) =
    1.0000   -0.5000
    0.9000    1.0000
h(:,:,2) =
    1.0000   -0.4000
   -0.3000   -0.9000
>> N=8;
>> >> H=(10)*fft(h, N, 3)
H(:,:,1) =
  20.0000 + 0.0000i  -9.0000 + 0.0000i
   6.0000 + 0.0000i   1.0000 + 0.0000i
H(:,:,2) =
  17.0711 - 7.0711i  -7.8284 + 2.8284i
   6.8787 + 2.1213i   3.6360 + 6.3640i
H(:,:,3) =
  10.0000 -10.0000i  -5.0000 + 4.0000i
   9.0000 + 3.0000i  10.0000 + 9.0000i
H(:,:,4) =
   2.9289 - 7.0711i  -2.1716 + 2.8284i
  11.1213 + 2.1213i  16.3640 + 6.3640i
H(:,:,5) =
   0.0000 + 0.0000i  -1.0000 + 0.0000i
  12.0000 + 0.0000i  19.0000 + 0.0000i
H(:,:,6) =
   2.9289 + 7.0711i  -2.1716 - 2.8284i
  11.1213 - 2.1213i  16.3640 - 6.3640i
H(:,:,7) =
  10.0000 +10.0000i  -5.0000 - 4.0000i
   9.0000 - 3.0000i  10.0000 - 9.0000i
H(:,:,8) =
  17.0711 + 7.0711i  -7.8284 - 2.8284i
   6.8787 - 2.1213i   3.6360 - 6.3640i
```

The tonal channels are conjugate because the original (matrix) channel was real. Note $n = 1$ corresponds to DC because matlab has positive integer indices. $n = 4$ is the Nyquist tone.

GDFE designs would require a tonal discrete modulator which if it had one unit of energy/sample would reduce to $N/(N+\nu) = 8/9$ because of cyclic prefix. The commands:

```
A=zeros(2,2,8);
for n=1:N
A(:,:,n) = sqrt(8/9)*eye(2);
end
```

generate a spatially white input directly into each and every tone with energy/tone 8/9 instead of 1 because of the cyclic prefix energy loss. The computeGDFE function[35] could be used to generate 8 GDFE's according to

```
cb=1;
Lx=2*(9/8);
GU=zeros(2,2,8);
WU=zeros(2,2,8);
SO=zeros(2,2,8);
MSWMFU=zeros(2,2,8);
b=zeros(2,1,8);
bbar=zeros(1,8)
for n=1:N
[snrGDFEu(1,n), GU(:,:,n), WU(:,:,n), SO(:,:,n), MSWMFU(:,:,n), b(:,:,n), bbar(n),~] = ...
computeGDFE(H(:,:,n), A(:,:,n), cb, Lx);
end
>> snrGDFEu = (in dB)
 16.2546   19.6868   20.8260   19.4629   11.9618   19.4629   20.8260   19.6868
>> GU
GU(:,:,1) =
   1.0000 + 0.0000i  -0.3991 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,2) =
   1.0000 + 0.0000i  -0.2928 + 0.0737i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,3) =
   1.0000 + 0.0000i   0.0931 + 0.1414i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,4) =
   1.0000 + 0.0000i   0.9056 + 0.1552i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,5) =
   1.0000 + 0.0000i   1.5833 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,6) =
   1.0000 + 0.0000i   0.9056 - 0.1552i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,7) =
   1.0000 + 0.0000i   0.0931 - 0.1414i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,8) =
   1.0000 + 0.0000i  -0.2928 - 0.0737i
   0.0000 + 0.0000i   1.0000 + 0.0000i
   >> MSWMFU
MSWMFU(:,:,1) =
   0.0487 + 0.0000i   0.0146 + 0.0000i
  -0.0865 + 0.0000i   0.2821 + 0.0000i
MSWMFU(:,:,2) =
   0.0460 + 0.0191i   0.0186 - 0.0057i
  -0.0409 + 0.0060i   0.0705 - 0.0787i
MSWMFU(:,:,3) =
   0.0366 + 0.0366i   0.0329 - 0.0110i
  -0.0364 - 0.0175i   0.0476 - 0.0370i
MSWMFU(:,:,4) =
   0.0166 + 0.0402i   0.0632 - 0.0120i
  -0.0381 - 0.0564i   0.0431 - 0.0177i
MSWMFU(:,:,5) =
   0.0000 + 0.0000i   0.0884 + 0.0000i
  -0.2792 + 0.0000i   0.0411 + 0.0000i
MSWMFU(:,:,6) =
   0.0166 - 0.0402i   0.0632 + 0.0120i
  -0.0381 + 0.0564i   0.0431 + 0.0177i
MSWMFU(:,:,7) =
   0.0366 - 0.0366i   0.0329 + 0.0110i
  -0.0364 + 0.0175i   0.0476 + 0.0370i
MSWMFU(:,:,8) =
   0.0460 - 0.0191i   0.0186 + 0.0057i
  -0.0409 - 0.0060i   0.0705 + 0.0787i
```

The bit distribution and total bits for each tone are, and then total and per real dimension:

```
>> reshape(b,[2,8])  =
    8.6020    8.4535    8.0156    7.3838    7.0112    7.3838    8.0156    8.4535
    3.6233    6.2959    7.5772    7.1999    2.1297    7.1999    7.5772    6.2959
>> bbar =
 5.4334    6.5553    6.9301    6.4817    4.0627    6.4817    6.9301    6.5553
```

[35] Here the Lx input is changed to include the 9/8 **dimensionality** loss that occurs because of the cyclic prefix.

```
>> sum(b,'all')  =  111.2179
>> sum(bbar)/16 = 3.0894
>> SNRgdfe,u = 10*log10(2^(2*ans)-1)  = 18.5396 dB
```

To obain bits per real dimension, the division was by 16 (because the Lx, which is really repeated for each user) to get bits/real dimension and the cyclic prefix penalty was included on the computeGDFE program input; it could be divided by 8 and then the next line would read only 2^{ans} instead of 2^{2*ans}. Both calculations produce the same SNR (as they should). The input could also use Lx=2, but then the 9/8 adjustment must be done externally. Note the factor of 2×9 to encompass the cyclic prefix of 1 sample (dimension) that is lost to implementation simplification. Also note the biased geometric SNR is the arithmetic average of the *biased* SNRgdfe (not an output) in dB, but the unbiased is really the true measure and is not exactly the arithmetic mean in dB.

5.3 GDFE Transmit Optimization

The GDFE exists for any input autocorrelation matrix $R_{\boldsymbol{xx}}$. There is one best $R_{\boldsymbol{xx}}$ that maximizes the SNR_{GDFE}. This section revisits this best $R_{\boldsymbol{xx}}$'s determination through finite-length symbol partitioning and subsequent water-fill loading. Subsection 5.3.1 returns to vector coding (VC) to compute the best $R_{\boldsymbol{xx}}$ while also revisiting briefly VC's parallel-channel set. Subsection 5.3.2 then constructs this best $R_{\boldsymbol{xx}}$ using the GDFE's triangular white input through Generalized Cholesky Factorization. Subsection 5.3.3 then specifically investigates GDFE loading and the inherent interpolation in the transmit signal construction.

Subsection 5.2.3's Circulant DFE has the same performance as DMT when both have the same circulant H, $R_{\boldsymbol{nn}}$, and $R_{\boldsymbol{xx}}$. This CDFE can also converge to Chapter 3's canonical MMSE-DFE set. Subsection 5.3.4 uses an easily designed DMT system to optimize CDFE performance through two methods – discrete-time sampling-rate interpolation and continuous-time sampling interpolation, finding both in best cases more complex than the original DMT system, but with the same performance at any block size $\overline{N}$ and guard-period ν. Chapter 3's continuous-time symbol-rate and carrier-frequency optimization then transform into Subsection 5.3.3's CDFE with that subsection's generalized-Cholesky interpolation. This section largely considers only $L_x = L_y = 1$, although expansion to vector DMT or Chapter 3's MIMO MMSE-DFEs is straightforward although notationally tedious.

5.3.1 The channel-dependent optimized input

Subsection 5.2.1 showed that VC is a special GDFE case where the set of $\overline{N}^*$ parallel channels correspond to a zeroed feedback section, i.e., $G = I$. The forward and backward canonical channels always have the same mutual information, but since there is no feedback, both zero-forcing (forward) and MMSE (backward) have the same optimal performance on each of the scalar AWGNs generated (that is, ZF and MMSE are the same for VC because the subchannels are each independent **scalar** AWGNs for which zero-forcing and MMSE produce the same result). VC enables direct calculation of a water-filling $R_{\boldsymbol{xx}}$ input, which previous Sections 5.1 and 5.2 presume exists with energies $S_x = \text{Diag}\{\boldsymbol{\mathcal{E}_x}\}$. To compute the water-filling spectrum, $R_{\boldsymbol{vv}}$ must initially relax to be diagonal with variable energy/dimension, as in Chapter 4. The diagonal energy scaling then absorbs into the discrete matrix modulator A. With such a calculated $R_{\boldsymbol{xx}}$, the design removes input singularity (and of course channel singularity, which does not depend on $R_{\boldsymbol{xx}}$) and forms $R_{\boldsymbol{uu}}$. The resultant nonsingular $R_{\boldsymbol{uu}}$ then factors to $R_{\boldsymbol{uu}} = \Phi \cdot \Phi^*$. From this $R_{\boldsymbol{xx}}$, then $\widetilde{H} = R_{\boldsymbol{nn}}^{-1/2} \cdot H \cdot M \cdot S_x^{1/2}$ for vector coding, or more generally from any A such that $A \cdot A^* = M \cdot S_x \cdot M^*$.

VC's diagonalized equivalent forward and backward channel: The VC cases' forward canonical channel has an input $\boldsymbol{\nu}$ with components $v_n/\|v_n\|$ that has identity autocorrelation $R_{\boldsymbol{\nu\nu}} = I$, $R_f = \widetilde{H}^* \cdot \widetilde{H} = M^* \cdot M \cdot \Lambda^* \cdot F \cdot F^* \cdot \Lambda \cdot M^* \cdot M = \Lambda^2$ for any diagonal $R_{\boldsymbol{vv}}$. The forward canonical channel model then is

$$\boldsymbol{z} = R_f \cdot \boldsymbol{\nu} + \boldsymbol{n}' = \Lambda^2 \cdot \boldsymbol{\nu} + \boldsymbol{n}' = \text{diag}(\lambda_n^2 \cdot \frac{v_n}{|v_n|} + n'_n) \quad , \tag{5.182}$$

where $\boldsymbol{n}'$ has autocorrelation matrix $R_{\boldsymbol{n}'\boldsymbol{n}'} = M \cdot \Lambda^2 \cdot M^*$. With $\mathcal{E}_n \triangleq \mathbb{E}\left[|v_n\|^2\right]$, each subchannel has[36] $\text{SNR}_n = \lambda_n^2$, which tacitly absorbs the $\mathcal{E}_n$ energy distribution. Waterfilling thus finds a set of energies $\{\mathcal{E}_{x,n}\}_{n=0,\ldots,\overline{N}^*-1}$ from the original channel $R_{\boldsymbol{nn}}^{-1/2} \cdot H$ and energy constraint $\mathcal{E}_x \leq trace\{R_{\boldsymbol{xx}}\}$. These energies are the diagonal elements of the S_x used to form $\widetilde{H}$. The the new input has energy 1 unit per (real or complex) dimension, so the S_x scaling absorbs into A. The backward channel can proceed then as always to be

$$R_b^{-1} = S_0 = R_f + I = \text{diag}(SNR_n + 1) \quad , \tag{5.183}$$

or

$$R_b = R_{\boldsymbol{ee}} = R_{\boldsymbol{e}'\boldsymbol{e}'} = \text{diag}(\frac{1}{1 + \text{SNR}_n}) \quad . \tag{5.184}$$

[36] Recalling that λ_n includes the effect of the noise normalization in (5.157), the g_n of Chapter 4's loading discussion is $g_n = |\lambda_n|^2$ here. Further, there is a $\mathcal{E}_n^2$ in the signal energy gain, but $\mathcal{E}_n$ in the noise gain, leaving $SNR_n = \lambda_n^2$.

Equation (5.184) validates the relation concerning the (biased) SNR on each GDFE dimension

$$\text{SNR}_{\text{GDFE},n} = R_b = R_{\boldsymbol{ee},n} = R_{\boldsymbol{e'e'},n} = 1 + \text{SNR}_n \quad . \tag{5.185}$$

Each forward subchannel's unbiased SNR in (5.182) is also the corresponding backward channel's unbiased SNR_n, which thus proves VC's forward-and-backward channel equivalence. Thus, VC has $A = M' \cdot S_x^{1/2}$, with M' as the corresponding first $\overline{N}^*$ columns of M. There is similarly a reduced F' that deletes any columns corresponding to zeroed singular values and/or energies.) The best VC structure thus exhibits a unique property within all possible GDFE designs: ZF and MMSE are the same in this very special case. Recall in Chapter 3 that this occurred with nonzero noise only when there is no ISI ($Q(D) = 1$), while in VC it is occurring with ISI overall, but because the MMSE has no ISI component in the special case when channel singular vectors are the discrete modulator, $A = M' \cdot S_x^{1/2}$.

The VC receiver, which does not have a feedback section; essentially it just multiplies $\boldsymbol{y}$ by $\Lambda^{-1}(F')^*$, which is equivalent to the combination of matched filtering by $\tilde{H}^*$, feedforward filtering by $\boldsymbol{S}_0^{-1}$, and removing each dimension's bias:

$$\text{Diag}\left(\frac{\text{SNR}_n + 1}{\text{SNR}_n}\right) \cdot S_0^{-1} \cdot \tilde{H}^* = \tag{5.186}$$

$$= \text{Diag}\left(\frac{\lambda_n^2 + 1}{\lambda_n^2}\right) \cdot \text{Diag}\left(\frac{1}{\lambda_n^2 + 1}\right) \cdot \Lambda \cdot F^*$$

$$= \text{Diag}\left(\frac{1}{\lambda_n^2}\right) \cdot \text{Diag}(\lambda_n) \cdot F^* \tag{5.187}$$

$$= \text{Diag}\left(\frac{1}{\lambda_n}\right) \cdot F^* \tag{5.188}$$

$$= \Lambda^{-1} \cdot F^* \quad . \tag{5.189}$$

This unbiased feedforward section then simply completes the parallelization into uncorrelated dimensions and scales each subchannel by $1/\lambda_n$ (the noise variance scales by $1/\lambda_n^2$ but the path from normalized input $\boldsymbol{v}$ to channel output has gain λ^4 so again $\text{SNR} = \lambda^2$). VC's subchannels usually carry different amounts of information: For any $\mathcal{E}_n$ distribution, and of course including waterfill or any other set the designer so creates,

$$\bar{b}_n = \frac{1}{2} \cdot \log_2\left(1 + \frac{\text{SNR}_n}{\Gamma}\right) \tag{5.190}$$

and $\bar{b}_n = \bar{\mathcal{I}}_n$ when $\Gamma = 0$ dB. Separation-Theorem invocation permits constant constellation size's (not constant mutual information) use with a good code, as with C-OFDM and DMT. Exceptionally important is that in the DMT/VC case, the (unbiased MMSE) error on each subchannel contains no component of the signal, has no need of a feedback section, and ML detection simplifies greatly with only a linear modulator/demodulator, namely to a GDFE.

5.3.1.1 nonzero Gap Inputs

Any GDFE's optimum input code has $\Gamma = 0$ dB. However, VC/DMT systems have an optimum decoder for codes of any gap $\Gamma > 0$ dB because each subchannel is exactly an AWGN. When $G \neq I$, as in the triangular GDFE, then the GDFE error vector contains a (non-Gaussian, since $\Gamma > 0$ dB) channel-input-based component. Correspondingly these alternative GDFEs create different SNR sets even though all minimize MMSE for the same input autocorrelation matrix $R_{\boldsymbol{xx}}$, channel H, and noise $R_{\boldsymbol{nn}}$ and thus have the same mutual information $\mathcal{I}(\boldsymbol{x}; \boldsymbol{y})$. The gap-dependent GDFE overall geometric SNR is

$$\text{SNR}_{\text{GDFE}}(\Gamma) = \Gamma \cdot \left\{ \left[\prod_{i=1}^{\overline{N}^*} \left(1 + \frac{\text{SNR}_n}{\Gamma}\right) \right]^{\frac{1}{\overline{N}_x}} - 1 \right\} \quad . \tag{5.191}$$

Since different choices of the input realization lead to different SNR_n sets, a nonzero gap causes SNR_{gdfe} to vary with the particular set of SNR_n's. All GDFE's **only** have the same $\text{SNR}_{gdfe,u} = 2^{2 \cdot \overline{\mathcal{I}}(\boldsymbol{x}\,\boldsymbol{y})} - 1$

when $\Gamma = 0$ dB (or again good code within any Voronoi shaping region with gap equal to the shaping gain loss). In fact when $\Gamma > 0$ dB, the $\text{SNR}_{\text{GDFE,U}}$ in (5.191) is the largest with the SNR_n's of VC/DMT.

Theorem 5.3.1 [VC/DMT Optimality at nonzero Gap] *Vector coding (which has a special case DMT with cyclic prefix use) has the highest SNR among all GDFE's for the same matrix AWGN channel $\widetilde{H}$ and input when the gap is nonzero, $\Gamma > 0$ dB.*

proof: Because VC has no signal-error component, division by the gap measures each subchannel's performance, as intended with the gap. The overall gap works in exactly the same way for the overall channel as it does for each VC/DMT subchannel. However, any SNR that has a nonzero MMSE signal-dependent component will have that component also experience multiplication by $\Gamma > 0$, which increases the MSE's signal component (but the signal does not increase because the transmit energy remains the same). Thus, the overall computed SNR in (5.191) will be too low on each subchannel, and indeed then the product of such terms is also lower than the exact $\text{SNR}_{gdfe,u} = \Gamma \cdot (2^{2 \cdot \overline{\mathcal{I}}(\boldsymbol{x}\,\boldsymbol{y})} - 1)$ of the VC/DMT system. Further, VC/DMT is exactly an ML detector while the GDFE is not, guaranteeing at least slightly higher VC/DMT performance with $\Gamma > 0$ dB. **QED.**

Theorem 5.3.1 suggests the non-VC GDFE's computed SNR under-estimates its performance when $\Gamma > 0$ dB; acknowledging that the gap is an approximation. However as the proof notes, because the VC/DMT detector is maximum likelihood, and since the SNR accurately represents exactly what is happening on each VC dimension, then no other detector could have better performance. The SNR_{gdfe} directly relates (via the gap) to VC's constant symbol-error probability. However, for other GDFE's with non-ideal codes, the error is not exactly Gaussian for nonzero gaps, and thus the detector is thus no longer exactly ML, and its performance is worse than a VC/DMT detector unequivocally (even if the SNR calculation is somewhat pessimistic and underestimates non-VC/DMT GDFEs' performance). Larger gap also means a larger deviation in computed SNR, causing the underperformance magnitude to increase with Γ.

EXAMPLE 5.3.1 Section 4.6's $1 + .9 \cdot D^{-1}$ DMT example for $N = 8$ and $\Gamma = 0$ dB had $\text{SNR}_{DMT} = 7.6$ dB, which exactly equals the CDFE SNR for that same channel and input in Section 5.2. However, if those examples are reworked with $\Gamma = 8.8$ dB, the two SNR's are then

$$\text{SNR}_{dmt}(\Gamma = 8.8) = 9.5 \text{ dB} > \text{SNR}_{cdfe,u}(\Gamma = 8.8) = 8.6 \text{ dB} \quad , \tag{5.192}$$

a difference of almost 1 dB. These SNR's are higher than the true SNR of 7.6 dB but do not represent higher performance because a gap of 8.8 dB essentially means performance is as if the noise were 8.8 dB larger than with capacity-achieving codes (so performance of the nonzero-gap DMT system would be 9.5-8.8 or 0.7 dB, which is below the earlier example's 7.6 dB when $\Gamma = 0$ dB). Thus (5.192)'s 8.6 dB may be lower than a precise calculation, but overall CDFE performance still cannot exceed DMT's 9.5 dB with $\Gamma = 8.8$ dB.

Coded-OFDM also will not perform as well as the DMT system, because Chapter 4's Separation Theorem loses accuracy as Γ increases. C-OFDM's discrete modulator may use the same C (with good code) on all channels. The ML receiver must weight better dimensions more heavily in decoding, which tends also to confuse the nonzero gap's applicability (an area that yet awaits motivated study).

5.3.2 GDFE SNR Maximization over $R_{\boldsymbol{xx}}$

$R_{\boldsymbol{xx}}$ design first optimizes for VC where $A = M$ and subchannel gains are $g_n = \frac{|\lambda_n|^2}{\sigma^2}$. GDFE optimization then maximizes $\text{SNR}_{\text{GDFE,U}}$

$$\text{SNR}_{\text{GDFE,U}} = 2^{2 \cdot \overline{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})} - 1 \quad . \tag{5.193}$$

which has maximum with water-filling solution

$$\mathcal{E}_n + \frac{1}{g_n} = K \quad , \tag{5.194}$$

where K is the constant determined by either the data rate (MA problem) or energy constraint (RA) problem. In the RA case,

$$\mathrm{SNR}_{\mathrm{GDFE,U}} = 2^{2\cdot\bar{C}} - 1 \quad , \tag{5.195}$$

where $\bar{C}$ is the capacity or maximum value for $\overline{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})$ The number of bits on each subchannel is

$$\bar{b}_n \leq \overline{\mathcal{C}}_n = \frac{1}{2}\cdot\log_2(1+\mathrm{SNR}_n) \quad . \tag{5.196}$$

Again, the water-filling procedure determines $\overline{N}^*$, the optimum number of energized input dimensions. The GDFE design's input-singularity removal eliminates zero-energy dimensions from the input when constructing the canonical forward and backward channels.

The optimum VC autocorrelation is then (with $\boldsymbol{m}$ determined from the SVD of $F\cdot\Lambda\cdot M^* = R_{\boldsymbol{nn}}^{-1/2}\cdot H$ without the input scaling included)

$$R_{\boldsymbol{xx}}(opt) = R_{\boldsymbol{x}'\boldsymbol{x}'}(opt) = \sum_{n=1}^{\overline{N}^*} \mathcal{E}_n\cdot\boldsymbol{m}_n\cdot\boldsymbol{m}_n^* \quad , \tag{5.197}$$

which tacitly reorders indices so that the energized dimensions are $n = 1,...,\overline{N}^*$.

5.3.3 Triangular GDFE $R_{\boldsymbol{xx}}$ Optimization

To optimize the triangular GDFE, the designer first constructs M through SVD of the channel matrix $R_{\boldsymbol{nn}}^{-1/2}\cdot H$, and then computes the water-filling dimensional energies $\mathcal{E}_n$, and constructs the optimum $R_{\boldsymbol{xx}}$ as in (5.197). Other VC input energy distributions may also find use, and this subsection's procedure applies to any nonsingular $R_{\boldsymbol{xx}}$. A non-VC's triangular G implementation can admit recursive (precoder) causal transmitter implementation from $\boldsymbol{v}$ to best $R_{\boldsymbol{xx}}$. Such an implementation may be of interest for several reasons, including recursive time-series-like implementation or for Chapter 2's BC.

The objective is a relationship

$$\boldsymbol{x} = A\cdot\boldsymbol{\nu} \quad , \tag{5.198}$$

where $R_{\boldsymbol{\nu\nu}} = I$, and where A is generalized triangular:

> **Definition 5.3.1** [Generalized Triangular] *An $(\overline{N}+\nu)\times\overline{N}^*$ discrete-modulator matrix A is generalized triangular if the lower $\overline{N}^*$ rows are upper triangular. The remaining rows can be arbitrary.*

A generalized triangular matrix essentially interpolates in a step-by-step causal-like manner from a set of $\overline{N}^*$ white-input dimensions ν_n to a larger set of $\overline{N}+\nu$ discrete-modulator output dimensions, recalling $\nu = \frac{v}{|v|}$.

5.3.3.1 Reordering singular-autocorrelation dimensions

A first $R_{\boldsymbol{xx}}$-optimization step reorders $R_{\boldsymbol{xx}}$'s first $\overline{N}^*$ indices that correspond to the nonsingular information-bearing input dimensions in $\boldsymbol{v}$. The permutation matrix $J_{i,j}$ differs from an identity matrix only in that the 1's that would have been in the i^{th} row and i^{th} column and in the j^{th} row and j^{th} column appear instead in the i^{th} row and j^{th} column, and in the j^{th} row and i^{th} column, respectively.

The product $J_{i,j} \cdot \boldsymbol{x}$ switches the position of the i^{th} and j^{th} elements of the vector $\boldsymbol{x}$. For example, with a 4-dimensional input

$$J_{2,3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.199}$$

and $J_{2,3} \cdot [1234]^* = [1324]^*$. $J_{i,j}$ is symmetric $J_{i,j} = J_{j,i}^* = J_{i,j}^2$, unitary, and $J_{i,j} \cdot J_{i,j}^* = I$. Therefore, or trivially, $|J_{i,j}| = 1$. If $\boldsymbol{x}' = J_{i,j} \cdot \boldsymbol{x}$, then $R_{\boldsymbol{x}'\boldsymbol{x}'} = J_{i,j} \cdot R_{\boldsymbol{x}\boldsymbol{x}} \cdot J_{i,j}$, and $|R_{\boldsymbol{x}\boldsymbol{x}}| = |R_{\boldsymbol{x}'\boldsymbol{x}'}|$.

The submatrix $R_{\boldsymbol{x}\boldsymbol{x}}(i)$, $i = 1, ..., \overline{N}_x - 1$ is the $(i+1) \times (i+1)$ lower-right-hand-corner of $R_{\boldsymbol{x}\boldsymbol{x}}$. The following algorithm reorders $\boldsymbol{x}$'s input dimension index so that its first $\overline{N}^*$ dimensions are non-singular, that is $|R_{\boldsymbol{x}\boldsymbol{x}}(i)| > 0$ for $i = 0, ..., \overline{N}^* - 1$.

1. initialize $i = 0$, $\delta = 1$, $J_A = I_{\overline{N}_x}$.
2. while $|R_{\boldsymbol{x}\boldsymbol{x}}(i)| > 0$
   (a) $i \leftarrow i + 1$
   (b) $J_A \leftarrow J_A$
3. If $i = \overline{N}^*$, exit this algorithm
4. $J_A \leftarrow J_{i,\overline{N}_x - \delta} \cdot J_A$, $R_{\boldsymbol{x}'\boldsymbol{x}'} = J_{i,\overline{N}_x - \delta} \cdot R_{\boldsymbol{x}\boldsymbol{x}} \cdot J_{i,\overline{N}_x - \delta}$
5. $\delta \leftarrow \delta + 1$
6. Go to step 2

This process interpolates the remaining dimensions' entries from the nonzero inputs and constructs the desired covariance for the new $R_{\boldsymbol{x}\boldsymbol{x}}$. Those last $\overline{N}_x - \overline{N}^*$ dimensions are linearly dependent on the first $\overline{N}^*$ dimensions. The first $\overline{N}^*$ components are denoted $\boldsymbol{x}_v$ and the rest are $\boldsymbol{x}_{\bar{v}}$ so

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{\bar{v}} \\ \boldsymbol{x}_v \end{bmatrix} \quad . \tag{5.200}$$

Any discrete modulator A for this newly ordered input must restore order in a final $A \rightarrow J_A \cdot A$, that is the resulting order matrix J_A is absorbed into A for all further GDFE calculations,

$$A \leftarrow J_A \cdot A \quad . \tag{5.201}$$

5.3.4 CDFE with Causal Triangular input

The optimum CDFE spectrum derives from water-filled DMT system with transmit IDFT matrix $\mathcal{Q}^*$ and tonal energy distribution $\mathcal{E}_n$. After sampling-rate adjustment (in each band if there are multiple bands) so that water-filling energizes all tones/dimensions, then this input becomes nonsingular. The designer may then proceed with direct application of Section 5.2.3's triangular or circular GDFEs with this nonsingular water-fill input. Example 5.3.4 later in this subsection takes this approach. The resampling occurs independently for each discontiguous frequency band, which can require complex implementation. A simpler approach uses a digital-interpolation technique that separates the A-matrix "modulation" into a carrier-frequency-and-interpolation part (which is never triangular) and a "causal" (triangular) part. This method finds Chapter 3's independent CDEF bands by essentially extracting those same bands from a DMT design of the same block-length. This subsection first addresses that digital-interpolation approach.

The optimum CDFE first finds a good $\overline{N} \times \overline{N}$ autocorrelation matrix $R_{\boldsymbol{x}\boldsymbol{x}}$ for a cyclic channel H from a DMT system,[37]

$$R_{\boldsymbol{x}\boldsymbol{x}} = \mathcal{Q}_{\overline{N}}^* \cdot R_{\boldsymbol{X}\boldsymbol{X}} \cdot \mathcal{Q}_{\overline{N}} \quad , \tag{5.202}$$

[37] This procedure works for any DMT energy distribution, not just limited to, but of course including, water-filling. Also, $\overline{N} = \overline{N}_x - \nu$ for the CDFE.

where $R_{\boldsymbol{XX}}$ is an $\overline{N} \times \overline{N}$ diagonal matrix containing the DMT input energies on the diagonal. The matrix $\mathcal{Q}_{\overline{N}}$ is $\overline{N} \times \overline{N}$ and implements an FFT. For real-baseband DMT, then (5.202) has $\overline{N}$ replaced by $N = 2\overline{N}$ but the diagonal matrix $R_{\boldsymbol{XX}}$ is such that $R_{\boldsymbol{XX}}(i,i) = R^*_{\boldsymbol{XX}}(N-i, N-i)$ to ensure real time-domain outputs. These vectors $\boldsymbol{X}$ and $\boldsymbol{x}$ reorder in frequency or time from smallest index 0 at the bottom to highest index $\overline{N} = \tilde{\overline{N}}$ at the top. Because of the DMT system used in design and frequency-index ordering, there arise $M \geq 1$ bands that each have ALL nonzero entries, or equivalently set of adjacent nonzero-energy frequency indices for $R_{\boldsymbol{XX}}$. In fact, each such band has $\overline{N}_i$ $i = 1, ..., M$ nonzero input energies and[38]

$$\overline{N}^* = \sum_{i=1}^{M} \overline{N}_i \quad . \tag{5.203}$$

The input's information-bearing part contains nonzero tone inputs $\tilde{\boldsymbol{X}}$, which are interpolated from $\overline{N}^*$ used tones to $\overline{N}$ total tones according to

$$\boldsymbol{X} = J_g \cdot \tilde{\boldsymbol{X}} \quad , \tag{5.204}$$

where J_g has the following structures:

Complex baseband $\boldsymbol{x}$: J_g is an $\overline{N} \times \overline{N}^*$ permutation-and-decimation matrix[39] with no more than one nonzero unit-value entry in each row (and all zeros in $\overline{N} - \overline{N}^*$ of the rows) with structure

$$J_g = \begin{bmatrix} 0_{\overline{N}_{z,M+1} \times \overline{N}_M} & 0_{\overline{N}_{z,M+1} \times \overline{N}_{M-1}} & \cdots & 0_{\overline{N}_{z,M+1} \times \overline{N}_1} \\ I_{\overline{N}_M \times \overline{N}_M} & 0_{\overline{N}_M \times \overline{N}_{M-1}} & \cdots & 0_{\overline{N}_M \times \overline{N}_1} \\ 0_{\overline{N}_{z,M} \times \overline{N}_M} & 0_{\overline{N}_{z,M} \times \overline{N}_{M-1}} & \cdots & 0_{\overline{N}_{z,M} \times \overline{N}_1} \\ 0_{\overline{N}_{z,M-1} \times \overline{N}_M} & I_{\overline{N}_{M-1} \times \overline{N}_{M-1}} & \cdots & 0_{\overline{N}_{z,M-1} \times \overline{N}_1} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{\overline{N}_{z,1} \times \overline{N}_M} & 0_{\overline{N}_{z,1} \times \overline{N}_{M-1}} & \cdots & I_{\overline{N}_1 \times \overline{N}_1} \\ 0_{\overline{N}_{z,1} \times \overline{N}_M} & 0_{\overline{N}_{z,1} \times \overline{N}_{M-1}} & \cdots & 0_{\overline{N}_{z,1} \times \overline{N}_1} \end{bmatrix} \quad , \tag{5.205}$$

Real baseband case: J_g is a more complicated $N \times N^*$ permutation/decimation matrix. Some additional definitions are necessary. First, only band 1 can contain the single-real-dimensional DC frequency and only band M can contain the single-dimensional Nyquist frequency. The remaining bands will all have N_m energized real dimensions and $N_{z,m}$ zeroed real dimensions, $m = 2, ..., M-1$. Each of these bands has corresponding images for dimensions $n > N/2$ that have the same number of zeroed and nonzero real dimensions. However, band $m = 1$ will have $N^+_{z,1}$ zeroed real dimensions and N^+_1 energized real dimensions, one of which will include DC or $n = 0$. Thus, when DC carries no information (whence the superscript of + for "positive frequences"), there will be $N^-_{z,1} = N^+_{z,1} - 1$ image zeros, and otherwise $N^-_{z,1} = N^+_{z,1}$ image-zeroed real dimensions. Similarly N^-_1 will be the number of nonzero image real dimensions and equal to $N^+_1 - 1$ when DC carries information and equal to N^+_1 when DC carries no information. Band M can include Nyquist, either with zero or nonzero energy: this leads to $N^-_{z,M} = N^+_{z,M} + 1$ when Nyquist carries no information and $N^-_{z,M} = N^+_{z,M}$ otherwise. Similarly, $N^-_M = N^+_M + 1$ when Nyquist does carry information and $N^-_{z,M} = N^+_{z,M}$ otherwise. With these definitions J_g for the real baseband case becomes

$$J_g = \begin{bmatrix} J^-_g & 0_{\frac{N}{2} \times \left(N^-_1 + N^-_M + \sum_{m=2}^{M-1} N_m\right)} \\ 0_{\frac{N}{2} \times \left(N^+_1 + N^+_M + \sum_{m=2}^{M-1} N_m\right)} & J^+_g \end{bmatrix} \quad , \tag{5.206}$$

[38] the total number of real dimensions is, as always consistently, $N_i = 2\overline{N}_i$, twice the number of complex tones.
[39] J^*_g reorders and interpolates.

where J_g^- and J_g^+ are respectively defined by

$$J_g^- = \begin{bmatrix} 0_{N_{z,1}^- \times N_1^-} & 0_{N_{z,1}^- \times N_2} & \cdots & 0_{N_{z,1}^- \times N_M^-} \\ I_{N_1^- \times N_1^-} & 0_{N_1^- \times N_2} & \cdots & 0_{N_1^- \times N_M^-} \\ 0_{N_{z,2} \times N_1^-} & 0_{N_{z,2}^- \times N_2} & \cdots & 0_{N_{z,2} \times N_M^-} \\ 0_{N_2 \times N_1^-} & I_{N_2 \times N_2} & \cdots & 0_{N_2 \times N_M^-} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{N_M^- \times N_1^-} & 0_{N_M^- \times N_2} & \cdots & I_{N_M^- \times N_M^-} \\ 0_{N_{z,M}^- \times N_1^-} & 0_{N_{z,1}^- \times N_2} & \cdots & 0_{N_{z,M}^- \times N_M^-} \end{bmatrix} \tag{5.207}$$

and

$$J_g^+ = \begin{bmatrix} 0_{N_{z,M}^+ \times N_M^+} & 0_{N_{z,M}^+ \times N_{M-1}} & \cdots & 0_{N_{z,M}^+ \times N_1^+} \\ I_{N_M^+ \times N_M^+} & 0_{N_M^+ \times N_{M-1}} & \cdots & 0_{N_M^+ \times N_1^+} \\ 0_{N_{z,M-1} \times N_M^+} & 0_{N_{z,M-1} \times N_{M-1}} & \cdots & 0_{N_{z,M-1} \times N_1^+} \\ 0_{N_{M-1} \times N_M^+} & I_{N_{M-1} \times N_{M-1}} & \cdots & 0_{N_{M-1} \times N_1^+} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{N_1^+ \times N_M^+} & 0_{N_1^+ \times N_{M-1}} & \cdots & I_{N_1^+ \times N_1^+} \\ 0_{N_{z,1}^+ \times N_M^+} & 0_{N_{z,1}^+ \times N_{M-1}} & \cdots & 0_{N_{z,1}^+ \times N_1^+} \end{bmatrix} . \tag{5.208}$$

5.3.4.1 IFFT matrices

Since the zero-energy tones between the bands carry no information, a nonsingular input $\boldsymbol{u}$ can be associated with only the information-carrying (nonzero input) tones in two cases as:

- **complex case** M size-$\overline{N}_i$ IFFT's $\mathcal{Q}_{\overline{N}_i}^*$ or
- **real case** M size-N_i conjugate-symmetric IFFT's.

A block FFT matrix operates on each of the non-singular input components individually:

Complex DMT case: The entire set of non-singular-part $\boldsymbol{X}$ components can be represented by

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{X}_M \\ \boldsymbol{X}_{M-1} \\ \vdots \\ \boldsymbol{X}_1 \end{bmatrix} = \begin{bmatrix} \mathcal{Q}_{\overline{N}_M} & 0 & \cdots & 0 \\ 0 & \mathcal{Q}_{\overline{N}_{M-1}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{Q}_{\overline{N}_1, n_1} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_M \\ \boldsymbol{u}_{M-1} \\ \vdots \\ \boldsymbol{u}_1 \end{bmatrix} = \tilde{\mathcal{Q}} \cdot \boldsymbol{u} \quad . \tag{5.209}$$

Real baseband-DMT case: The additional subdivision of an FFT matrix is necessary as

$$\mathcal{Q}_{N_i} = \begin{bmatrix} \mathcal{Q}_{N_i}^- \\ \mathcal{Q}_{N_i}^+ \end{bmatrix} \quad . \tag{5.210}$$

In the real case, band 1's matrix $\mathcal{Q}_{N_1}^+$ is a special case where DC may or may not carry nonzero energy. With nonzero DC energy, no alteration is necessary (and N_1 is necessarily odd). When DC has zero energy (meaning N_1 is even), the corresponding non-singular time-domain input u_k adjusts through multiplication by $e^{\jmath \frac{2\pi}{N_1} k}$ so

$$\mathcal{Q}_{N_1} \rightarrow \mathcal{Q}_{N_1} \cdot \text{diag} \left[e^{\jmath \frac{2\pi}{N_1}(N_1 - 1)} \ \ldots \ e^{\jmath \frac{2\pi}{N_1}(1)} \ 1 \right] \quad . \tag{5.211}$$

Similarly Band M's matrix $\mathcal{Q}_{N_M}^-$ is a special case where Nyquest may or may not carry nonzero energy. When Nyquist carries zero energy, implying even N_M, then no offset is necessary. With nonzero Nyquist

energy (implying that N_M is odd), the alteration multiplies the corresponding nonsingular time-domain input u_k by $e^{\jmath \frac{2\pi}{N_M} k/2}$ (an offset of half a carrier) so

$$\mathcal{Q}_{N_M} \rightarrow \mathcal{Q}_{N_M} \cdot \text{diag}\left[e^{\jmath \frac{\pi}{N_M}(N_M-1)} \; ... \; e^{\jmath \frac{\pi}{N_M}(1)} \; 1 \right] \quad . \tag{5.212}$$

With these possibly modified baseband transforms, then the (correspondingly modified) matrices corresponding to (5.210) become components of the formation

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{X}_1^- \\ \boldsymbol{X}_2^- \\ \vdots \\ \boldsymbol{X}_M^- \\ \boldsymbol{X}_M^+ \\ \vdots \\ \boldsymbol{X}_2^+ \\ \boldsymbol{X}_1^+ \end{bmatrix} = \begin{bmatrix} 0 & 0 & ... & \mathcal{Q}_{N_1}^- \\ \vdots & \vdots & \iddots & \vdots \\ 0 & \mathcal{Q}_{N_{M-1}}^- & ... & 0 \\ \mathcal{Q}_{N_M}^- & 0 & ... & 0 \\ \mathcal{Q}_{N_M}^+ & 0 & ... & 0 \\ 0 & \mathcal{Q}_{N_{M-1}}^+ & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & \mathcal{Q}_{N_1}^+ \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_M \\ \boldsymbol{u}_{M-1} \\ \vdots \\ \boldsymbol{u}_1 \end{bmatrix} = \tilde{\mathcal{Q}} \boldsymbol{u} \quad . \tag{5.213}$$

Also, then for both real and complex cases, understanding that both J_g and $\tilde{\mathcal{Q}}$ are different for these cases,

$$\boldsymbol{X} = J_g \cdot \tilde{\boldsymbol{X}} = J_g \cdot \tilde{Q} \cdot \boldsymbol{u} \quad . \tag{5.214}$$

Autocorrelation formation in complex case: For the complex case, Equation (5.214) leads to a computation $\boldsymbol{u}$'s autocorrelation matrix as

$$R_{\boldsymbol{u}\boldsymbol{u}} = \tilde{\mathcal{Q}}^* \cdot R_{\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}} \cdot \tilde{\mathcal{Q}} \quad . \tag{5.215}$$

Autocorrelation formation in the real baseband case: There is some regrouping of positive and negative frequencies together for each of the bands in forming $R_{\boldsymbol{u}\boldsymbol{u}}$ terms. In either the real or complex cases, if $\boldsymbol{u}$ is a stacked vector of $\overline{N}_i$ baseband-equivalent and decimated time-domain contributions of the different bands. Those contributions are $\boldsymbol{u}_i$ $i = 1, ..., M$, and

$$R_{\boldsymbol{u}\boldsymbol{u}} = \begin{bmatrix} R_{\boldsymbol{u}\boldsymbol{u}}(M) & 0 & ... & 0 \\ 0 & R_{\boldsymbol{u}\boldsymbol{u}}(M-1) & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & ... & 0 & R_{\boldsymbol{u}\boldsymbol{u}}(1) \end{bmatrix} \quad . \tag{5.216}$$

In the real case, it is easier from a notational standpoint to compute the $R_{\boldsymbol{u}\boldsymbol{u}}(i)$ for the i^{th} band according to

$$R_{\boldsymbol{u}\boldsymbol{u}}(i) = \tilde{\mathcal{Q}}_{\overline{N}_i}^* \cdot \begin{bmatrix} R_{\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}}^-(i) & 0 \\ 0 & R_{\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}}^+(i) \end{bmatrix} \cdot \tilde{\mathcal{Q}}_{\overline{N}_i}^- \tag{5.217}$$

Discrete Moduator: The relationship between the modulated channel input $\boldsymbol{x}$ and $\boldsymbol{u}$ is

$$\boldsymbol{x} = \mathcal{Q}^* \cdot \boldsymbol{X} = \mathcal{Q}^* \cdot J_g \cdot \tilde{\mathcal{Q}} \cdot \boldsymbol{u} \quad . \tag{5.218}$$

The M corresponding summed components of $\boldsymbol{x}$ could be called $\boldsymbol{x}_i$, each corresponding to a $\boldsymbol{u}_i$. Then

$$\boldsymbol{x} = \sum_{i=1}^{M} \boldsymbol{x}_i \quad , \tag{5.219}$$

where $\boldsymbol{x}_i$ forms from all other components of $\boldsymbol{u}_{j \neq i} = 0$. Each of these bands' baseband equivalent could appear stationary if considered by themselves. The multiplication of $\boldsymbol{u}$ by the matrix $\mathcal{Q}^* \cdot J_g \cdot \tilde{\mathcal{Q}}$

essentially performs carrier modulation and interpolation (including the construction of the baseband equivalent). The minimal number of bands is M, but it is possible to subdivide a band into adjacent subbands and execute the same procedure. In the limit, if each band is subdivided into its constituent DMT frequencies used in water-filling, the entire modulation reduces to DMT.

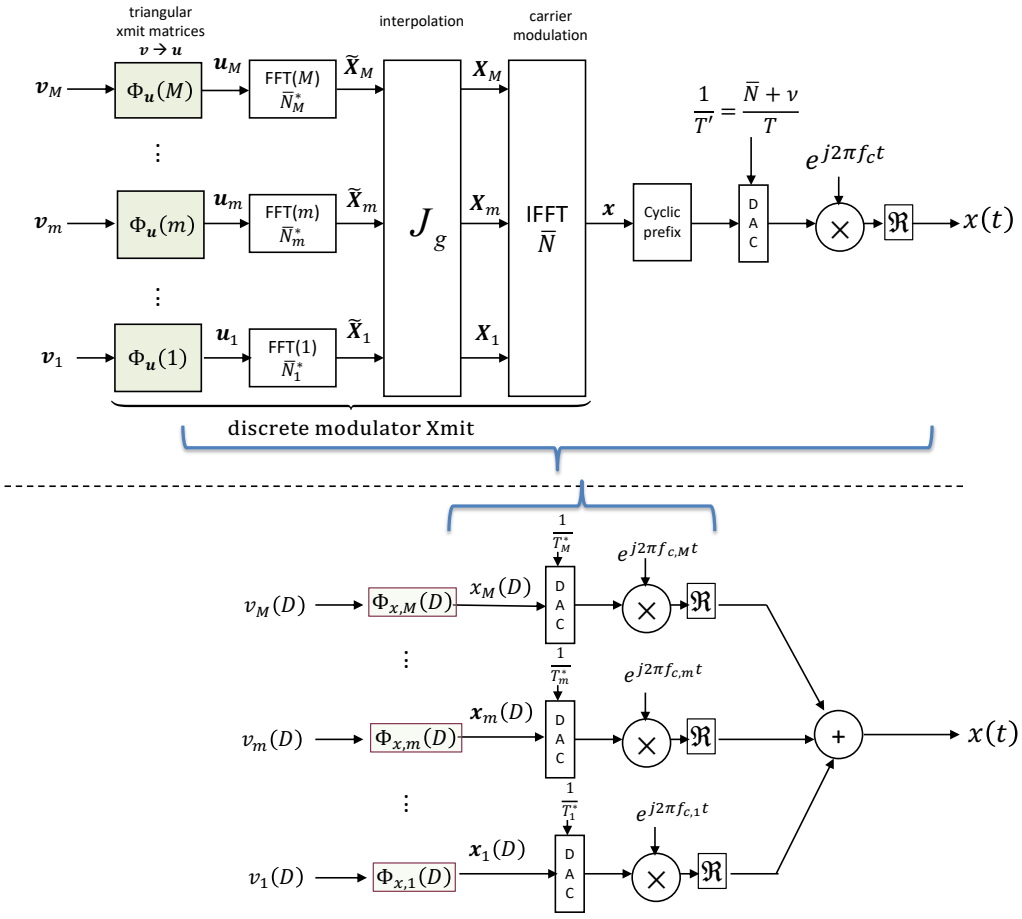


Figure 5.17: Comparison of interpolated CDFE transmitter structure for multiple bands with the infinite-length multi-MMSE DFE transmitter. (FFT's for Bands 1 and M may contain rotation factor for zero DC or nonzero Nyquist in real baseband case.)

The full A Matrix: The non-white inputs $\boldsymbol{u}_i$ relate to white inputs $\boldsymbol{v}_i$ by the usual Cholesky factorization

$$R_{\boldsymbol{uu}} = \Phi \cdot \Phi^* = G_x \cdot S_x \cdot G_x^* \quad , \tag{5.220}$$

or

$$R_{\boldsymbol{uu}}(i) = \Phi(i) \cdot \Phi^*(i) = G_x(i) \cdot S_x(i) \cdot G_x^*(i) \quad , \tag{5.221}$$

decomposing the Cholesky factorization into M such factorizations. Φ is block diagonal with each block upper triangular. The blocks are of different sizes when the N_i are different. The matrix A for CDFE construction is then

$$A = \underbrace{\mathcal{Q}^* \cdot J_g \cdot \tilde{\mathcal{Q}}}_{f_{c,i} \text{ mods}} \cdot \underbrace{\Phi}_{\substack{\text{causal} \\ \text{bb-equiv}}} \quad . \tag{5.222}$$

The Φ will converge to stationary filters in each block-Cholesky factors' middle rows as the individual block lengths $N_i \to \infty$. These are the baseband-equivalent transmit filters of the corresponding M

MMSE-DFE's, each effectively operating at a sampling rate equal to the optimized symbol rate for that MMSE-DFE band.

Any nonsingular input with transmit matrix $\mathcal{Q}$ and $\mathcal{E}_n \neq 0 \; \forall \; n = 1, ..., N$ can be used to construct an acceptable (PWC-satisfying) CDFE input. Such an input necessarily will have a circulant $R_{\boldsymbol{xx}} = \mathcal{Q}^* \cdot \text{diag}\{\mathcal{E}_n\} \cdot \mathcal{Q}$ (the IDFT in this form always produces a circulant matrix). Thus, the CDFE channel input $R_{\boldsymbol{xx}}$ is circulant, and the resultant $\boldsymbol{x}$ represents a sum of contributions from the different bands. As $N \to \infty$, all the diagonal terms of $\mathcal{E}_n \to \mathcal{E}$ will become constant, and the constant rows of $\Phi \to G(D)$ become each bands' DFE feedback section.

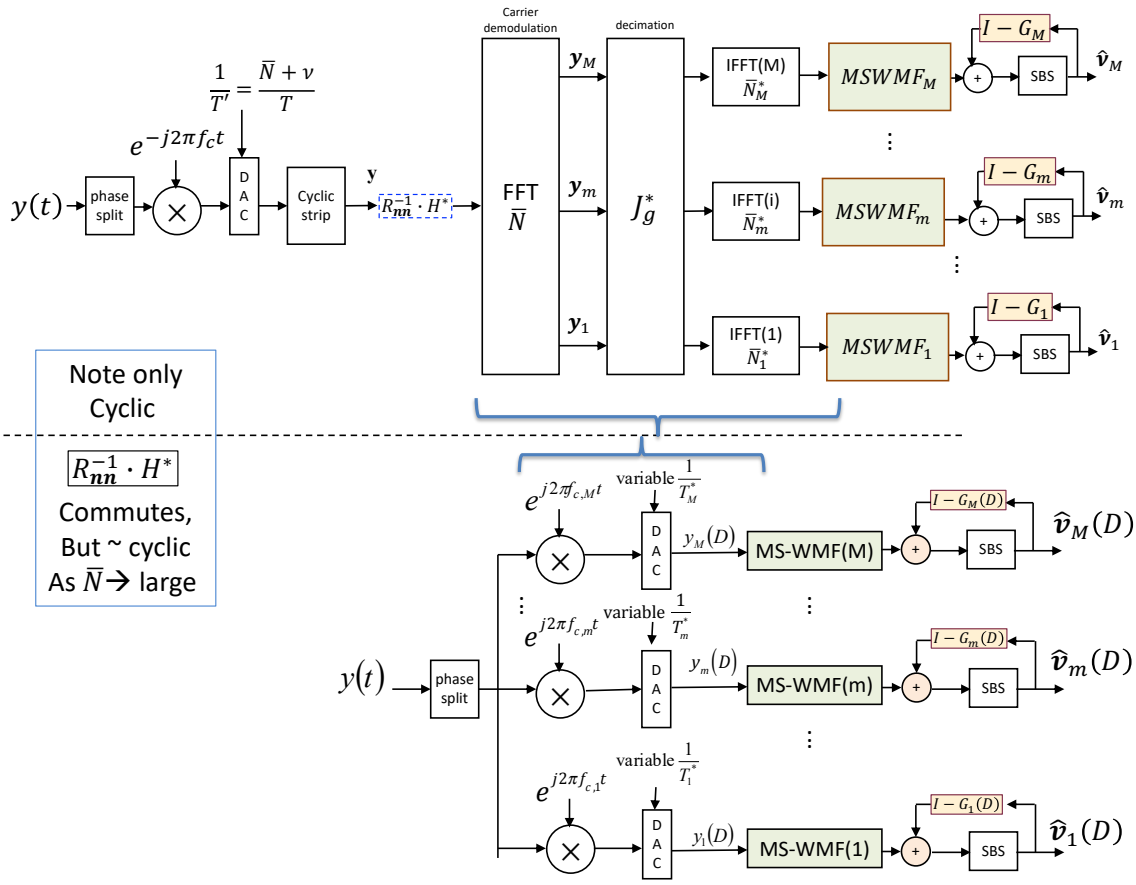


Figure 5.18: Comparison of decimated CDFE receiver structure for multiple bands with infinite-length multi-MMSE DFE receiver. (IFFT's for Bands 1 and M may contain rotation factor for zero DC or nonzero Nyquist in real baseband case.)

Carrier Frequencies and Symbol Rates: Then for complex systems, each band's actual carrier frequencies of each band are (where Δ_i is the first energized tone index of the i^{th} band)

$$f_{c,i} = f_c + \frac{2\pi}{\bar{N}T'} \cdot \left(\frac{\bar{N}_i}{2} + \Delta_i \right) \quad , \tag{5.223}$$

Each band's symbol rate is

$$\frac{1}{T_i^*} = \frac{\bar{N}_i}{\bar{N} \cdot T'} \quad . \tag{5.224}$$

Figures 5.17 and 5.18 compare the CDFE transmitter and receiver respectively with the corresponding limiting structures of the multi-band MMSE-DFE. The $\widetilde{H} \cdot R_{\boldsymbol{nn}}^{-1}$ is cyclic for white noise, but approximately cyclic for non-white noise with large $\bar{N}$. Cyclic matrices commute, but the combination of J_g and the smaller DFT matrices only become effectively cyclic as $\bar{N} \to \infty$.

EXAMPLE 5.3.2 [Cellular's Uplink] Cellular's (that is 3GPP 4G and 5G and probably 6G) uplink reuses a CDFE structure. The CDFE transmitter structure is as in Figure 5.17, where the sizes of the sub-blocks are cellular's 12-tone resource blocks. Each uplink transmit IFFT operates on an integer multiple of 12 input subsymbols (tones). The carrier spacing is $\Delta f = 15$ kHz, and the same number of bits is used on each of the 12 tones (so 2, 4, 6, 8 , ... bits/tone).

Cellular's different uplink users use these different 12-tone groups, but share the same symbol clocks and carrier frequencies (they are synchronized to common clocks at the base station). DSL's zippering used in DSL is not needed in cellular systems that are either frequency-division or time-division multiplexed so that systems with different clocks are separated greatly in frequency or time for uplink and downlink.

Thus, the cellular uplink transmitter has both a $k \cdot 12$ FFT and a larger full IFFT following the interpolation matrix J_g that inserts zeros. Many such synchronized uplink transmitters will have zeros in different places and may use an appropriate IFFT simplification if many of the inputs are zeroed to it (wherever they don't energize a resource block).

The name single-carrier OFDM is somewhat inappropriate in that neither is quite right. The system is circulant and uses basically the CDFE (that was introduced in earlier versions of this text long before any cellular standards committee re-invented the technique and gave it a different name). The $12 \cdot k$ time-domain uses of a QAM input for v_m.

The base station's uplink receiver best uses Figure 5.18's structure with feedback sections. Some implementations may attempt to circumvent the feedback section by using just a feedforward section that attempts elimination of all intersymbol interference. This is similar to C-OFDM's FEQ system, but with an IFFT to return to time domain dimensions and then a linear filter as in Chapter 3's MMSE-LE (or DFE with feedback section included). Clearly such an exclusively linear filter can perform no better than a CDFE system (and performs worse if $G \neq I$).

EXAMPLE 5.3.3 [Real baseband optimized CDFE for 0 dB Gap] The $1 + .9 \cdot D^{-1}$ example 4.6.1 is revisted here with a gap of 0 dB and the CDFE. The following matlab commands illustrate the sequence of computations

```
>> [gn,en_bar,bn_bar,Nstar,b_bar]=DMTra([.9 1],.181,1,8,0)
gn =      19.9448   17.0320   10.0000    2.9680    0.0552    2.9680   10.0000   17.0320
en_bar = 1.2415    1.2329    1.1916    0.9547         0    0.9547    1.1916    1.2329
bn_bar = 2.3436    2.2297    1.8456    0.9693         0    0.9693    1.8456    2.2297
Nstar = 7
b_bar = 1.3814
>> 10*log10(2^(2*b_bar) -1) = 7.6247 dB
>> rXX=diag([en_bar(8:-1:1)]) =
    1.2329         0         0         0         0         0         0         0
         0    1.1916         0         0         0         0         0         0
         0         0    0.9547         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0    0.9547         0         0         0
         0         0         0         0         0    1.1916         0         0
         0         0         0         0         0         0    1.2329         0
         0         0         0         0         0         0         0    1.2415
>> rXXbar=diag([en_bar(8:-1:6) en_bar(4:-1:1)]) =
    1.2329         0         0         0         0         0         0
         0    1.1916         0         0         0         0         0
         0         0    0.9547         0         0         0         0
         0         0         0    0.9547         0         0         0
         0         0         0         0    1.1916         0         0
         0         0         0         0         0    1.2329         0
         0         0         0         0         0         0    1.2415
>> J=hankel([zeros(1,7),1]);
>> Q=(1/sqrt(8))*J*fft(J);
```

```
>> J7=hankel([zeros(1,6),1]);
>> Qtilde=(1/sqrt(7))*J7*fft(J7);
>> ruu=real(Qtilde'*rXXbar*Qtilde) =
    1.1429    0.0755   -0.0377    0.0116    0.0116   -0.0377    0.0755
    0.0755    1.1429    0.0755   -0.0377    0.0116    0.0116   -0.0377
   -0.0377    0.0755    1.1429    0.0755   -0.0377    0.0116    0.0116
    0.0116   -0.0377    0.0755    1.1429    0.0755   -0.0377    0.0116
    0.0116    0.0116   -0.0377    0.0755    1.1429    0.0755   -0.0377
   -0.0377    0.0116    0.0116   -0.0377    0.0755    1.1429    0.0755
    0.0755   -0.0377    0.0116    0.0116   -0.0377    0.0755    1.1429
>> norm(imag(Qtilde'*rXXbar*Qtilde)) =  1.0645e-16
(proves taking real part only for appearance)
>> Phibar=lohc(ruu) =
    1.0625    0.0763   -0.0357    0.0077    0.0159   -0.0400    0.0706
         0    1.0652    0.0738   -0.0352    0.0088    0.0132   -0.0353
         0         0    1.0658    0.0735   -0.0357    0.0101    0.0108
         0         0         0    1.0658    0.0736   -0.0361    0.0108
         0         0         0         0    1.0660    0.0731   -0.0353
         0         0         0         0         0    1.0667    0.0706
         0         0         0         0         0         0    1.0690
>> Phi=Phibar*inv(diag(diag(Phibar))) =
    1.0000    0.0716   -0.0335    0.0072    0.0149   -0.0375    0.0660
         0    1.0000    0.0692   -0.0330    0.0082    0.0123   -0.0330
         0         0    1.0000    0.0690   -0.0335    0.0095    0.0101
         0         0         0    1.0000    0.0691   -0.0338    0.0101
         0         0         0         0    1.0000    0.0685   -0.0330
         0         0         0         0         0    1.0000    0.0660
         0         0         0         0         0         0    1.0000
>> Sx=(diag(diag(Gubar)))*(diag(diag(Gubar))) =
    1.1289         0         0         0         0         0         0
         0    1.1347         0         0         0         0         0
         0         0    1.1360         0         0         0         0
         0         0         0    1.1360         0         0         0
         0         0         0         0    1.1363         0         0
         0         0         0         0         0    1.1379         0
         0         0         0         0         0         0    1.1429
>> Jg = [
     1     0     0     0     0     0     0
     0     1     0     0     0     0     0
     0     0     1     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     1     0     0     0
     0     0     0     0     1     0     0
     0     0     0     0     0     1     0
     0     0     0     0     0     0     1 ];
>> A=real(Q'*Jg*Qtilde*Gubar) =
    0.9690   -0.0430    0.0265   -0.0400    0.0643   -0.1047    0.2044
    0.3040    0.9208   -0.1373    0.0784   -0.0748    0.0937   -0.1427
   -0.1969    0.4609    0.8251   -0.1903    0.1093   -0.0932    0.1060
    0.1576   -0.2170    0.6189    0.6929   -0.2052    0.1182   -0.0938
   -0.1314    0.1408   -0.2126    0.7640    0.5365   -0.1870    0.1060
    0.1064   -0.0937    0.1084   -0.1770    0.8833    0.3691   -0.1427
   -0.0729    0.0515   -0.0489    0.0606   -0.1068    0.9658    0.2044
   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000    1.0000
>> C=[.9
zeros(6,1)
1];
>> R=[.9 1 0 0 0 0 0];
>> H=toeplitz(C,R) =
    0.9000    1.0000         0         0         0         0         0         0
         0    0.9000    1.0000         0         0         0         0         0
         0         0    0.9000    1.0000         0         0         0         0
         0         0         0    0.9000    1.0000         0         0         0
         0         0         0         0    0.9000    1.0000         0         0
         0         0         0         0         0    0.9000    1.0000         0
         0         0         0         0         0         0    0.9000    1.0000
    1.0000         0         0         0         0         0         0    0.9000
>> Ht=(1/sqrt(.181))*H*;
>> [snrGDFEu, GU, WU, SO, MSWMFU,b,bbar] = computeGDFE(Ht, A, 2, 9)
(note use of Nx = 9 because nu = 1, but also A reduced to 7 dimensions)
snrGDFEu =  7.6247 dB
GU =
    1.0000    0.4654   -0.0309   -0.0024    0.0245   -0.0574    0.4464
         0    1.0000    0.5340   -0.0310   -0.0052    0.0327   -0.2178
         0         0    1.0000    0.5510   -0.0307   -0.0091    0.1120
         0         0         0    1.0000    0.5554   -0.0289   -0.0499
         0         0         0         0    1.0000    0.5549   -0.0102
         0         0         0         0         0    1.0000    0.5555
         0         0         0         0         0         0    1.0000
WU =
    0.0776         0         0         0         0         0         0
```

```
   -0.0387    0.0896         0         0         0         0         0
    0.0222   -0.0453    0.0924         0         0         0         0
   -0.0122    0.0257   -0.0470    0.0932         0         0         0
    0.0045   -0.0139    0.0265   -0.0474    0.0933         0         0
    0.0037    0.0046   -0.0139    0.0265   -0.0473    0.0932         0
   -0.0649    0.0279   -0.0055   -0.0122    0.0315   -0.0598    0.1178
S0 =
   13.8924         0         0         0         0         0         0
         0   12.1626         0         0         0         0         0
         0         0   11.8204         0         0         0         0
         0         0         0   11.7315         0         0         0
         0         0         0         0   11.7157         0         0
         0         0         0         0         0   11.7319         0
         0         0         0         0         0         0    9.4913
MSWMFU =
    0.2144    0.0140   -0.0036    0.0019   -0.0022    0.0042   -0.0120    0.1767
    0.0788    0.2646    0.0434   -0.0124    0.0080   -0.0090    0.0157   -0.0972
   -0.0572    0.0191    0.2737    0.0812   -0.0222    0.0152   -0.0179    0.0609
    0.0413   -0.0281   -0.0237    0.2623    0.1233   -0.0296    0.0215   -0.0421
   -0.0318    0.0250   -0.0033   -0.0541    0.2364    0.1663   -0.0321    0.0320
    0.0279   -0.0225    0.0093    0.0179   -0.0731    0.1999    0.2058   -0.0254
   -0.1112    0.0652   -0.0290   -0.0061    0.0501   -0.1141    0.2104    0.1753
  >> b' =    1.8981    1.8022    1.7816    1.7762    1.7752    1.7762    1.6233
>> bbar =    1.3814
```

The limiting transmit filter appears to be $G_{unb} \to 1 + .55 \cdot D - .031 \cdot D^2$, and $W_{unb} \to .093 - .047D^{-1} + .0265D^{-2} - .0139D^{-3}$ on interior rows, but that A does not approach any (obvious) stationary form. As $N \to \infty$, the energy of the components of $\boldsymbol{v}$ become constant. The modulation and interpolation process is not triangular (causal), and will require N^2 operations in the transmitter. Similarly H^* in the receiver, because of its dependence on A, also requires N^2 operations. The complexity of this interpolating system is N^2 no matter how $G_u(D)$, $G(D)$, and $W(D)$ may converge. Example 5.3.4 will avoid the non-causality and A altogether through analog resampling.

The following example provides an illustration an implementation that a second version of the example resamples for the same $1 + .9 \cdot D^{-1}$ channel to the nearly optimum non-singular input for the CDFE on the previously well-known $1 + .9 \cdot D^{-1}$ channel.

EXAMPLE 5.3.4 *[CDFE for $1 + . \cdot D^{-1}$ channel]* Water-fill loading for DMT modulation on the $1 + .9 \cdot D^{-1}$ channel appeared as an example in Section 4.6. There and here, $N = 8$ and $\nu = 1$ with energies

n	0	1	2	3	4	5	6	7
$\mathcal{E}_n$	1.24	1.23	1.19	.96	.96	1.19	1.23	0

Or, equivalently from matlab

```
>> [gn,en_bar,bn_bar,Nstar,b_bar]=DMTra([.9 1],.181,1,8,0)
gn =         19.9448   17.0320   10.0000    2.9680    0.0552    2.9680   10.0000   17.0320
en_bar =    1.2415    1.2329    1.1916    0.9547         0    0.9547    1.1916    1.2329
bn_bar =    2.3436    2.2297    1.8456    0.9693         0    0.9693    1.8456    2.2297
Nstar =     7
b_bar =    1.3814
```

Figure 5.19 helps illustrate the resampling at a lower rate 7/8 instead of the original rate 1.

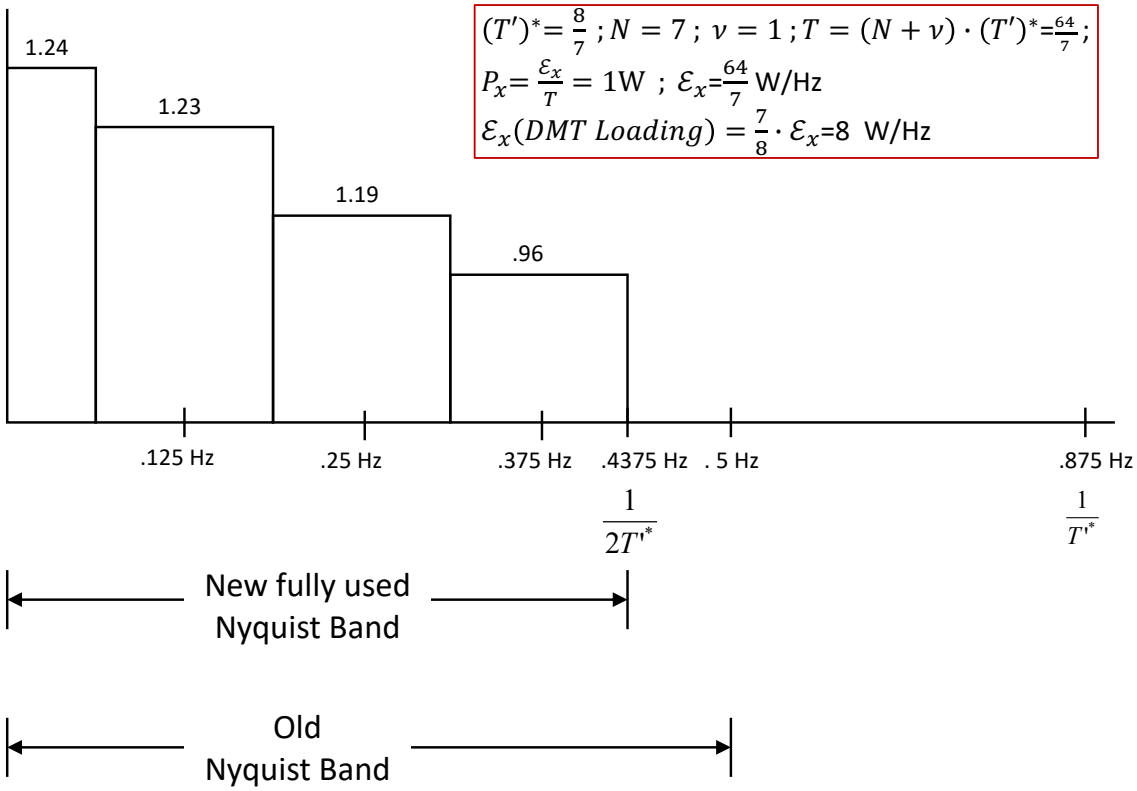


Figure 5.19: Resampled $1 + .9D^{-1}$ channel.

The channel is sampled at rate $1/T^* = 7/8$ or equivalently $T^* = 8/7$ and all tones will now be used. The symbol period is now $64/7$, corresponding to 8 dimensions at spacing $T^* = 8/7$. The DFT size will be $N = 7$ in this case, allowing for one dimension of cylic prefix. Chapter 4's TEQ might be necessary to restore intersymbol interference to one dimension, but since ISI is less than this example's original invocation over a wider band, this revisited example assumes any TEQ loss to be negligible. To maintain the same power, $\mathcal{E}_{\boldsymbol{x}} = 64/7$, but because $1/8$ of this energy is lost in the cyclic prefix, the energy available for DMT loading is 8 $(= \frac{64}{7} \cdot \frac{7}{8})$ units.

The following sequence of matlab commands shows the resampling interpolation:

```
>> D=exp(j*[0:100]*(7/8)*.01*pi);
>> H7=sqrt(8/7)*(ones(1,101)+.9*D);
>> H7=[H7,conj(H7(101:-1:2))];
>> h7=real(ifft(H7));
>> h=[h7(200:201),h7(1:5)] =
   -0.1011    0.9393    1.1979   -0.0603    0.0394   -0.0292    0.0232
>> H=toeplitz([h(1),h(7:-1:2)]',h) =
   -0.1011    0.9393    1.1979   -0.0603    0.0394   -0.0292    0.0232
    0.0232   -0.1011    0.9393    1.1979   -0.0603    0.0394   -0.0292
   -0.0292    0.0232   -0.1011    0.9393    1.1979   -0.0603    0.0394
    0.0394   -0.0292    0.0232   -0.1011    0.9393    1.1979   -0.0603
   -0.0603    0.0394   -0.0292    0.0232   -0.1011    0.9393    1.1979
    1.1979   -0.0603    0.0394   -0.0292    0.0232   -0.1011    0.9393
    0.9393    1.1979   -0.0603    0.0394   -0.0292    0.0232   -0.1011
>> H=sqrt(1/.181)*H;
```

This channel will now have 7 nonzero water-fill input energies allocated to it, as executed by the following matlab commands:

```
>> J7=hankel([zeros(1,6),1]');
>> Q7=(1/sqrt(7))*J7*fft(J7);
>> rXX=diag([1.23,1.19,.96,.96,1.19,1.23,1.24]);
>> rxx=real(Q7'*rXX*Q7)  =
    1.1429    0.0735   -0.0364    0.0115    0.0115   -0.0364    0.0735
    0.0735    1.1429    0.0735   -0.0364    0.0115    0.0115   -0.0364
   -0.0364    0.0735    1.1429    0.0735   -0.0364    0.0115    0.0115
    0.0115   -0.0364    0.0735    1.1429    0.0735   -0.0364    0.0115
    0.0115    0.0115   -0.0364    0.0735    1.1429    0.0735   -0.0364
   -0.0364    0.0115    0.0115   -0.0364    0.0735    1.1429    0.0735
    0.0735   -0.0364    0.0115    0.0115   -0.0364    0.0735    1.1429
```

The designer then relates this input to a diagonal input over the new interpolated band easily because it is nonsingular, via Cholesky factorization.

```
>> Phibar=lohc(rxx);
>> Phi=Phibar*inv(diag(diag(Phibar)));
>> Sx=diag(diag(Phibar))^2 =
    1.1297         0         0         0         0         0         0
         0    1.1352         0         0         0         0         0
         0         0    1.1363         0         0         0         0
         0         0         0    1.1364         0         0         0
         0         0         0         0    1.1366         0         0
         0         0         0         0         0    1.1381         0
         0         0         0         0         0         0    1.1429
 >> A=Phibar;
```

The proximity of the input with resampling to a white input in that Φ is close to a diagonal matrix, converging to $\Phi(D) = 1 + .07D$. The CDFE then follows in the usual manner:

```
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar] = computeGDFE(H, A, 2, 8)
snrGDFEu =    9.1416 dB (higher, but at lower symbol rate)
GU =
    1.0000    0.4783   -0.0492   -0.0074    0.0208   -0.0760    0.4583
         0    1.0000    0.5663   -0.0507   -0.0100    0.0385   -0.2577
         0         0    1.0000    0.5952   -0.0517   -0.0208    0.1470
         0         0         0    1.0000    0.6049   -0.0463   -0.0833
         0         0         0         0    1.0000    0.6042   -0.0105
         0         0         0         0         0    1.0000    0.6074
         0         0         0         0         0         0    1.0000
WU =
    0.0686         0         0         0         0         0         0
   -0.0360    0.0805         0         0         0         0         0
    0.0237   -0.0443    0.0845         0         0         0         0
   -0.0144    0.0287   -0.0471    0.0858         0         0         0
    0.0072   -0.0174    0.0305   -0.0480    0.0862         0         0
    0.0034    0.0070   -0.0173    0.0304   -0.0479    0.0861         0
   -0.0708    0.0336   -0.0072   -0.0147    0.0374   -0.0652    0.1166
S0 =
   15.5671         0         0         0         0         0         0
         0   13.4197         0         0         0         0         0
         0         0   12.8352         0         0         0         0
         0         0         0   12.6530         0         0         0
         0         0         0         0   12.5968         0         0
         0         0         0         0         0   12.6122         0
         0         0         0         0         0         0    9.5762
MSWMFU =
   -0.0173    0.0040   -0.0050    0.0068   -0.0103    0.2054    0.1611
    0.1971   -0.0222    0.0069   -0.0089    0.0125   -0.1032    0.1701
    0.1583    0.2097   -0.0250    0.0095   -0.0127    0.0677   -0.0866
   -0.0808    0.1539    0.2145   -0.0268    0.0118   -0.0445    0.0590
    0.0559   -0.0786    0.1520    0.2166   -0.0283    0.0302   -0.0397
   -0.0401    0.0562   -0.0788    0.1522    0.2163   -0.0258    0.0308
    0.0724   -0.0769    0.0956   -0.1281    0.2194    0.1136   -0.0825
 b' =    0.9313    0.8775    0.8700    0.8691    0.8690    0.8697    0.7970
bbar =    1.6013
>> R=bbar*(7/8) = 1.4718    (bits/sec) - a little better yet
```

Thus, the higher SNR=9.14 dB corresponds to a larger number of bits per (resampled) dimension to obtain the same data rate, or reducing the SNR by the factor of $10 \log_{10}(8/7) =$

1.3 dB produces 7.8 dB, so close to the other design value, but slightly better. In this case, the good DFE performance is obtained by analog resampling rather than by digital interpolation, so differences here are in approximations made. Analog resampling appears simpler here, but such a variable sampling device would have to cover the range of channel bandwidths in practice, often a much more difficult analog design than using the FFT-based interpolation in digital-signal processing.

The upper rows again converge to a nearly constant feedback filter. As $N \to \infty$, these upper rows would converge to the $G_{unb}(D)$ of the MMSE-DFE for this new sampling rate on this channel of $1/T = 7/8$, which looks close to $1 + .6D - .05D^2$, while $W_{unb}(D) \approx .86 - .048 \cdot D^{-1} + .305 \cdot D^{-2} - .017 \cdot D^{-3}$. The combination of matched filter and feedforward matrix must be interpreted (with delay) as non-causal, so viewed about the diagonal, or basically $.056 \cdot D^{-4} - .079 \cdot D^{-3} + .152 \cdot D^{-2} + .216 \cdot D^{-1} + .027 + .01D$ A larger N is necessary to see the types of 10-tap feedforward filters for this example that were found to be necessary in Chapter 3.

Complexity Observations: As $\Gamma \to 0$, the CDFE and DMT perform the same when the CDFE correctly resamples/interpolates and thus both are nonsingular for the same $R_{\boldsymbol{xx}}$. DMT would clearly be preferred, because it has no feedback section and can be implemented with $N \cdot \log_2(N)$ operations as compared to N^2 for the CDFE. Also, DMT has a higher SNR for any nonzero gap, and thus would be better in any practical system where gaps must be greater than 0 dB. The CDFE may see many internal values of the matrix G and the feedforward filter becoming zero or negligible, thus allowing a highly channel-dependent computational reduction. In Example 5.3.4, with 6 feedforward taps and 3 feedback taps, the receiver complexity per 8-dimensional symbol is 48+24 or 72 operations/symbol. The DMT system of same block size has $8 \log_2(8) = 24$ operations/symbol, 3 times less complex. In Example 5.3.3, the CDFE receiver complexity is $1.5N^2 + N$ per symbol, or 104 operations per dimension (but does avoid the resampling, which was not included in the complexity calculation for Example 5.3.4.) In general, any MMSE-DFE that requires more than $log_2(N)$ taps in both feedback and feedforward filters together to approximate the same performance as a DMT system of symbol length N would be more complex to implement. For example, when $N = 16$ even at the higher sampling rate of $1/T = 1$, DMT achieves the maximum of 8.8 dB on the $1 + .9 \cdot D^{-1}$ channel. However, a CDFE or MMSE-DFE clearly requires more than $log_2(N = 16) = 4$ taps to achieve such a high SNR on this channel (this example's CDFE feedforward filter already has more than 4 taps that cannot be ignored at $N = 7$).

In practice, no matter how severe the channel ISI, DMT requires $\log_2(N)$ operations per dimension, while the CDFE would need to have the capability to perform up to N operations per dimension. The CDFE is nearly always more complex than DMT, no matter what the channel for any specific choice of N. If some of the coefficients of CDFE filters are "zero at the end," then in those situations complexity and delay are less than N. In order for this complexity to reduce below the $\log_2(N)$, it is likely the channel has very mild intersymbol interference, for which analog resampling was used to reduce the ISI, but then DMT can use a smaller N also, and still likely get the same performance. No known examples of lower CDFE complexity are known to the author. Thus, on severe ISI channels, the complexity advantage is likely to be large for DMT while as the ISI severity decreases, the complexity advantage diminishes to the case of the AWGN channel with no ISI, in which case both have the same complexity and performance ($\Gamma = 0$ dB, $N = 1$, and $\nu = 0$). The sole possible reason for using a MMSE-DFE might be that with continuous transmission, the length of the equalizer filters might lead to a delay of less than $\overline{N}$, as Example 5.3.4 shows, thus leading with very long block lengths to a shorter delay to get essentially the same SNR – equivalently, there is no "resetting" of the packet to length N necessary in a very long N CDFE as the transmitter and receiver simply ignore block boundaries and use MMSE-DFE continuously. Good codes' $\Gamma = 0$ dB implies infinite coding and decoding delay although with best codes this delay is often on the order of a (GDFE) symbol period if the code is applied across the symbol's dimensions. Basically there is little reason to use single-carrier on any filtered AWGN, which is why most modern transmission systems use DMT/C-OFDM. This recognization took decades to permeate communication engineering, while many viewed a single carrier design with some filters as easier to understand, and thus (erroneously) believed to less complex.

Two-Band Example: A last example shows a 2-band optimization[40]. This example should help designers with any multi-band channel analysis they may encounter in the field as almost all the detailed matrix issues arise in this example.

EXAMPLE 5.3.5 **[CDFE for $1+.5\cdot D+D^2+0.5\cdot D^3$ two-band real baseband channel]** The sampling rate for Figure 5.20's channel with $H(D) = 1 + .5D + D^2 + 0.5D^3$ is 1 MHz, and AWGN with $\frac{\mathcal{N}_0}{2} = 0.1$ nW/Hz. The allowed transmission power is 0.5 mW. The first step of optimized design begins with determination of the water-filling bandwidth for this channel and noise, which initially occurs for a $\Gamma = 0$ dB. The energy per dimension is the power times the sampling period, so $.005 \times 10^{-6} = 0.5$ nW/Hz. With this and $H(D)$ known, DMTra determines a good water-filling solution as follows:

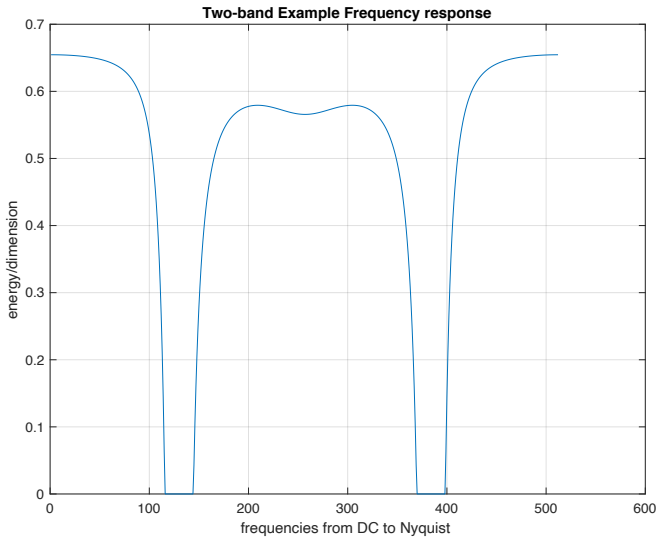


Figure 5.20: Two-band Channel Transfer Function Magnitude.

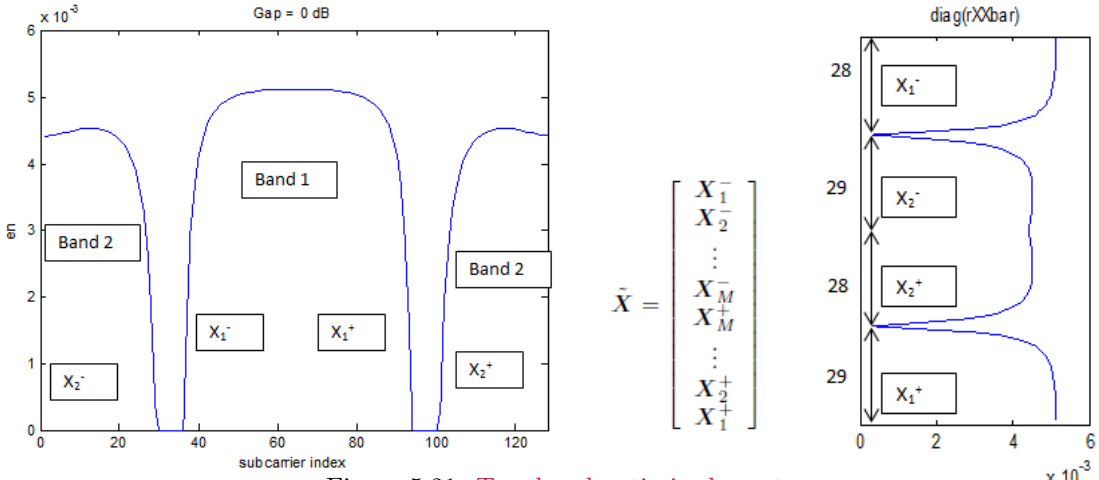
```
>> N=128;
>> [gn,en,bn,Nstar,b_bar,snrdmt]=DMTra([1 0.5 1 0.5],0.1,0.5,N,0);
>> snrdmt = 8.4000 dB
>> b_bar = 1.4926
>> N=256;
>> [gn,en,bn,Nstar,b_bar,snrdmt]=DMTra([1 0.5 1 0.5],0.1,0.5,N,0);
>> snrdmt = 8.5188 dB
>> b_bar = 1.5099
>> N=512;
>> [gn,en,bn,Nstar,b_bar,snrdmt]=DMTra([1 0.5 1 0.5],0.1,0.5,N,0);
>> snrdmt = 8.5792 dB
>> b_bar = 1.5187
```

$\bar{b}$ can be as high as 1.53 bits/dimension asymptotically, but $N = 512$ leads to less than 1% loss at 1.52 Mbps. The next step finds indices for bands.

```
>> find_bands= find(en(1:256) == 0);
>> b1plus_start = 1;
>> b1plus_end = find_bands(1)-1  = 115
>> b1plus_len=b1plus_end - b1plus_start +1 = 115
>> b2plus_start = find_bands(end)+1 = 145
>> b2plus_end = 256;
>> b2plus_len = b2plus_end - b2plus_start +1 = 112
>> find_bands= find(en(257:512) == 0);
>> b1neg_start = 256+find_bands(end)+1 = 399
>> b1neg_end = 512;
>> b1neg_len=b1neg_end - b1neg_start +1 = 114
>> b2neg_start=257;
```

[40]Thanks to former teaching assistant Dr. Haleema Mehmood for providing/correcting the details of this example.

```
>> b2neg_end=256+find_bands(1)-1 =  369
>> b2neg_len=b2neg_end - b2neg_start +1 = 113
>> rXXtilde = diag([ en(512:-1:399) en(369:-1:257) en(256:-1:145) en(115:-1:1)]);
```



$$\bar{X} = \begin{bmatrix} X_1^- \\ X_2^- \\ \vdots \\ X_M^- \\ X_M^+ \\ \vdots \\ X_2^+ \\ X_1^+ \end{bmatrix}$$

Figure 5.21: Two-band optimized spectra.

Next, the example finds the interpolated FFT cascade:

```
>> num_zeros = 29; (must sum to 256)
>> Jgneg=[eye(114) zeros(114,227-114)
zeros(29,227)
zeros(113,227-113) eye(113)];
>> Jgpos=[eye(112) zeros(112,227-112)
zeros(29,227)
zeros(115,227-115) eye(115)];
>> Jg=[ Jgneg zeros(256,227)
zeros(256,227) Jgpos ];
>> JN=hankel([zeros(1,N-1) 1]);
>> Q=1/sqrt(N)*JN*fft(JN);
>> Jb1=hankel([zeros(1,228) 1]);
>> Qtb1=1/sqrt(229)*Jb1*fft(Jb1);
>> Jb2=hankel([zeros(1,224) 1]);
>> Qtb2=1/sqrt(225)*Jb2*fft(Jb2);
>> twid=exp(-sqrt(-1)*(2*pi/225)*(0.5*ones(1,225)).*([0:224])); % no DC offset
>> twid=diag(twid);
>> Qtb2=Qtb2*twid;
>> Qtb1neg=Qtb1(1:114,:);
Qtb1plus=Qtb1(115:229,:);
Qtb2neg=Qtb2(1:113,:);
Qtb2plus=Qtb2(114:225,:);
Qtilde=[ zeros(114,225) Qtb1neg
Qtb2neg zeros(113,229)
Qtb2plus zeros(112,229)
zeros(115,225) Qtb1plus ];
>> ruu = Qtilde'*rXXtilde*Qtilde;
>> norm(imag(ruu)) = 1.4629e-15
>> ruu=real(ruu);
>> size(ruu) =   454   454
>> Jnt = hankel([zeros(1,453),1]);
>> Phibar = lohc(ruu);
>> Phi=Phibar*inv(diag(diag(Phibar)));
>> Sx=diag(diag(Phibar))^2;
>> size(Sx) =
   454   454
>> H=toeplitz([1 zeros(1,508) 0.5 1 0.5 ],[1 0.5 1 0.5 zeros(1,508)]);
>> A=real(Q'*Jg*Qtilde*Phibar);
>> Ht=(1/sqrt(.1))*H;
>> [snrGDFEu, GU, WU, SO, MSWMFU] = computeGDFE(Ht, A, 2, N+3);
>> snrGDFEu =   Inf (numerical issues with large N, compute with logs and sums)
```

```
>> SNR_CDFE = 10*log10(10^(sum(log10(diag(S0)))/(N+3))-1)
SNR_CDFE =    7.1067. dB
>> GU(106:112,106:112) =
    1.0000   -0.2851   -0.2102   -0.0608   -0.0307   -0.0190   -0.0131
         0    1.0000   -0.2852   -0.2101   -0.0608   -0.0307   -0.0190
         0         0    1.0000   -0.2853   -0.2100   -0.0608   -0.0307
         0         0         0    1.0000   -0.2853   -0.2098   -0.0607
         0         0         0         0    1.0000   -0.2853   -0.2097
         0         0         0         0         0    1.0000   -0.2852
         0         0         0         0         0         0    1.0000
>> WU(106:112,106:112) =
    0.2285         0         0         0         0         0         0
    0.0525    0.2263         0         0         0         0         0
    0.0505    0.0522    0.2244         0         0         0         0
    0.0316    0.0503    0.0519    0.2229         0         0         0
    0.0240    0.0315    0.0501    0.0517    0.2217         0         0
    0.0181    0.0240    0.0314    0.0499    0.0516    0.2208         0
    0.0142    0.0181    0.0239    0.0314    0.0498    0.0515    0.2203
```

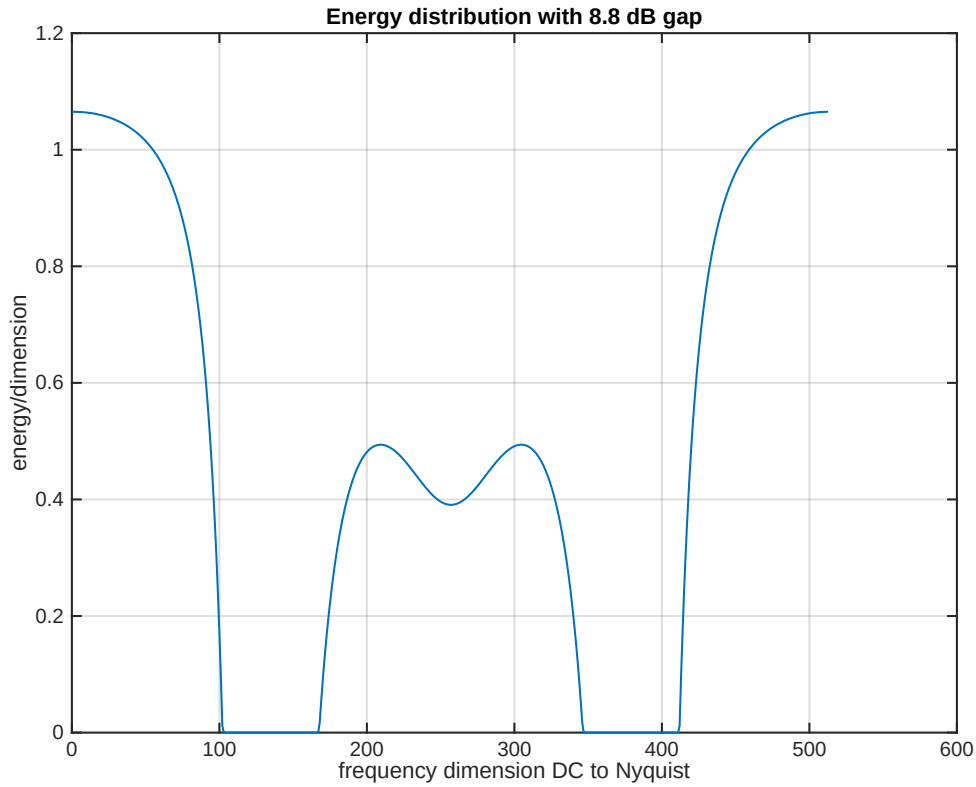


Figure 5.22: Two-band Example with 8.8 dB gap.

```
>>GU(256+114:256+122,256+114:256+122)  =
```

```
    1.0000    0.7224   -0.0151    0.0154   -0.0105    0.0074   -0.0056    0.0043   -0.0035
         0    1.0000    0.7224   -0.0151    0.0154   -0.0105    0.0074   -0.0056    0.0043
         0         0    1.0000    0.7224   -0.0151    0.0154   -0.0105    0.0074   -0.0056
         0         0         0    1.0000    0.7224   -0.0151    0.0154   -0.0105    0.0074
         0         0         0         0    1.0000    0.7223   -0.0151    0.0154   -0.0105
         0         0         0         0         0    1.0000    0.7223   -0.0151    0.0154
         0         0         0         0         0         0    1.0000    0.7223   -0.0151
         0         0         0         0         0         0         0    1.0000    0.7223
         0         0         0         0         0         0         0         0    1.0000
>> WU(256+114:256+122,256+114:256+122) =
    0.0496         0         0         0         0         0         0         0         0
   -0.0341    0.0496         0         0         0         0         0         0         0
    0.0242   -0.0341    0.0496         0         0         0         0         0         0
   -0.0179    0.0242   -0.0341    0.0496         0         0         0         0         0
    0.0137   -0.0179    0.0242   -0.0341    0.0496         0         0         0         0
   -0.0107    0.0137   -0.0179    0.0242   -0.0341    0.0496         0         0         0
    0.0086   -0.0107    0.0136   -0.0179    0.0242   -0.0341    0.0496         0         0
   -0.0070    0.0086   -0.0107    0.0136   -0.0179    0.0242   -0.0341    0.0496         0
    0.0058   -0.0070    0.0086   -0.0107    0.0136   -0.0179    0.0242   -0.0341    0.0496
```

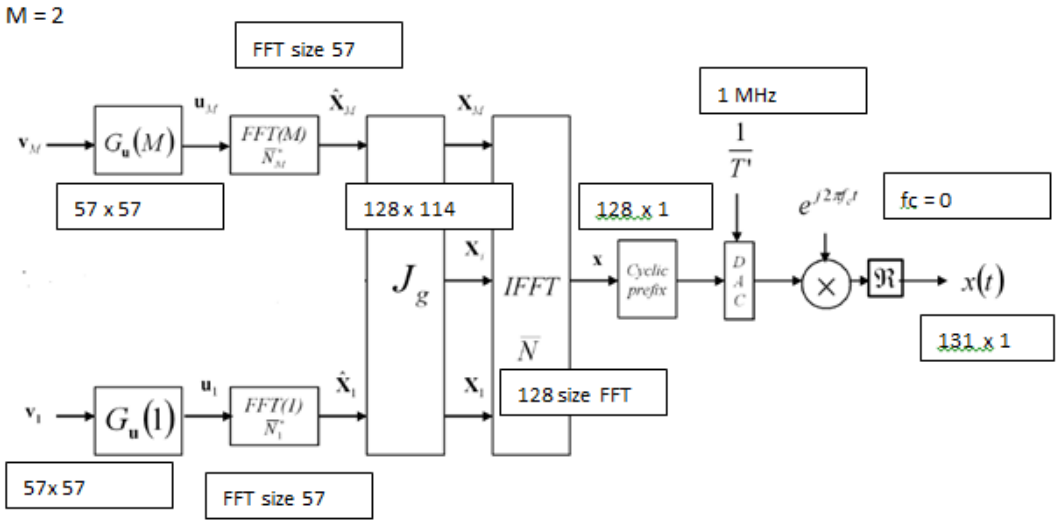


Figure 5.23: Two Band Example Transmitter

The feedback and feedforward sections for bands 2 and 1 (includes DC) are respectively:

$$
\begin{aligned}
G_{2,unb}(D) &= 1 + .722 \cdot D - .0151 \cdot D^2 + .0154 \cdot D^3 - .0105 \cdot D^4 \\
W_{2,unb}(D) &= 0.0496 - .0341 \cdot D^{-1} - 0.0242 \cdot D^{-2} - 0.0179 \cdot D^{-3} + 0.0137 D^{-4} + 0.086 \cdot D^{-5} \\
G_{1unb}(D) &= 1 - 0.285 \cdot D - .208 \cdot D^2 - 0.06 D^3 - .03 \cdot D^4 - 0.0189 D^5 - 0.0130 D^6 \\
W_{1,unb}(D) &= .225 + .052 \cdot D^{-1} + .05 D^{-2} + .0315 \cdot D^{-3} + .024 \cdot D^{-4} + .018 \cdot D^{-5} + .014 \cdot D^{-6}
\end{aligned}
$$

The complexity per dimension is roughly 25-30 multiplies, while for the DMT system it is 9, again about 3 times less complex.

Figure 5.22 shows the two bands with $\Gamma = 8.8$ dB, as in Figures 5.23 and 5.24 detail he transmitter and receiver structures.

Thus, this more realistic example also refuts claims that the designer may encounter that a MMSE-DFE with QAM performs the same, but is less complex. By contrast, a very special effort might make

the performance the same, but in which case the MMSE-DFE is always more complex, although with possibly lower delay.

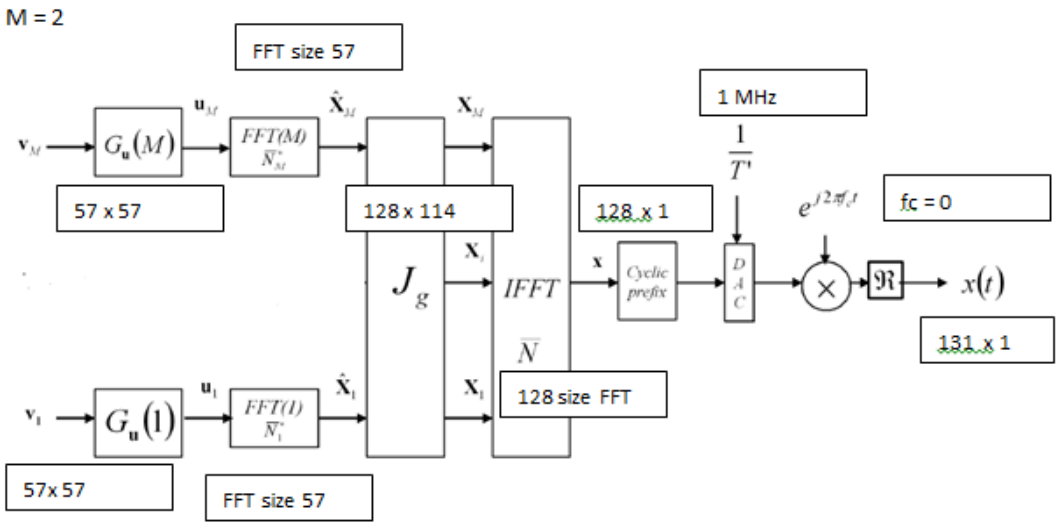


Figure 5.24: Two Band Example Receiver - H. Mehmood

5.3.5 Some Simplifying ZF/MMSE-Equivalence Relationships

This subsection parallels Chapter 3 and also Chapter 2's Section 2.8 on the BC transmitter, finding an interesting relationship between Subsection 5.1.3's ZF and MMSE GDFEs with water-filling and finding their equivalence under worst-case-noise situations with any input. All relations here occur on a nonsingular square channel, which of course is the result of water-filling for the GDFE with all singularity removed. This nonsingular situation also corresponds to Chapter 2's all-primary-user components, and consequently maximum sum-rate is possible with good $R_{\boldsymbol{xx}}$ design for both Section 2.8's BC and Section 2.7's energy-sum MAC. However, in the single-user case here, all primary users just corresponds to energizing only the best dimensions.

Nonsingular Square Channel: This section's results presume a square non-singular equivalent channel $\widetilde{H}$ (so after any channel-singularity elimination and input-singularity elimination). A tacit assumption is that waterfill's $\overline{N}^* = \overline{N}$ (a later comment relaxes this assumption). The white-noise equivalent channel is

$$\boldsymbol{z} = \widetilde{H} \cdot \boldsymbol{u} + \boldsymbol{n} \tag{5.225}$$

where $R_{\boldsymbol{nn}} = I$. SVD, as well as QR factorization, are unambiguous[41] on this non-singular $\widetilde{H}$:

$$\widetilde{H} = D \cdot G_{zf} \cdot Q^* \tag{5.226}$$

$$D^{-1} \cdot \widetilde{H} = F \cdot \Lambda \cdot M^* \quad , \tag{5.227}$$

where D is nonsingular positive-real diagonal matrix, G_{zf} is monic lower triangular, and Q is a unitary matrix.[42] F, Λ, and M are the nonsingular square SVD matrices. K is the water-filling constant such that $\text{diag}(\boldsymbol{\mathcal{E}}) = K \cdot I - \Lambda^{-2}$. Diagonal D^{-1} scaling in the receiver is 1-to-1, simple, does not change performance and simplifies presentation of this subsection's results.

[41] The word "unambiguous" here implies unique to within a possible dimensional reordering that is irrelevant for single-user channels.

[42] The Matlab rq.m at web site can be used.

5.3.5.1 Water-fill Simplifications

Vector coding determines best water-filling input covariance as

$$R_{\boldsymbol{uu}} = M \cdot \left[K \cdot I - \Lambda^{-2}\right] \cdot M^* \quad . \tag{5.228}$$

Removing the channel's Q factor by absorbing Q^* into input $\boldsymbol{v}$ construction uses Cholesky Decomposition of $Q^* \cdot R_{\boldsymbol{uu}} \cdot Q$ to produce

$$Q^* \cdot R_{\boldsymbol{uu}} \cdot Q = \Phi \cdot \Phi^* \quad , \tag{5.229}$$

with $R_{\boldsymbol{\nu\nu}} = I$ and $\boldsymbol{u} = Q \cdot \Phi \cdot \boldsymbol{\nu}$. Φ is upper triangular but not necessarily monic. The zero-forcing cascade then has G_{zf} following Φ, as in Figure 5.25's upper diagram, and needs scaling by diagonal S_{tot} to make G_{tot} monic:

$$G_{tot} = G_{zf} \cdot \Phi \cdot S_{tot} \quad , \tag{5.230}$$

or equivalently after performing the Cholesky factorization in (5.229), the ZF GDFE's overall monic G_{tot} factors S_{tot} to the right in (5.230). This factorization does not (yet) exploit water-filling and could work for any ZF GDFE. Water-fill relates to this Cholesky triangular form through

$$R_{\boldsymbol{uu}} = M \cdot \left[K \cdot I - \Lambda^{-2}\right] \cdot M^* = Q \cdot \Phi \cdot \Phi^* \cdot Q^* \quad . \tag{5.231}$$

Using (5.227) in (5.231):

$$K \cdot I = Q \cdot \Phi \cdot \Phi^* \cdot Q^* + \underbrace{Q \cdot G_{zf}^{-1} \cdot G_{zf}^{-*} \cdot Q^*}_{M \cdot \Lambda^{-2} \cdot M^*} \tag{5.232}$$

$$K \cdot G_{zf} \cdot G_{zf}^* = G_{tot} \cdot S_{tot}^{-2} \cdot G_{tot}^* + I \tag{5.233}$$

$$\underbrace{K \cdot G_{zf} \cdot G_{zf}^*}_{R_b^{-1}} = \underbrace{G_{tot} \cdot S_{tot}^{-2} \cdot G_{tot}^*}_{R_f} + I \quad . \tag{5.234}$$

Equation (5.234) observes the forward channel is on the right with the triangular G_{tot} of the ZF structure.

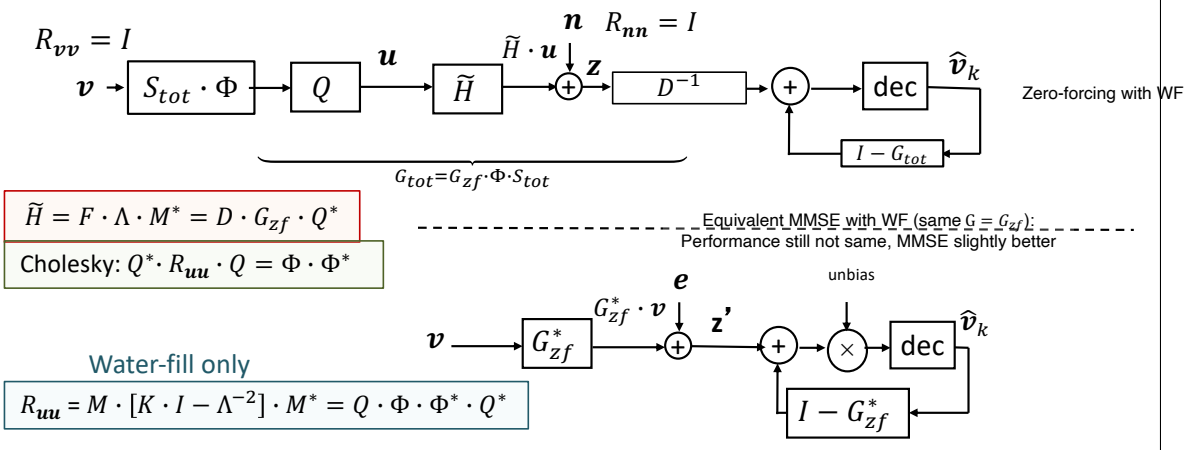


Figure 5.25: Relationship diagram for the ZF-GDFE and GDFE with water-filling input choice.

The left side of (5.234) has a Cholesky Factorization that provides the MMSE GDFE, and indeed the monic upper triangular factor is G_{zf} that follows from the easier QR factorization of the channel $\tilde{H}$. The SVD is necessary only to compute the water-fill energies, but the feedback section basically follows from the QR decomposition and the Cholesky factorization. Figure 5.25 illustrates the ZF-GDFE, which actually has feedback section G_{tot}. When the input is flat so that S_{tot} is a constant diagonal, then

$G_{zf} = G_{tot}$. These two also asymptotically equal one another as $\overline{N} \to \infty$, explaining an infinite-lenght equivalent of this result in Section 3.12. While the MMSE-GDFE feedback section may easily follow directly from RQ factorization, Figure 5.25's feedforward section is NOT the MMSE feedforward section. Indeed the MMSE GDFE still performs (slightly) better.

ZF-GDFE SNRs: Figure 5.25's has corresponding SNRs

$$\text{SNR}_n = D_n^2 \quad . \tag{5.235}$$

Despite the common shape, the ZF-GDFE detector will perform worse than the MMSE-GDFE unless $\Phi = s \cdot I$ (white input), when they are the same. This white input is best for a unitary (all-pass) channel.

5.3.5.2 Worst-case noise simplifications:

Chapter 2, Section 8 for the BC, and also Chapter 3, Section 3.12.8 for continuous frequency/time, found an $R_{\boldsymbol{xx}}$-dependent worst-case noise autocorrelation with fixed (known) diagonal elements of $R_{\boldsymbol{nn}}$. The BC worst-case noise correlation results in a diagonal MMSE GDFE receiver, which now is recognized as that of this section. With singularity removed, the worst-case-noise MMSE GDFE then becomes equivalent to the BC case of all primary users. Worst-case noise applies for any input $R_{\boldsymbol{xx}}$ (not just water-filling as in Subsection 5.3.5.1), and a corresponding discrete-modulator matrix A can be found to synthesize $R_{\boldsymbol{xx}}$ as well as diagonalize the GDFE feedforward processing. In the single user case, the worst-case noise situation allows almost all GDFE complexity to be in the transmitter modulation matrices (including the noiseless precoder). The following lemma also then applies:

> **Lemma 5.3.1** **[Best performance with diagonalized processing]** *For a given $R_{\boldsymbol{xx}}$ and corresponding R_{wcn}, the canonical (best) performing GDFE requires no feedforward signal processing (coordination among receiver dimensions), presuming $\boldsymbol{x}$ and the noise are Gaussian.*
>
> **Proof:** *If a better receiver for any $R_{\boldsymbol{xx}}$ and corresponding R_{wcn} existed with coordinated signal processing, it would be a possible setting for the GDFE considered in Section 2.8's optimization over BC rate sum, and thus would exceed the canonical performance data rate, which would violate basic capacity limits.* **QED.**

Equivalence of ZF and MMSE GDFE's under worst-case noise While Section 2.8's BC design process found best BC design may assume $R_{\boldsymbol{nn}} = R_{wcn}$, the $\mathcal{I}(\boldsymbol{x};\boldsymbol{y})$-minimizing "worst-case" noise for given per user/receiver-noise autocorrelation matrices $R_{\boldsymbol{nn}}(u)$, this did not mean the noise was actually worst-case. If instead, $R_{\boldsymbol{nn}} = R_{wcn}$ is actually true, then the ZF and MMSE designs also perform the same and have the same (unbiased case for MMSE) lossless precoder for any channel H is (square).

This chapter's MMSE-GDFE linear-component receiver design of

$$W_{unb} = (S_0 - I)^{-1} \cdot G^{-*} \cdot R_{\boldsymbol{nn}}^{-1} \cdot H^* \tag{5.236}$$

is the same as Section 2.8's (unbiased) BC receiver(s). $\mathcal{W}_{unb}$ is diagonal when R_{wcn} is the noise autocorrelation[43]. This diagonal receiver is the highest performance (reliably decodable data rate) at $\mathcal{I}_{wcn}$, which was the BC's rate sum and is the single-user data rate here.

The ZF-GDFE uses the worst-case noise and "QR" factors[44] the inverse-noise-weighted nonsingular (block) square channel matrix to obtain

$$\widetilde{H} \triangleq R_{wcn}^{+} \cdot H = R_{ZF} \cdot Q_{ZF}^{*} \quad . \tag{5.237}$$

[43] There is no concern here for sub-users in the single-user case; indeed the user block-diagonal issues also trivially become single dimensions also when H is nonsingular.

[44] This is not the usual square-root-noise-whitened $R_{wcn}^{-1/2} \cdot H$ channel, but actually an "overwhitened" $R_{wcn}^{-1} \cdot H$ causing a deviation in $\widetilde{H}$ definition.

R_1, the upper (nonzero) triangular portion of R_{ZF}, corresponds to columns Q_1 of Q_{ZF}, allowing construction of a filtered input that can be Cholesky factored to a unit-variance vector input:

$$R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} = Q_1^* \cdot R_{\boldsymbol{x}\boldsymbol{x}} \cdot Q_1 = \Phi \cdot \Phi^* \; ; . \tag{5.238}$$

An invertible transmitter modulator matrix $A = Q_{ZF}$ so that of $\boldsymbol{x} = Q_{ZF} \cdot \boldsymbol{v}$ with a lossless precoder of $G = G_{ZF}$ does not change the mutual information, and individual (ML for applied zero-gap code) decoders on each output dimension canonically achieve this data rate. However, **it is possible** that

$$R_{\boldsymbol{x}\boldsymbol{x}} \neq Q_{ZF} \cdot R_{\boldsymbol{v}\boldsymbol{v}} \cdot Q_{ZF}^* \triangleq R_{\boldsymbol{x}\boldsymbol{x}}(ZF) \quad . \tag{5.239}$$

The overall ZF triangular section cascades the upper triangular Φ and R_1 to form

$$D_A \cdot G_{ZF} = R_1 \cdot \Phi \tag{5.240}$$

where D_A is the left-side extracted diagonal that makes G_{ZF} monic. The discrete modulator matrix is then

$$A = Q_1 \cdot \Phi \quad . \tag{5.241}$$

The MMSE-GDFE with this modulator matrix has diagonal receiver forward processing as in Section 2.8.3.

Some Examples of WCN and GDFE

EXAMPLE 5.3.6 [Vector-coded input for WCN] This first example uses the 3×4 channel matrix well known in this text as the $1 + .9 \cdot D^{-1}$ channel. An input with discrete modulator $C = I$ and equal energy on each of the 3 pass modes of this channel is constructed, consequent worst-case noise determined, a GDFE designed, and the overall feedforward-processing determined to be consequently diagonal.

```
>> H =
    0.9000    1.0000         0         0
         0    0.9000    1.0000         0
         0         0    0.9000    1.0000
>> H=(1/sqrt(.181))*H;
>> Rxx=eye(4);
>>  [Rwcn,rate]=wcnoise(Rxx,H,1)
Rwcn =
    1.0000    0.0694   -0.0282
    0.0694    1.0000    0.0694
   -0.0282    0.0694    1.0000
rate =    4.8024
>> Htilde=inv(Rwcn)*H  =
    2.1281    2.2118   -0.0990    0.0785
   -0.1527    1.9670    2.2214   -0.1697
    0.0707   -0.0742    1.9584    2.3645
>> [R,Q]=rq(Htilde);
```

Unlike the multiuser BC case, there is no concern for the rq.m program's dimensional reordering in the single-user case.

```
>> R =
         0   -2.7323   -1.4039    0.0071
         0         0   -2.7077   -1.2346
         0         0         0   -3.0719
>> Q1=Q(:,2:4);
>> Rxxrot=Q1'*Rxx*Q1 =
    1.0000   -0.0000         0
   -0.0000    1.0000    0.0000
         0    0.0000    1.0000
>>  Phibar=lohc(Rxxrot)
    1.0000   -0.0000         0
         0    1.0000    0.0000
         0         0    1.0000
>>  Rup=R(:,2:4) =
   -2.7323   -1.4039    0.0071
         0   -2.7077   -1.2346
```

```
         0         0   -3.0719
>> DA=diag(diag(Rup*Phibar)) =
   -2.7323         0         0
         0   -2.7077         0
         0         0   -3.0719
>> G=inv(DA)*Rup*Phibar
    1.0000    0.5138   -0.0026
         0    1.0000    0.4559
         0         0    1.0000
>> A=Q1*Phibar
   -0.8133    0.0669   -0.0230
   -0.4305   -0.7375    0.0241
    0.3067   -0.5297   -0.6375
   -0.2433    0.4136   -0.7697
>> sri=inv(sqrtm(Rwcn)) =
    1.0022   -0.0357    0.0160
   -0.0357    1.0037   -0.0357
    0.0160   -0.0357    1.0022
>> [snrGDFEu, GU, WU, S0, MSWMFU,b,bbar] = computeGDFE(sri*H, A, 2, 4)
snrGDFEu =    1.1340 dB
GU =
    1.0000    0.5826   -0.0030
         0    1.0000    0.5160
         0         0    1.0000
WU =
    0.1339         0         0
   -0.0676    0.1317         0
    0.0244   -0.0470    0.1031
S0 =
    8.4657         0         0
         0    8.5956         0
         0         0   10.7006
MSWMFU =
   -0.3657   -0.0128    0.0054
   -0.0125   -0.3560   -0.0125
    0.0047   -0.0111   -0.3164
>> b' =    1.5408    1.5518    1.7098
>> bbar =    1.2006
>> sum(b) =    4.8024 (checks)
```

The MSWMFU is not diagonal, but recall the unusual $\widetilde{H}$ definition that effectively adds an additional square-root matrix that the above matrix still includes. To remove it, that factor extracts by multiplying MSWMFU by the square-root (conjugate transpose) inverse above, so

```
>> MSWMFU*sri' =
   -0.3660   -0.0000    0.0000
   -0.0000   -0.3565    0.0000
    0.0000   -0.0000   -0.3167
```

is indeed diagonal and admits the receiver's independent dimensional processing in parallel. The input $R_{\boldsymbol{xx}}$ is rank $\varrho_x = 4$ but the chanel has rank $\varrho_H = 3 < \varrho_x$; but the wcn program finds the correct worst-case noise corresponding to the part of $R_{\boldsymbol{xx}}$ in the pass space. The Φ matrix is trivially an identity because this example's input was initially white - this does not always occur. The inverse square-root of R_{wcn} was matlab's symmetric square root; indeed any square root could be used, but the corresponding correction factor for the full MSWMF (the sri' above) that includes the noise whitening would be the conjugate transpose.

This first discrete modulator above has a diagonal $R_{\boldsymbol{xx}}$ as the input autocorrelation.

The next example illustrates the more general case where the input covariance is more arbitrarily chosen, thus the initial modulation matrix C is not a diagonal nor based on M, but the channel is the same otherwise.

```
>> C=[  1     1     1
    -1     1     1
     1    -1    -1
    -1    -1     1
];
>> Rxx=C*C'
     3     1    -1    -1
     1     3    -3     1
    -1    -3     3    -1
```

```
    -1     1    -1     3
   >> Rxx = (1/3)*(3/4)*Rxx;
>> [Rwcn,rate]=wcnoise(Rxx,H,1)
Rwcn =
    1.0000   -0.0556   -0.1112
   -0.0556    1.0000   -0.0004
   -0.1112   -0.0004    1.0000
rate =    2.7280
>> Htilde=inv(Rwcn)*H;
>> [R,Q]=rq(Htilde)
R =
         0   -2.5801   -1.9449   -0.8193
         0         0   -2.7372   -1.7985
         0         0         0   -3.2336
Q =
    0.5776   -0.8130    0.0048   -0.0739
   -0.5198   -0.3660   -0.7670   -0.0865
    0.4678    0.3905   -0.4282   -0.6673
   -0.4211   -0.2294    0.4778   -0.7360
>> Rup=R(:,2:4);
>> Q1=Q(:,2:4);
>> Rxxrot=Q1'*Rxx*Q1;
>> Phibar=lohc(Rxxrot);
>> DA=diag(diag(Rup*Phibar));
>> G=inv(DA)*Rup*Phibar =
    1.0000   -0.0161   -0.5946
         0    1.0000    0.0095
         0         0    1.0000
>> A=Q1*Phibar =
   -0.8226    0.0383    0.3490
   -0.3704   -0.0139    0.4432
    0.3951   -0.0357   -0.4357
   -0.2321    0.0301   -0.5525
>> sri=inv(sqrtm(Rwcn)) =
    1.0059    0.0281    0.0561
    0.0281    1.0012    0.0026
    0.0561    0.0026    1.0047
>> [snrGDFEu, GU, WU, S0, MSWMFU,b,bbar] = computeGDFE(sri*H, A, 2, 4)
snrGDFEu = 1.9699 dB
GU =
    1.0000   -0.0185   -0.6818
         0    1.0000    0.7633
         0         0    1.0000
MSWMFU =
   -0.3823    0.0107    0.0213
    0.2459   -8.8169    0.0087
    0.0251    0.0004   -0.4497
b'=    1.4832    0.0090  1.2358
bbar =    0.6820
>> 4*bbar =    2.7280 (checks)
>> MSWMFU*sri'
   -0.2051   -0.0000   -0.0000
    0.0000   -4.2034    0.0000
    0.0000    0.0000   -0.2235
```

In this case, Φ is triangular, but not diagonal. However, there still exists an input that diagonalizes the GDFE feedforward processing (including the first noise-whitening step). The adventurous reader might wonder what is largest possible data rate for this channel with worst-case noise maximum input energy 12? This would correspond to the maximum of a BC.

```
>> [Rxx, Rwcn, bmax] = bcmax((3/4)*eye(4), H, 1)
Rxx =
    0.6318    0.3441   -0.2357    0.1800
    0.3441    0.7754    0.2588   -0.1977
   -0.2357    0.2588    0.8127    0.2602
    0.1800   -0.1977    0.2602    0.7800
Rwcn =
    1.0000    0.0811   -0.0352
    0.0811    1.0000    0.0811
   -0.0352    0.0811    1.0000
bmax =    4.8105 bits per 4-dimensional input symbol
>> bmax/4 =    1.2026
--------
check with vector coding:
>> [F,L,M]=svd(H);
L =
    4.1270         0         0         0
```

```
         0    3.1623         0         0
         0         0    1.7228         0
>> gn=diag(L);
>> gn=gn.^2 =
   17.0320
   10.0000
    2.9680
>> [bn, en , Nstar] = waterfill_gn(gn', 1, 0 , 2)
bn =    2.1554    1.7713    0.8950
en =    1.1065    1.0652    0.8283
Nstar =     3
>> sum(bn) =    4.8217
>> sum(bn)/4 =    1.2054
(> bmax/4 = 1.2026 bits/symbol but pretty close meaning that the
water-fill against wcn is only slightly worse than full water fill).
```

The examples emphasize that only for the worst-case noise, no matter what is the input covariance, there always exists an input modulation that diagonalizes the feedforward section of the GDFE. This is independent of "water-filling" or "vector-coding" input selections. That diagonalization also corresponds to a ZF-GDFE solution. Further, the output dimensions not zeroed by such diagonalization would correspond to primary users only if a BC with $L_{y,u} = 1$.

Following from the BC analogy, when both water-fill and worst-case noise simultaneously occur, the data rate is maximum over all receivers that cannot coordinate between dimensions. The bcmax.m program from Section 2.8 and also Appendix G can be used to find this rate.

5.3.6 Asymptotic Stationary Convergence of certain GDFEs

Section 5.2 observes that the CDFE with nonsingular, Toeplitz, circulant H and stationary (circulant also) $R_{\boldsymbol{xx}}$ is very special, and appeared in all examples to converge to Chapter 3's MMSE-DFE as $N \to \infty$. This section further investigates such convergence. The results here are often known in the statistical signal-processing area as arising from "Toeplitz Distribution Theory."

There are 3 GDFE-essential asymptotic results of interest:

1. Toeplitz and cyclic autocorrelation matrices - Subsection 5.3.6.1
2. linear prediction and canonical factorization - Subsection 5.3.6.2
3. Use of 1 and 2 in the GDFE - Subsection 5.3.6.3.

5.3.6.1 Stationary Channels

Strictly speaking, for any finite $\overline{N}$, random vectors like the transmission channel's $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{n}$ in $\boldsymbol{y} = H \cdot \boldsymbol{x} + \boldsymbol{n}$ are never stationary at subsymbol-sample level[45], even when $L_x = L_y = 1$ and there is then no space-time MIMO. These vectors are vector-random processes,, e.g. $\boldsymbol{x} \to \boldsymbol{x}_k$, when $N < \infty$. When the same autocorrelation matrix for each such vector recurs at all discrete-time indices $k \in \mathbb{Z}$, the vector process is **block stationary**, which corresponds to these random-process vectors' time-indexed elements being **cyclo-stationary** with period equal to $\overline{N}_x = \overline{N} + \nu$ dimensions (samples).

However, singular random processes with $\varrho_x < \overline{N}$ are not block nor cyclo-stationary unless the singularity has been removed asymptotically, as in Section 3.11. The GDFE attains an SNR that is canonical in all cases for the block-stationary case, and the data rate of $\overline{\mathcal{I}}$ is only reliably achievable if the transmission system $(R_{\boldsymbol{xx}}, H, R_{\boldsymbol{nn}})$ is block stationary. Equivalently, Chapter 3's MMSE-DFE is the nonsingular (PWC-satisfying) stationary limit as $\overline{N} \to \infty$.

However, for the ISI-specific situation investigated throughout Chapters 3 and 4 and often in this chapter, another subsymbol-level stationarity exists: The subsymbol random processes $\boldsymbol{x}_k$, $\boldsymbol{y}_k$, and $\boldsymbol{n}_k$ are stationary[46]. In this case only, additional GDFE structure occurs that can lead (after all singularity's removal) to (a set of) stationary MMSE-DFE(s). In this case, (possibly block in MIMO case)Toeplitz

[45] The term "stationary" in this chapter means "wide-sense stationarity," and so only means, covariances need be invariant to time.

[46] Often these are scalars $\boldsymbol{x}_k = x_k$, $\boldsymbol{y}_k = y_k$, and $\boldsymbol{n}_k = n_k$; however, there can be vectors of vector-elements to which Section 3.10 applies.

autocorrelation matrices for all processes can occur at each and every block length $\overline{N}$ (or $\overline{N}_x = \overline{N} + \nu$), and convergence to fixed structures occurs as $\overline{N} \to \infty$. This convergence requires care in structuring the successive processing of symbols (or blocks) even when the underlying channel and its input are stationary.

For instance, a scalar FIR channel $H(D)$ with stationary AWGN noise and a stationary input with thus an autocorrelation function $R_x(D)$ may not lead to Toeplitz matrices in partitioning. A easy example is the $1 + .9 \cdot D^{-1}$ channel with $\nu = 1$ that this text often investigates. For this channel H and the usual guard period, which is a Toeplitz $\overline{N} \times \overline{N}_x$ convolution matrix, the channel-output autocorrelation matrix is not Toeplitz, i.e., $R_{\boldsymbol{yy}} \not\to R_y(D)$. For instance, the real baseband case with $N = 2$ and $\bar{\mathcal{E}}_{\boldsymbol{x}} = 1$ yields

$$R_f = \begin{bmatrix} .81 & .9 & 0 \\ .9 & 1.81 & .9 \\ 0 & .9 & 1 \end{bmatrix} + \sigma^2 \cdot I \quad , \tag{5.242}$$

which is not Toeplitz (the diagonal is not constant). As $\overline{N} \to \infty$, this matrix will approach Toeplitz, but it is not Toeplitz for each finite $\overline{N}$. The repeated guard-period insertion thus essentially destroys the stationarity that is otherwise naturally present.

However, with cyclic-prefix uaw, the corresponding $R_{\boldsymbol{yy}}$ becomes

$$R_{\boldsymbol{yy}} = \begin{bmatrix} 1.81 & .9 \\ .9 & 1.81 \end{bmatrix} + \sigma^2 I \quad , \tag{5.243}$$

which is Toeplitz. Indeed, a cyclic prefix of length ν samples when the channel (or TEQ equalized channel) has length of ν or less will always produce a stationary $R_{\boldsymbol{yy}} = H^* \cdot R_{\boldsymbol{xx}} \cdot H + R_{\boldsymbol{nn}}$ as long as $R_{\boldsymbol{nn}}$ is Toeplitz (for instance white), and $R_{\boldsymbol{xx}}$ is not only Toeplitz, but also cyclic. [47] Clearly, the cyclic prefix insures a Toeplitz $R_{\boldsymbol{yy}}$ signal component.[48] Thus, the cyclic prefix has an additional benefit in asymptotic analysis in that Toeplitz matrices appear throughout the GDFE design for all $\overline{N}$, leading to various Cholesky factorizations appearing to converge for smaller $\overline{N}$ than would otherwise be necessary. This was clearly evident Subsection 5.3.4's earlier CDFE examples' convergence as $\overline{N} \to \infty$, while a more general GDFE appeared to have no easy convergence evident (at least the $\overline{N}$ values used in previous examples).

With multiple transmission bands ($M > 1$), the matrix $\widetilde{H} = H \cdot A$ of Section 5.3 leads to a $\widetilde{H}^* \cdot \widetilde{H}$ that is block diagonal. Each block along the diagonal is Toeplitz in this matrix. **Thus, the remarks in this section apply individually to each of these blocks,** and of course cannot apply to the entire multiband-channel matrix, which is not Toeplitz in the multiband case.

5.3.6.2 Canonical Factorization for finite- and infinite-length stationary sequences

Appendix D shows canonical factorization results and the Paley Weiner Criterion for infinite-length random processes and relates a stationary process' asymptotic finite-length Cholesky factorization to linear prediction. The results presume stationarity of all (possibly vector) random processes and thus for instance $R_{\boldsymbol{xx}}$ is (possibly block) Toeplitz.

In the finite-length case, the sequence v_k is the MMSE sequence corresponding to estimating x_k from its past dimensions x_{k-1} ,..., x_0, which follows directly from forming the m^{th}-order linear-prediction error

$$\boldsymbol{v}_{\overline{N}} = \boldsymbol{x}_{\overline{N}} - \boldsymbol{\phi}_1^* \cdot \boldsymbol{x}_{\overline{N}-1} - \ ... \ - \boldsymbol{\phi}_{\overline{N}-1}^* \cdot \boldsymbol{x}_0 \quad , \tag{5.244}$$

and from noting via the orthogonality criterion

$$\mathbb{E}\left[\boldsymbol{v}_{\overline{N}} \cdot \boldsymbol{x}_{\overline{N}-i}^*\right] = 0 \ \forall \ i = 1, ..., k-1 \quad , \tag{5.245}$$

[47] The product of Hermetian cyclic Toeplitz matrices can be shown to be Hermetian cylic Toeplitz - hint, think circular convolution.

[48] In fact, any noise whitening by cyclic $R_{\boldsymbol{nn}}^{-1/2}$ that is done for equivalent channels needs then also for $R_{\boldsymbol{nn}}$ and $R_{\boldsymbol{nn}}^{-1/2}$ to be cyclic – cyclic $R_{\boldsymbol{nn}}$ occurs naturally in the white-noise case, but rarely otherwise. However, nearly all noise is close to cyclic when $N \to \infty$, and because the noise is much smaller than signal in cases where equalization of any type is of interest, the approximation to a cyclic $R_{\boldsymbol{nn}}$ is usually very good. In fact, in DMT systems, noise-whitening is ignored and the SNRs measured directly with whatever noise at each frequency, tacitly assuming the cyclic nature of $R_{\boldsymbol{nn}}$.

when the $\boldsymbol{\phi}_i^*$'s are the entries from the corresponding row (s) of (possibly block) Cholesky factorization with subscript indices that now make the appearance more like a stationary convolutional filter.[49] Clearly then through the linear-prediction problem,

$$\lim_{\overline{N}\to\infty}\left[I\ \boldsymbol{\phi}_1^*\ ...\ \boldsymbol{\phi}_{\overline{N}-1}^*\right] \to \boldsymbol{\Phi}^*(D) \quad , \tag{5.246}$$

because the MMSE solution is unique when the inputs are stationary or equivalently all $R_{\boldsymbol{xx}}(\overline{N})$ as $\overline{N}\to\infty$ are nonsingular.

The diagonal matrix $R_{\boldsymbol{vv}}$ must also become constant as $\overline{N}\to\infty$ because the diagonal values represent the single constant input-energy value

$$\lim_{\overline{N}\to\infty} R_{\boldsymbol{vv}}(i) = \mathbb{E}\left[\boldsymbol{v}_i\cdot\boldsymbol{v}_i^*\right] = S_v = \mathcal{E}_x = \lim_{\overline{N}\to\infty}\frac{1}{\overline{N}}\cdot trace\{R_{\boldsymbol{x}_i\boldsymbol{x}_i}\} \quad \forall\, i\geq 0 \quad .^{50} \tag{5.247}$$

For GDFE designs that use $R_{\boldsymbol{\nu\nu}} = I$, the constant value S_v absorbs into the channel for analysis in previous sections. This result does not hold for a singular process – that means input singularity must be eliminated, or equivalently the PWC must be satisfied. When a water-filling or any other circulant $R_{\boldsymbol{xx}}$ has sets that contain some nonzero DFT values and also contain zeroed values, the consequent singularity must be eliminated as earlier in this section.

Then a set of separate Cholesky filters, each independently acting on its independent inputs, corresponds to and ultimately converges to each band's baseband-equivalent linear prediction filter. Canonical factorization across the whole band does not work for finite length nor for infinite-length when water-filling produces zero-energy bands. The reader may replace v_k by $\boldsymbol{v}_k$ and x_k by $\boldsymbol{x}_k$ for each subsymbol and the results apply to MIMO $\boldsymbol{x}(D)$ throughout this subsection, with Cholesky $\to$ Block Cholesky and a final vector coding on the $L_x\times L_x$ asymptotic $\boldsymbol{x}(D)$, $\boldsymbol{y}(D)$, $H(D)$ as $\overline{N}\to\infty$.

5.3.6.3 Convergence of the Canonical Channel Models and the CDFE

The CDFE must separate into the M distinct bands of the block triangular Φ, creating a set of $R_f(m)$ for $m = 1, ..., M$ when $M > 1$. The following results will not hold otherwise. Each band has a baseband equivalent. Each band's matrix R_f is Toeplitz when $R_{\boldsymbol{nn}}$ is Toeplitz, and H and $R_{\boldsymbol{xx}}$ are circulant. If $R_{\boldsymbol{nn}}$ is circulant[51], then so is R_f; then $R_b^{-1} = R_f + I$ is also a circulant matrix.

CDFE Filter Convergence: The Cholesky factorization of

$$R_b^{-1} = G^*\cdot S_0\cdot G \tag{5.248}$$

corresponds to a inverse-order prediction problem (sometimes called reverse or "backward" linear prediction), or equivalently

$$R_b = G^{-1}\cdot S_0^{-1}\cdot G^{-*} \tag{5.249}$$

corresponds to a normal "forward" linear-prediction problem with G^{-1} itself viewed as the upper-triangular causal matrix that relates computation of sequence values to their components on past innovations input values. In either case, the rows of G and therefore G^{-1} (or vice versa) converge to constant settings. This means the CDFE (or any GDFE that has asymptotically Toeplitz R_b) must then converge to a system with a constant feedback section (including DMT and VC, which have that constant equal to zero at all lengths). This filter is known to be the unique $G(D)$ of the corresponding band of the MMSE-DFE, so then[52]

$$\lim_{\overline{N}\to\infty} \text{ any row of } G = G(D) \quad . \tag{5.250}$$

[49] The statisfaction of (5.245) follows directly from $R_{\boldsymbol{vv}}$ being diagonal and thus v_{k-1} ... v_0 or $\boldsymbol{v}_{k-1}$ are all uncorrelated with v_k and thus so must also be x_{k-1} ... x_0 or $\boldsymbol{x}_{k-1}$ since $\boldsymbol{x}_{k-1} = \Phi(k-1)\cdot\boldsymbol{v}_{k-1}$ from Cholesky factorization of one order lower. The non-singularity of $R_{\boldsymbol{xx}}$ ensures Cholesky factorization of all orders. The use of the conjugate transpose instead of transpose is simple semantics as the optimization is over the quantity and would simply conjugate the ultimate optimum variable's value. In the scalar case they may be removed. For the scalar case, they are unnecessary.

[50] The limit here is taken in the two-sided sense of time going forward and backward to infinity so that is then true for all i.

[51] e.g., $R_{\boldsymbol{nn}} = I$ or anything close to constant-diagonal matrix.

[52] When $L - x > 1$, this becomes an $L_x\times L_x$ $G\to G(D)$ on block rows.

MMSE Convergence: The MMSE factorization must also then converge to a constant

$$\lim_{\overline{N}\to\infty} S_0(i) = s_0 \quad \forall\, i \quad . \tag{5.251}$$

S_0^{-1}'s diagonal entries also represent the GDFE's dimensional MMSE values[53] Thus because S_0 tends to a constant-diagonal matrix, there is then a constant MMSE for the GDFE as $\overline{N} \to \infty$, which is known from Chapter 3 as

$$s_0^{-1} = \frac{\frac{\mathcal{N}_0}{2}}{\gamma_0 \cdot \|h\|^2} \tag{5.252}$$

or

$$s_0 = \frac{\gamma_0 \cdot \|h\|^2}{\frac{\mathcal{N}_0}{2}} \quad . \tag{5.253}$$

SNR Convergence: The feedforward matrix $S_0^{-1} \cdot G^{-*}$ will be thus be anti-causal (lower triangular) as $\overline{N} \to \infty$. The circulant matched filter H^* can be mixed phase, and thus a fixed filter for all $\overline{N}$. Both the matched filter and the feedforward matrix need realization in practice with delay (which is at most $\overline{N} + \nu$ with GDFE's).

The CDFE SNR then converges in each band to

$$\begin{aligned}
\lim_{\overline{N}\to\infty} \mathrm{SNR}_{cdfe} &= \lim_{\overline{N}\to\infty} \left[|R_{\boldsymbol{vv}}| \cdot |S_0|\right]^{N+\nu} && (5.254)\\
&= \frac{S_x}{s_0} && (5.255)\\
&= \frac{\bar{\mathcal{E}}_{\boldsymbol{x}} \cdot \gamma_0 \cdot \|h\|^2}{\frac{\mathcal{N}_0}{2}} && (5.256)\\
&= \gamma_0 \cdot \mathrm{SNR}_{\mathrm{mfb}} && (5.257)\\
&= 2^{2\cdot \overline{\mathcal{I}}(X(D);Y(D)} && (5.258)
\end{aligned}$$

The unbiased SNR is as always $\mathrm{SNR}_{cdfe,u} \to \mathrm{SNR}_{\text{MMSE-DFE,U}} = \mathrm{SNR}_{\text{MMSE-DFE}} - 1$. An overall geometric SNR characterizes the DFE set, as in Section 3.12.

Some Final Comments:

1. All asymptotic results hold ν as a constant (correctly because it is a constant). The corresponding "wasted dimensions" and any energy "wasted" in the cyclic prefix clearly then asymptotically zeroes as block length $\overline{N}$ increases.

2. Error propagation may occur within each symbol in all GDFE's, including the CDFE. Shorter blocks, $\overline{N} \leq \infty$, can thus limit error bursts, while the MMSE-DFE has a potential for infinite catastrophic bursts. With $\Gamma \to 0$ dB, independent codes on each dimension, and infinite decoder delay, error propagation $\to 0$.

3. A designer could pick $\overline{N}$ too large, much larger than is necessary to approach closely infinite-length results, thus resulting in a huge unnecessary complexity for no performance gain. Complexity comparisons where DMT (VC) perform at lower complexity than CDFE (GDFE, respectively) are made in this same good-design context. Obviously, the channel $H(D) = 1$ works equally well with all methods with $\overline{N} = 1$ and $\nu = 0$. As ISI increases, there is a judicious choice of $\overline{N}$ that allows infinite-length performance without infinite-complexity in either DMT or CDFE. At this choice, DMT complexity is $\log_2(\overline{N})$ per sample while CDFE (or MMSE-DFE) complexity is upper bounded by $\overline{N}$ and typically between $\log_2(\overline{N})$ and $\overline{N}$, typically about $3 \cdot \log_2(\overline{N})$ ignoring modulation or interpolation complexity unique to CDFE, and about $5 \cdot \log_2(\overline{N})$ if those are included.

[53] S_0 becomes an $L_x \times L_x$ matric to which VC cam be appiied, or simply absorbed into the matrix factors of $G(D)$, leaving a diagonal $L_x \times L_x$ matrix S_o.

4. Any gap greater than 0 dB leads to the CDFE having a lower SNR than DMT because the CDFE is designed and optimized for 0 dB gap, while the DMT (because the feedback section is 0) simultaneously minimizes MSE, as well as maximizes product SNR, at all gaps. Beware of statements to the opposite regarding an "SNR averaging" property of the DFE – correct use of codes and gaps deviates from this purported property (and it is misunderstanding in codes' use that leads to many misconceptions in performance equivalence).

5. DMT is a maximum-likelihood detector and is thus both canonical and optimal. The CDFE is only canonical and does not have ML-detector performance, and as well may have error propagation.

6. With careful modulation and interpolation implementation (not shown in this text), some designs may reduce delay from input to output in a CDFE system with respect to DMT.

7. C-OFDM achieves the performance levels of DMT or CDFE only when the code has zero dB gap and the $R_{\boldsymbol{xx}}$ is the same. This result has significance in time-varying channels (i.e., some wireless applications) where it is not possible to convey to the transmitter the best transmit spectrum to use either because the channel is broadcast only (unidirectional, so no return channel to advise transmitter design) or because the channel changes so fast that the input dynamic spectra could not follow the channel. Thus, a fixed $\mathbb{R}_{\boldsymbol{\nu\nu}} = I$ has a certain $\overline{\mathcal{I}}$ that corresponds a certain performance level (in the absence of channel singularity) of both OFDM and MMSE-DFE systems if both use $\Gamma = 0$ dB codes. The $\mathcal{I}$ will need to be averaged over all channels and thus bound the data rate of the code[54]. Thus, in time-varying channels, there can be equivalent performance of Coded-OFDM system and a QAM-MMSE-DFE system, other than possibly the receiver's ability to track channel changes (which is likely going to favor the multicarrier system for which each band is separately equalized by a single coefficient, rather than a complicated matrix system with many more coefficients (N^2 versus N). For this reason, most wireless systems use C-OFDM instead of DMT.

8. DMT systems do not require an optimum transmit filter matrix, and use the IFFT – a known and fixed transmit partitioning system that can be designed once. CDFE systems alone use an optimum transmission filter (as well as $R_{\boldsymbol{xx}}$ that is common to DMT and CDFE) that is a consequence of the channel and thus must be implemented adaptively. MIMO VC systems, however, will need to specify a transmit matrix, at least one for each L_x-dimensional tone.

9. High-performance CDFE systems often exhibit the same "Gaussian-like" high peak-to-average ratio of DMT systems, this is particularly true as ISI becomes severe, or equivalently with large $\overline{N}$. Simply put, this is a function of the "central limit theorem" and the transmit signal being simply a sum of many independent random variables' contributions, and thus approaches Gaussian. PAR-reduction methods, such as those of Section 4.10 for DMT and OFDM, have not been studied for CDFE or MMSE-DFE (which, when well designed, share the high PAR problem). Such studies should investigate at least 4x the sampling rate of the transmission system to be of practical significance (some researchers forget that it is the analog PAR that is important, not the sample-time-only PAR), see Section 4.10.

10. The MMSE-DFE MS-WMF notching ability that occurs when the noise-equivalent ISI channel has notches is absolutely NOT equivalent to the use of multiple bands. This statement appears throughout the literature on MMSE-DFEs and is simply incorrect every single time it has been said. A single MMSE-DFE will perform worse than multiple MMSE-DFEs on each side of all the notches - no exceptions.

Ideal MMSE-DFE is lower bound for ML Performance The ideal MMSE-DFE (or ideal MMSE-GDFE) upper bounds the performance P_e of a maximum likelihood detector in that the sequence-error probability is always lower for ML. The ideal MMSE-DFE (and GDFE) assume(s) that all previous decisions are correct. Such correct previous-decision making may not be the case in practice, so at least

[54] This presumes the channel-gain distribution $g = |h|^2/\sigma^2$, p_g, is stationary.

there is a chance this ideal GDFE system's analysis could outperform the optimum detector because of the potentially false presumption of no decision errors. If results include error propagation; this theoretical anomaly cannot occur.

In short, Claude Shannon had it right as with so many other of his basic simple and information-age-envisioning results – a multi-carrier system is optimum for handling ISI.

5.4 The Gaussian MAC and GDFE

The GDFE's architecture, with mostly receiver processing, well matches the MAC's separate user locations. Namely, GDFE's apply directly to Chapter 2's Gaussian MAC's (block) diagonal $R_{\boldsymbol{xx}}$. Section 2.7's MMSE MAC thus has an additional GDFE interpretation, in which the MAC's transmitter input, $\boldsymbol{\nu}$ is the GDFE input with $\mathbb{R}_{\boldsymbol{\nu\nu}} = I$. That is $\boldsymbol{\nu}$ excites block diagonal $R_{\boldsymbol{xx}}^{1/2}$, equivalently $\boldsymbol{x}_u = R_{\boldsymbol{xx}}^{1/2}(u) \cdot \boldsymbol{\nu} = A_u \cdot \boldsymbol{\nu}$. Any such (block) diagonal channel input $R_{\boldsymbol{xx}}$ usually transmits energy in the channel null space $\mathcal{N}_{\widetilde{H}}$. Consequently, the lost energy reduces the MAC $\mathcal{I}(\boldsymbol{x};\boldsymbol{y})$'s maximum value relative to its largest single-user GDFE value for the same $\widetilde{H}$ and a sum-energy (that is like a single-user) constraint. Good single-user GDFE design zeroes the $\mathcal{N}_{\widetilde{H}}$ components through Section 5.1's non-singular discrete-modulator input construction. This construction is usually not feasible with a MAC. The MAC's discrete modulator $A = \text{blkdiag}\left\{R_{\boldsymbol{xx}}^{1/2}(u)\right\}$ thus often has loss with respect to single-user canonical performance, which this section also characterizes. The corresponding reduced-GDFE $\mathcal{I}(\boldsymbol{x};\boldsymbol{y})$ is the MAC's rate sum. The MAC's highest rate sum uses Subsection 2.7's Simultaneous Water Filling (SWF), which necessarily has performance bounded by single-user water filling.

This section finds best energy allocations, and associated information distribution $\{b_u = \mathcal{I}_u\}_{u=1,\ldots,U}$, for all $\boldsymbol{b} \in \mathcal{C}_{MAC}(\boldsymbol{b})$. Subsection 5.4.1 and characterizes the MAC loss with respect to single user for the same $(H, R_{\boldsymbol{nn}})$ pair or channel. Subsection 5.4.2 details more completely Section 4.6's Vector DMT channel decomposition into a tone-indexed matrix-AWGN set, ultimately leading to tonal MAC GDFE's - again one for each[55] "tone." Subsection 5.4.3 progresses to weighted-sums, energy and rate, that help in design of specific "best" energy vectors $\boldsymbol{\mathcal{E}}_x$ (or autocorrelation matrices $R_{\boldsymbol{xx}}$) that correspond to a desired multiuser rate vector $\boldsymbol{b}$. There can be multiple "best" solutions for rate vectors in $\mathcal{S}(\boldsymbol{b})$; while boundary points in $\mathscr{C}(\boldsymbol{b})$ have distinct specific optima. Thus, th associated GDFE structure there helps complete the optimal MAC design. This design process can also assist Section 5.5's optimal BC design through a vector-channel expansion of Section 2.8's duality.

5.4.1 The GDFE's Relationship to the MMSE MAC

Chapter 2's MMSE MAC receiver appears very similar to the GDFE. However, there are subtle notational differences: Both the MMSE MAC's $\boldsymbol{\nu}$ and the MAC-GDFE's input $\boldsymbol{v}$ must have diagonal $R_{\boldsymbol{\nu\nu}} = I$ with any per-user-energy/autocorrelation-matrix scalings absorbed into the channel matrix such that

$$R_{f,MAC} = A^* \cdot \widetilde{H}_{GDFE}^* \cdot \widetilde{H}_{GDFE} \cdot A \quad . \tag{5.259}$$

The MAC's $R_{\boldsymbol{xx}}$ must be block diagonal with corresponding block-diagonal square roots[56] that shape a white input $R_{\boldsymbol{\nu\nu}} = I$ to the channel input $R_{\boldsymbol{xx}}$. Also the MMSE MAC's $\boldsymbol{\nu}$ and the GDFEs differ in that $\boldsymbol{\nu}$ best contains no channel-null-space components, so the single-user GDFE rate upper bounds the MAC sum rate for the same channel $(H, R_{\boldsymbol{nn}})$. Another difference is that the GDFE does not have per-dimensional energy constraints; leaving the energy-sum MAC closer conceptually to the GDFE. Sometimes the vector $\boldsymbol{v}$ appears in both, when a precoder finds use, then the precoder input is $\boldsymbol{\nu}$ and its output $\boldsymbol{v}$. This is important especially for Section 5.5's BC designs.

EXAMPLE 5.4.1 *[Comparison of a simple degraded MAC and GDFE]* A matrix AWGN channel has $\sigma^2 = .01$ and $H = [8 \quad 5]$. The MMSE MAC receiver follows Section 2.7's developments:

```
>> H=[80 60];
>> Au=sqrt(1/2)*eye(2);
>>Lxu = [1 1];
>> cb=2;
>> [Bu, GU, WU, SO, MSWMFU] = mu_mac(H, Au, Lxu , cb)
Bu =
    5.8222    0.3218
GU =
    1.0000    0.7500
```

[55] Complex dimension (or real dimension for the one baseband DC dimension at $n = 0$.

[56] There can also be square roots that are not block diagonal, but are not of interest for the MAC.

```
         0    1.0000
WU =
    0.0003         0
   -1.3333    1.7783
S0 =   1.0e+03 *
    3.2010         0
         0    0.0016
MSWMFU =
    0.0177
    0.0236
>> bsum=sum(Bu) =    6.1440
>> 10*log10(2^(sum(Bu))-1) =   18.4334 dB
```

The above MMSE MAC design resembles the GDFE, but without the null-space energy removal. The MMSE MAC calculations consequently result in a lower mutual information because some energy inevitably must be lost *in this* MAC's null space (if both users are nonzero). This MMSE MAC clearly has primary component as user 2, corresponding to channel-gain 80. If all energy (1 unit in this example) is on this primary component, the maximum rate sum is

```
>> 0.5*log2(1 + 6400) =  6.3220
```

The input autocorrelation that corresponds to the maximum rate sum is trivially SWF on the one used dimension

$$R_{\boldsymbol{x}\boldsymbol{x}}^{o} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.260}$$

The single-user GDFE instead first removes the energy in the null space, and has performance computed by VC or any other equivalent GDFE:

```
>> [F,L,M]=svd(H)
F = 1
L =  100     0
M =
    0.8000   -0.6000
    0.6000    0.8000
    >> 0.5*log2(1+ 0.5*L(1,1)^2) =  6.1440
```

A single-user-best GDFE design places 1 energy unit on the nonsingular mode, so then

$$\boldsymbol{u} = \begin{bmatrix} 0.8000 \\ 0.6000 \end{bmatrix} \cdot v \quad , \tag{5.261}$$

and $R_{\boldsymbol{u}\boldsymbol{u}} = 1$. However, (5.261) tacitly requires coordination of the two inputs because v must enter both channel inputs. This entry to both channel inputs is not possible with the MAC, but is with the single-user GDFE.

A single-user-best GDFE data rate is $\mathcal{I}(\boldsymbol{x};\boldsymbol{y})$ or

```
>> 0.5*log2(1+ L(1,1)^2) =    6.6439
```

which is higher than the energy-sum MAC's best data rate. There is thus a MAC loss relative to the single-user situation for the same $\widetilde{H}$ and $\mathcal{E}_x = 1$. That overall single user to MAC reduction is

$$b_{MAC} = 6.144 < 6.64 \ . \tag{5.262}$$

The energy loss is

$$\gamma_{MAC} = 10 \cdot \log_{10} \frac{2^{2 \cdot \bar{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y})} - 1}{2^{2 \cdot b_{esum-MAC}} - 1} \text{ dB}, \tag{5.263}$$

```
10*log10( (2^(6.6439)-1) / (2^(6.1440)-1)) = 1.5 dB
```

so illustrating that the energy-sum MAC's best rate sum can never exceed the maximum single-user $\mathcal{I}(\boldsymbol{x};\boldsymbol{y})$ possible on the same channel $\widetilde{H}$ with the same (energy-sum for MAC) input-energy constraint.

Example 5.4.1 then suggests the following MAC-loss measure caused by the MAC's need for no input coordination:

Definition 5.4.1 [MAC loss relative to single user] *The MAC loss for any given matrix AWGN channel* $[H, R_{\boldsymbol{nn}}]$ *relative to the single-user for that same channel is*

$$\gamma_{MAC} = \frac{2^{2\cdot\overline{\mathcal{C}}} - 1}{2^{2\cdot\overline{\mathcal{C}}_{esum-MAC}} - 1} \tag{5.264}$$

The numerator uses the single-user water-filling apacity $\mathcal{C}$ *for the same channel,* $[H, R_{\boldsymbol{nn}}]$.

While not formally defined, a $\gamma_{MAC}(R_{\boldsymbol{xx}})$ for an (possibly non water-fill) input is

$$\gamma_{MAC}(R_{\boldsymbol{xx}}) = \frac{2^{2\cdot\bar{\mathcal{I}}_{single-user}(R_{\boldsymbol{xx}})} - 1}{2^{2\cdot\bar{\mathcal{I}}_{MAC}} - 1} \quad . \tag{5.265}$$

Gaussian MACs with variable dimensionality inputs: Example 5.4.1 uses Chapter 2's mu_mac.m (see also Appendix G) program:

```
function [b, GU, WU, SO, MSWMFU] = mu_mac(H, A, Lxu , cb)
------------------------------------------------------------------------
Per-tonal (temporal dimension) multiuser mac receiver and per-user bits
Inputs: H, A , Uind , cb
Outputs: b, GU, WU, SO, MSWMFU
definitions
H: noise-whitened channel matrix [HU ... H1] Ly x sum-Lxu
A:    Block Diag sq-root sum-Lxu x sum-Lxu discrete modulators,
        blkdiag([AU ... A1]);  The Au entries derive from each MAC user's
        Lxu x Lxu input autocorrelation matrix, where the trace of each such
        autocorrelation matrix is user u's energy/symbol. This is per-tone.
Lxu:  # of dimensions for each user U ... 1 in 1 x U row vector
cb:   = 1 if complex baseband or 2 if real baseband channel
GU:   unbiased feedback matrix sum-Lxu x sum-Lxu
WU:   unbiased feedforward linear equalizer sum-Lxu x sum-Lxu
SO:   sub-channel channel gains sum-Lxu x sum-Lxu
MSWMFU: unbiased mean-squared whitened matched filter, sum-Lxu x Ly
b     user u's bits/symbol 1 x U
   the user should recompute b if there is a cyclic prefix
```

The mu_mac program allows variable input user dimensionality through input Lxu, as in Example 5.4.2:

EXAMPLE 5.4.2 *[Split-dimensionality MAC]* A 2-user channel has user 2 with 1 dimension, but user 2 with 2 dimensions. The matrix channel (noise whitened) and input autocorrelation functions appear below, along with the multi-user mac design.

```
>> H=[10    30    50
    20    40    60];
>> r22=1;
>> r11=[ 2 1
1 2];
>> A=blkdiag(r22,sqrtm(r11))=
    1.0000         0         0
         0    1.3660    0.3660
         0    0.3660    1.3660
>> Lxu = [1 2];
>> cb=1;
>> [Bu, GU, WU, SO, MSWMFU] = mu_mac(H, A, Lxu , cb)
Bu =    8.9687   10.4596
GU =
    1.0000    4.2497    5.4497
```

```
         0    1.0000    1.4672
         0         0    1.0000
WU =
    0.0020         0         0
   -0.0115    0.0027         0
    0.2745   -0.5237    0.3579
SO =
  501.0000         0         0
         0  371.1782         0
         0         0    3.7938
MSWMFU =
    0.0200    0.0400
    0.0456   -0.0222
    0.0751   -0.0513
```

5.4.1.1 MACs with intersymbol interference

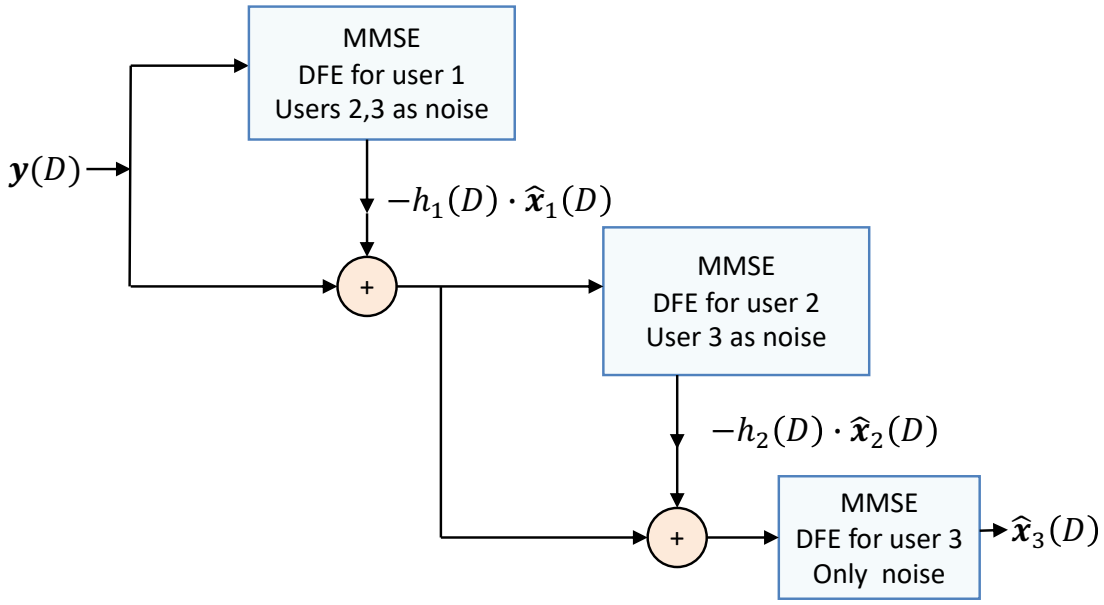


Figure 5.26: Nested GDFE.

The GDFE also provides a MAC design when there is ISI. The design recursively embeds the GDFE structures into Chapter 2's MAC with finite-length time-frequency symbols that each have intersymbol interference, as well as inter-user crosstalk and crosstalk-ISI. Figure 5.26 re-illustrates this **nested-GDFE-of-MMSE-DFE's** concept. With $\boldsymbol{x}(D)$ being the vector D-Transform (See Appendix D) of the U-multiple users' inputs, with U at the top, the chain rule expands to

$$\mathcal{I}\left(\boldsymbol{x}(D);\boldsymbol{y}(D)\right) = \sum_{u=1}^{U} \mathcal{I}\left(\boldsymbol{x}_u(D);\boldsymbol{y}(D)/\boldsymbol{x}_{u-1}(D)\right) \quad . \tag{5.266}$$

Figure 5.26's GDFE-MAC receiver uses Chapter 3's MMSE-DFE to decode first user 1 with all the other users as noise. Then this MAC-GDFE receiver removes user 1's crosstalk from the received signal $\boldsymbol{y}(D)$ and another MMSE-DFE decodes user 2. The MMSE-DFE design for user 1 proceeds as in Chapter 3, but adds all other users' received spectra to the noise spectrum. If the other users have $\Gamma = 0$ dB (i.e, close to Gaussian/AEP) codes, this approach reliably achieves a first-user data rate of $\mathcal{I}(x_1(D);\boldsymbol{y}(D))$. The receiver then processes the first MMSE-DFE's consequent ($\Gamma = 0$ dB) error-free decisions $\boldsymbol{x}_1(D)$ by a replica of the channel $\boldsymbol{h}_1(D)$ and subtracts it from $\boldsymbol{y}(D)$. This implies infinite delay, but captures

concept. This new signal enters a second MMSE-DFE for user 2 that treats users 3,...,U as noise. The design then recursively repeats to remove successively each user's effect upon remaining users. If there were no memory in the channel (thus no ISI within nor crosstalk among any of the users), each MMSE-DFE reduces to Chapter 2's simple scalar $1 \times U$ MAC.

Figure 5.26's MAC with $\overline{N} \to \infty$ has $SNR = 2^{2\overline{\mathcal{I}}(\boldsymbol{x}(D);\boldsymbol{y}(D))} - 1$ and is canonical, and the $\overline{\mathcal{I}}(\boldsymbol{x}(D);\boldsymbol{y}(D))$ can correspond to a chain-rule implementation for any order. Instead for $\overline{N} < \infty$, a GDFE can replace Figure 5.26's MMSE-DFE, which Example 5.4.3 illustrates:

EXAMPLE 5.4.3 [Two-user Nested GDFE's for ISI] Two synchronized users use a guard period of $\nu = 1$ sample with $N = 2$ output samples on the two independent paths of a real baseband MAC with $H_2(D) = 1 + .9 \cdot D$ and $H_1(D) = 1 - D$. The white noise has variance $\sigma^2 = .181$ and added as two dimensional noise to the two channel outputs' sum for each 2-sample packet at the common MAC receiver. These are no longer one-dimensional channels but are symbol-length approximations to the original MAC. There are 6 input dimensions, 3 for each user and 2 output dimensions. Nonetheless, the GDFE receiver applies for any set of input user-autocorrelation matrices. This example first examines the situation of 6 independent input dimensions, each of energy 1 per sample:

```
>> H =[H2 H1]
    1.0000    0.9000         0    1.0000   -1.0000         0
         0    1.0000    0.9000         0    1.0000   -1.0000
>> bsum=(.5/log(2))*log((det(H*eye(6)*H'+.181*eye(2))/det(.181*eye(2))))
    4.4622
```

The receiver GDFE decodes user 1 first on each of its 3 dimensions. The sum rate 4.4622 is not the highest possible, but corresponds to the choice of $R_{\boldsymbol{xx}} = I$ for the two users. While the users' inputs cannot coordinate, each user still unnecessarily wastes energy in the channel's respective null spaces. GDFE design continues as:

```
>> H=(1/sqrt(.181))*H;
Lxu = [3 3];
cb=2;
>> [Bu, GU, WU, S0, MSWMFU] = mu_mac(H, eye(6), Lxu , cb)
Bu =    3.2945    1.1677
GU =
    1.0000    0.9000         0    1.0000   -1.0000         0
         0    1.0000    0.8006    0.1227    0.7669   -0.8896
         0         0    1.0000   -0.5023    1.6134   -1.1111
         0         0         0    1.0000   -1.4520    0.4520
         0         0         0         0    1.0000   -0.5737
         0         0         0         0         0    1.0000
WU =
    0.1810         0         0         0         0         0
   -0.1227    0.1610         0         0         0         0
    0.5023   -0.6591    0.9558         0         0         0
   -1.0000   -0.4480    0.4068    1.5842         0         0
    0.4263   -0.1901   -0.5164    0.4263    0.7586         0
    0.0251    1.0226    0.9000    0.0251    0.9749    2.9885
S0 =
    6.5249         0         0         0         0         0
         0    7.2107         0         0         0         0
         0         0    2.0463         0         0         0
         0         0         0    1.6312         0         0
         0         0         0         0    2.3182         0
         0         0         0         0         0    1.3346
MSWMFU =
    0.4254         0
    0.0522    0.3785
   -0.2137    0.4727
    0.4254   -0.1923
   -0.1814    0.2441
   -0.0107   -0.4254
>> bvec=diag((0.5/log(2))*log(S0))
    1.3530
    1.4251
    0.5165
    0.3530
    0.6065
    0.2082
>> sum(bvec)  = 4.4622 (checks)
```

As expected, the mutual information is the sum of the data rates. User 1 (decoded first) is at a significant disadvantage in terms of data rate and SNR that is evident. The order of the two users can reverse to get a different decomposition of bit rates where user 2 is at a disadvantage, as per Chapter 2.

```
>> [Bu, GU, WU, S0, MSWMFU] = mu_mac([H(:,4:6) H(:,1:3)] , eye(6), Usize , cb)
Bu =    3.4207    1.0416
GU =
    1.0000   -1.0000         0    1.0000    0.9000         0
         0    1.0000   -0.8671   -0.1329    0.7475    0.7804
         0         0    1.0000   -0.4585   -1.4127   -0.9000
         0         0         0    1.0000    1.3585    0.4127
         0         0         0         0    1.0000    0.5443
         0         0         0         0         0    1.0000
WU =
    0.1810         0         0         0         0         0
    0.1329    0.1569         0         0         0         0
    0.4585    0.5415    0.7225         0         0         0
   -1.0000    0.5415    0.4585    1.7225         0         0
   -0.4391   -0.1657    0.6048   -0.4391    0.8800         0
   -0.0278   -1.0833    1.1111   -0.0278   -1.1362    3.9241
S0 =
    6.5249         0         0         0         0         0
         0    7.3716         0         0         0         0
         0         0    2.3841         0         0         0
         0         0         0    1.5806         0         0
         0         0         0         0    2.1364         0
         0         0         0         0         0    1.2548
MSWMFU =
    0.4254         0
   -0.0565    0.3689
   -0.1951   -0.4254
    0.4254    0.1951
    0.1868    0.2573
    0.0118    0.4727
>> bvec=diag((0.5/log(2))*log(S0))
bvec =
    1.3530
    1.4410
    0.6267
    0.3302
    0.5476
    0.1638
>> sum(bvec) =    4.4622
```

Perhaps of yet more interest would be the case where the design zeroes energy in the individual null spaces, $\mathcal{N}_1$ and $\mathcal{N}_2$. A better design instead reallocates the saved energy to the pass spaces $\mathcal{P}_1$ and $\mathcal{P}_2$ equally:

```
>> [F2, L2, M2]=svd(H(:,1:3));
>> M2ns=M2(1:3,1:2) =
   -0.4295    0.7412
   -0.8161   -0.0741
   -0.3866   -0.6671
>> Sx2=1.5*eye(2);
>> A2=M2(1:3,1:2)*sqrtm(Sx2);
```

This F1 is essentially the same as F2, a happy coincidence that might be exploited with simplifications that could reduce feedback with carefully designed input.

```
>> [F1, L1, M1]=svd(H(:,4:6));
>> M1ns=M1(1:3,1:2)
   -0.4082   -0.7071
    0.8165   -0.0000
   -0.4082    0.7071
>> Sx1=1.5*eye(2);
>> A1=M1(1:3,1:2)*sqrtm(Sx1);
>> [Bu, GU, WU, S0, MSWMFU] = mu_mac(H , blkdiag(A2 , A1), Usize , cb)
Bu =    3.8233    1.2019
GU =
    1.0000    0.0000    0.0000    0.6075
         0    1.0000   -1.8157    0.0000
         0         0    1.0000   -0.0000
         0         0         0    1.0000
WU =
```

```
    0.0445         0         0         0
   -0.0000    0.1326         0         0
   -0.0000    0.5508    0.3436         0
   -1.6462   -0.0000    0.0000    2.8307
S0 =
   23.4586         0         0         0
         0    8.5414         0         0
         0         0    3.9107         0
         0         0         0    1.3533
MSWMFU =
   -0.1492   -0.1492
    0.2575   -0.2575
   -0.1418    0.1418
   -0.2456   -0.2456
>> b=0.5*log2(det(S0)) =    5.0252
>> bvec=0.5*log2(diag(S0)) =
    2.2760
    1.5472
    0.9837
    0.2182
```

The sum rate is higher because the $R_{\boldsymbol{xx}}$ changed, and the revised inputs each avoid their individual channels' null spaces.

5.4.2 Vector DMT for the MAC

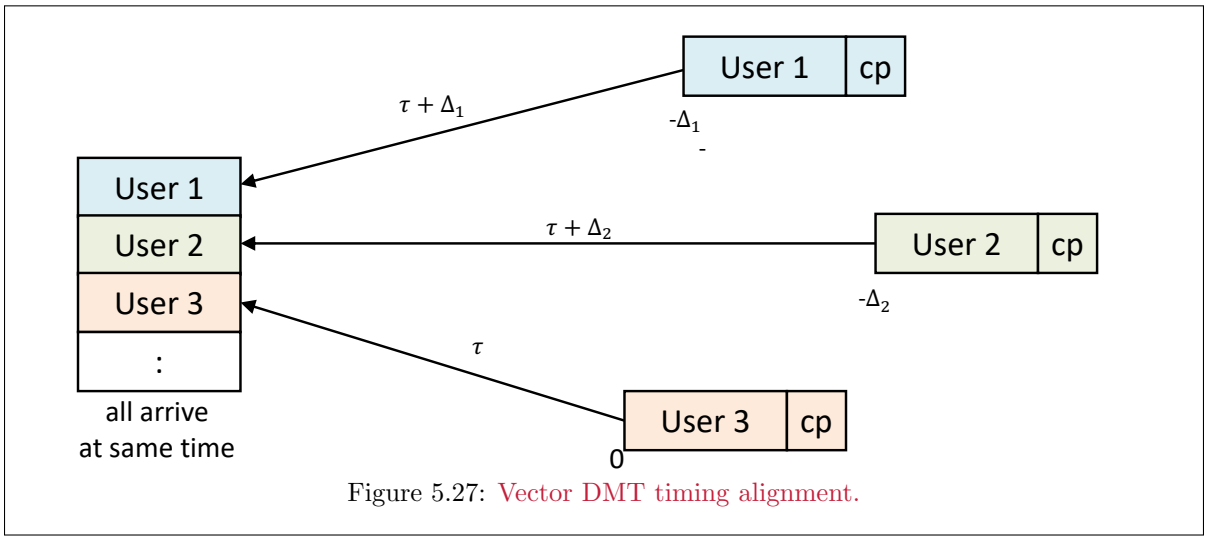


Figure 5.27: Vector DMT timing alignment.

Section 4.7 introduced Vector DMT for a linear time-invariant channel. Figure 5.27 illustrates DMT-symbol synchronization at a common receiver. Each transmitter uses the same length $(\bar{N} + \nu)$ DMT symbol and offsets transmit symbol boundaries so that all users arrive at the common receiver symbol boundary. Such alignment presumes the system supplies a common symbol clock to all MAC transmitters (essentially telling these transmitters to advance or delay their transmit symbol boundary until all align at the common MAC receiver). Section 4.6's digital duplexing or "zippering" allows such alignment.[57] Transmitter u has $L_{x,u}$ IFFT's, one at each of the U users' transmitters, and $\mathscr{L}_x = \sum_{u=1}^{U} L_{x,u}$. This $\mathscr{L}_x$ is common[58] to all tones n. There are L_y FFT's implemented at the common receiver. Such synchronization occurs with cyclic extension length that satisfies $\nu T' \geq \max_{u,u'} \{\text{length}\,(h_{u,u'}(t)) + \Delta_u\}$, which leads to no intersymbol interference. This alignment also ensures that any tone's crosstalk is a function ONLY of other users' signals on that same tone n.

[57] This digital duplexing also allows simultaneous synchronization in the opposite-direction BC, as in Section 5.5.

[58] Tones with zeroed dimensions and correspondingly zeroed (dummy) inputs may simplify concept and mathematical description, but of course do not improve $\mathcal{I}$ nor SNRs.

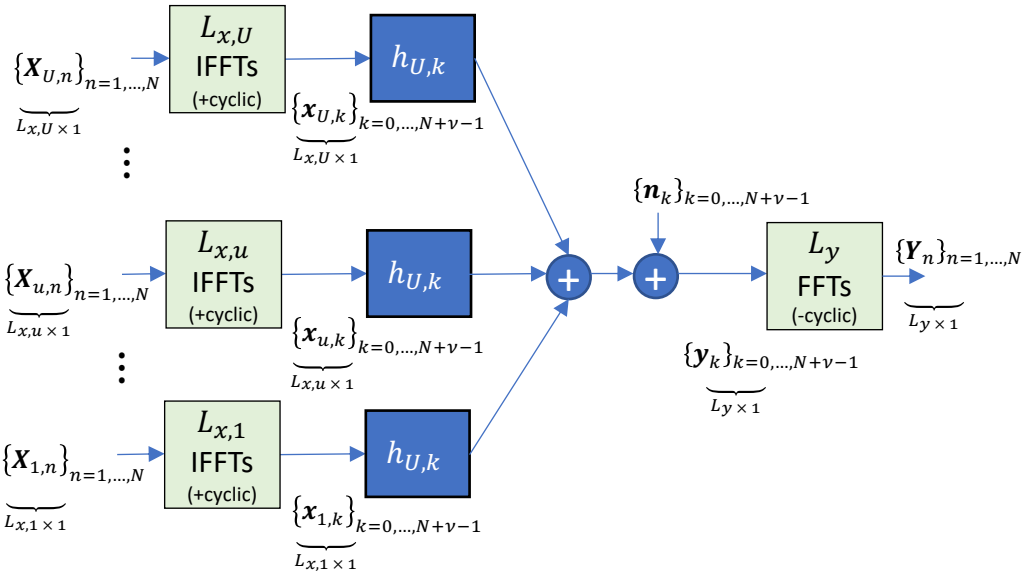


Figure 5.28: The Vector DMT system.

Tonal Channels: Figure 5.28's L_y receiver-FFT outputs jointly satisfy

$$\underbrace{\boldsymbol{Y}_n}_{L_y \times 1} = \underbrace{H_n}_{L_y \times \mathscr{L}_x} \cdot \underbrace{\boldsymbol{X}_n}_{\mathscr{L}_x \times 1} + \underbrace{\boldsymbol{N}_n}_{L_y \times 1} \quad , \; n = 0, ..., \bar{N} - 1 \tag{5.267}$$

where

$$H_n = [H_{U,n} \; ... \; H_{1,n}] \tag{5.268}$$

$$\boldsymbol{X}_n = \begin{bmatrix} \boldsymbol{X}_{U,n} \\ \vdots \\ \boldsymbol{X}_{1,n} \end{bmatrix} \tag{5.269}$$

$$\boldsymbol{Y}_n = \begin{bmatrix} y_{L_y,n} \\ \vdots \\ y_{1,n} \end{bmatrix} \tag{5.270}$$

$$\boldsymbol{X}_{u,n} = \begin{bmatrix} x_{u,L_{x,u},n} \\ \vdots \\ x_{u,1,n} \end{bmatrix} \quad . \tag{5.271}$$

Often $L_{x,u} = L_x$ in practice, although MAC "uplink" users need not have the same number of transmit antennas/dimensions. There is an individual GDFE for each tone in the Vector DMT/OFDM case.

Energy and autocorrelation-matrix constraints: The tonal autocorrelation matrix is

$$R_{\boldsymbol{X}\boldsymbol{X}}(u,n) = \mathbb{E}\left[\boldsymbol{X}_{u,n} \cdot \boldsymbol{X}_{u,n}^*\right] \quad . \tag{5.272}$$

This is essentially a 4-dimensional tensor[59] with square $L_{x,u} \times L_{x,u}$ matrix for each value of $u = 1, ..., U$ and $n = 0, ..., \bar{N} - 1$. Similarly, the $(l_y, l_x)^{th}$ entry of $H_{u,n}$ is DFT-tone-n's transfer gain/phase from

[59] A matlab object array dimensioned as $(L_{x,u} \times L_{x,u}, U, n)$ where size can be read for each $u = 1, ..., U$ and $n = 0, ..., \bar{N}-1$.

line/antenna l_x of user u to line/antenna l_y of the common output, where $\ell_x = 1,,,,,L_{x,u}$ and $\ell_y = 1,...,L_y$. The energy constraints become

$$\sum_{n=0}^{\overline{N}-1} \text{trace}\left\{R_{\boldsymbol{X}\boldsymbol{X}}(u,n)\right\} \leq \mathcal{E}_u \; \forall \; u = 1,...,U \quad , \tag{5.273}$$

which entries $\mathcal{E}_u$ comprise the energy vector $\boldsymbol{\mathcal{E}}$. For an energy-sum MAC, the individual user energies are variable and need only satisfy

$$\sum_{u=1}^{U} \sum_{n=0}^{\overline{N}-1} \text{trace}\left\{R_{\boldsymbol{X}\boldsymbol{X}}(u,n)\right\} \leq \mathcal{E}_x \quad . \tag{5.274}$$

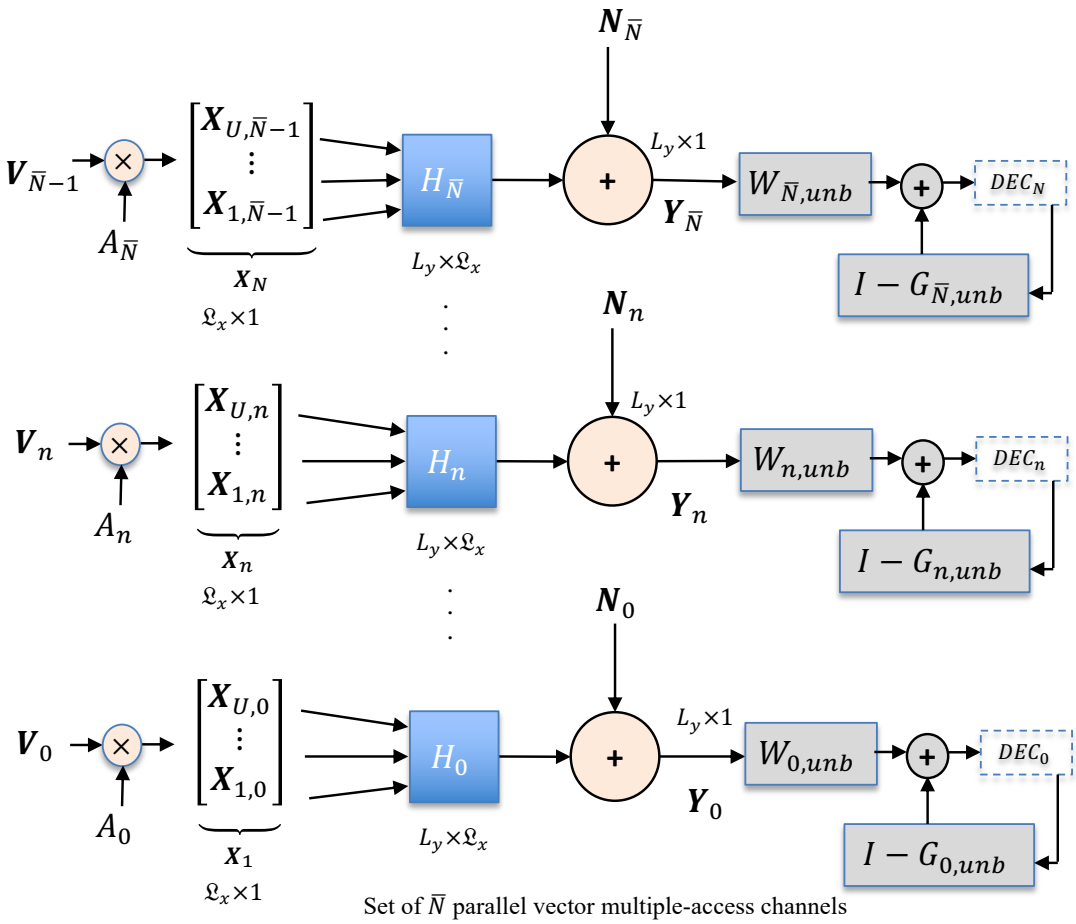


Figure 5.29: Tonal GDFEs with Vector DMT.

Tonal GDFE: Figure 5.29's tone-indexed model, with an individual GDFE for each tone, simplifies greatly full successive decoding (or GDFE) structure across all frequency and space dimensions. Essentially that structure now applies individually to each tone. Effectively, $\overline{N}$ GDFEs of size $L_y \times \mathscr{L}_x$ replace a giant GDFE of size $[L_y \cdot \overline{N}] \times [\mathscr{L}_x \cdot \overline{N}]$. The complexity reduction is enormous in application with large $\overline{N}$.

Encoders, Decoders, and Delay: Codes for any user can apply dimensionality over tones and of course time, and C-OFDM must do this, to reduce the delay. Each user (component) must however have its own code. The user bits/symbol has several useful forms:

$$b_{u,\ell} = \sum_{n=0}^{\overline{N}-1} b_{u,\ell,n} \tag{5.275}$$

$$b_{u,n} = \sum_{\ell=1}^{L_{x,u}} b_{u,\ell,n} \tag{5.276}$$

$$b_u = \sum_{\ell=1}^{L_{x,u}} \sum_{n=0}^{\overline{N}-1} b_{u,\ell,n} \tag{5.277}$$

$$b = \sum_{u=1}^{U} \sum_{\ell=1}^{L_{x,u}} \sum_{n=0}^{\overline{N}-1} b_{u,\ell,n} \quad . \tag{5.278}$$

5.4.2.1 Tonal GDFE Specification

Figure 5.29 shows the GDFE for any input-autocorrelation-matrix set $\{R_{\boldsymbol{xx}}(u,n)\}$. To implement the tonal GDFE with known $R_{\boldsymbol{XX}}(n) = A_n \cdot A_n^*$ and identity input $R_{\boldsymbol{VV}}(n) = I$, the receiver forms

$$\boldsymbol{Z}_n = \left[A_n^* \cdot H_n^* \cdot R_{\boldsymbol{NN}}^{-1}(n)\right] \cdot \boldsymbol{Y}_n \tag{5.279}$$

$$= A_n^* \cdot H_n^* \cdot R_{\boldsymbol{NN}}^{-1}(n) \cdot H_n \cdot A_n \cdot \boldsymbol{V}_n + \boldsymbol{N}'_n \tag{5.280}$$

$$= R_{f,n} \cdot \boldsymbol{V}_n + \boldsymbol{N}'_n \quad , \tag{5.281}$$

a forward canonical channel for tone n across all MAC users. The given GDFE order, which the ordering of inputs in $\boldsymbol{V}_n$ implies, tacitly assigns later users (higher-indexed dimensions) as uncanceled crosstalk noise. Different orders then simply correspond to re-indexing the GDFE dimensions. There are $\overline{N}$ such forward canonical channels that act independently[60], one for each tone $n = 0, ..., \overline{N} - 1$. The canonical backward channel has Cholesky factorization

$$R_{b,n}^{-1} = R_{f,n} + I = G_n \cdot S_{0,n} \cdot G_n^* \tag{5.282}$$

for each tone. The MAC order $\boldsymbol{\pi}$ is the same for all tones. The upper triangular Cholesky factor G_n determines that tone's feedback section. The diagonal matrix $\boldsymbol{SNR}$ contains the biased SNR's for the estimation of all users' dimensions and is

$$\boldsymbol{SNR}_n = S_{0,n} \quad . \tag{5.283}$$

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

$$W_{n,unb} = [\boldsymbol{SNR}_n] \cdot [\boldsymbol{SNR}_n - I]^{-1} \cdot S_{0,n}^{-1} \cdot G_n^{-*} \cdot A_n^* \cdot H_n^* \cdot R_{\boldsymbol{NN}}^{-1}(n) \quad . \tag{5.284}$$

The unbiased[61] feedback section is

$$G_{n,unb} = I + \boldsymbol{SNR}_n \cdot [\boldsymbol{SNR}_n + I]^{-1} \cdot [G_n - I] \quad . \tag{5.285}$$

When the channel rank is $\varrho_{\tilde{H}} = \mathscr{L}_x = U$, then there is no "other user" noise[62] and each (consequently primary) user occupies its own dimension(s). This corresponds to only primary-user components' dimensional use on tone n. When the rank is less, other (necessarily secondary) user components naturally become significant constituents of most dimensions' error signals in Figure 5.29. Decisions occur for each element of the vector $\boldsymbol{V}_n$ in succession from bottom to top (user U)according to the users' order. A GDFE structure exists for each and every order, which from a single-user GDFE perspective simply re-indexes dimensions and thus changes nothing fundamentally for multi-user, the components of $\boldsymbol{b}$ can vary with order, but $\mathbf{1}^* \cdot \boldsymbol{b} = b$ for all orders.

[60] Apart from any outer code's potential application across n to reduce implementation delay for a given $\Gamma > 0$ dB.
[61] A superscript of "unb" is used to denote "unbiased" to avoid confusion with the use of U as a user index.
[62] In a MMSE sense.

EXAMPLE 5.4.4 *[Revisit ISI Channel Example]* This revisits Example 5.4.3. The channel is real baseband. This ISI channel has a guard period so any SNR calculations should use an exponent of $1/N_x = 1/6$. A Vector DMT system could be used with equal energy on each of $\overline{N} = 8$ tones. Because such a system is not quite as good as vector-coded (or GDFE based on same input $R_{\boldsymbol{xx}}$ as vector-coded), the performance might be expect to approach Example 5.4.3 original design's data rate as $\overline{N}$ increases.

```
>> h0=[1 1];
>> h1=[.9 -1];
>> N=8;
>> H=(1/sqrt(.181))*fft(h, N, 3)
```

> **Comment:** Matlab's FFT command does not preserve energy, but rather increases the energy, or sum of squared sample magnitudes, by DFT size N (or $\overline{N}$ for complex baseband channels). So designers might then multiply the DFT output by $1/\sqrt{N}$ to restore DFT output energy. However noise whitening - even if just scaling a sampled wihite noise to unit variance – also corresponds also to a noise increase by N (or a reduction by $1/\sqrt{N}$ occurs in the denominator of the noise-equivalent channel, which means multiply by $\sqrt{N}$. With white-noise scaling, these two factors offset so the noise-whitened channel should not include the $1/\sqrt{N}$ energy normalization because the implied scaling in the noise exactly offsets this and the signal-to-noise ratio at each tone index n remains correct.

Thus, the preceding matlab command for H correctly uses no energy normalization by $\overline{N} = 8$. The designer also must understand the matlab program inputs. The program mu_mac program below has input A is for each user's subsymbol, thus $R_{\boldsymbol{xx}}(u) = 1$ $u = 1, 2$ for each symbol or $A_n = 1$, which corresponds to a two-dimensional identity matrix for each tone. Outside of the mu_mac use, the vector DMT system loses one energy unit to cyclic prefix, so then $A = \sqrt{8/9} \cdot I_2$. Also, the cb of mu_mac should be set to 2 for this real baseband example. The program executes for each value of N.

```
>> cb=2;
>> Lxu=[1 1];
>> nu=1;
A=zeros(2,2,N);
>> for n=1:N
A(:,:,n)=sqrt(8/9)*eye(2);
end
b=zeros(2,8);
GU=zeros(2,2,8);
WU=zeros(2,2,8);
SO=zeros(2,2,8);
MSWMFU=zeros(2,1,8);
>> for n=1:N
[ b(:,n), GU(:,:,n), WU(:,:,n), SO(:,:,n), MSWMFU(:,:,n)] = mu_mac(H(:,:,n), A(:,:,n), Lxu , cb);
end
>> b =
    2.1136    2.0063    1.6529    0.9316    0.0346    0.9316    1.6529    2.0063
         0    0.1183    0.4976    1.2438    2.1510    1.2438    0.4976    0.1183
>> (3/8)*(8/9)*sum(sum(b)) =    5.7334
```

The last calculation adjusts for the original vector-coded design being 3 dimensional, while this design increases to 8 dimensions, really 9 dimensions with cyclic prefix penalty, so the proper comparison accounts for the $\frac{3}{8} \cdot \frac{8}{9} = \frac{1}{3}$ adjustment factor. The data rate improves because the use of more frequency dimensions makes the 2/3 dimensional loss in the original system look large. This always happens with DMT designs - as $\overline{N}$ increases, the design performance improves; this remains true with tonal GDFE use. The feedback and MSWMFU are

```
>> GU
GU(:,:,1) =
   1.0000 + 0.0000i  -0.3991 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i
```

```
GU(:,:,2) =
   1.0000 + 0.0000i  -0.2928 + 0.0737i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,3) =
   1.0000 + 0.0000i   0.0931 + 0.1414i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,4) =
   1.0000 + 0.0000i   0.9056 + 0.1552i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,5) =
   1.0000 + 0.0000i   1.5833 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,6) =
   1.0000 + 0.0000i   0.9056 - 0.1552i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,7) =
   1.0000 + 0.0000i   0.0931 - 0.1414i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,8) =
   1.0000 + 0.0000i  -0.2928 - 0.0737i
   0.0000 + 0.0000i   1.0000 + 0.0000i
   >> MSWMFU
MSWMFU(:,:,1) =
   0.2375 + 0.0000i
   0.0000 + 0.0000i
MSWMFU(:,:,2) =
   0.2395 + 0.0932i
   0.2256 - 0.5447i
MSWMFU(:,:,3) =
   0.2493 + 0.2244i
   0.2256 - 0.2256i
MSWMFU(:,:,4) =
   0.3054 + 0.5346i
   0.2256 - 0.0935i
MSWMFU(:,:,5) =
   4.5125 + 0.0000i
   0.2256 + 0.0000i
MSWMFU(:,:,6) =
   0.3054 - 0.5346i
   0.2256 + 0.0935i
MSWMFU(:,:,7) =
   0.2493 - 0.2244i
   0.2256 + 0.2256i
MSWMFU(:,:,8) =
   0.2395 - 0.0932i
   0.2256 + 0.5447i
```

Example 5.4.4 has all user inputs as the same size (1 antenna each). Variable size inputs follow through the selection of the Lxu ($L_{x,u}$) parameter; the dimensionality is the same on all tones.

5.4.2.2 SWF for the Vectored DMT MAC

Simultaneous water-filling (SWF) for Vector DMT follows Section 2.7's maximum MAC rate sum. SWF selects any (and all) user u's energy through water-fill with all other users' crosstalk energy included as noise in autocorrelation matrix

$$\mathcal{R}_{noise}(u,n) = \sum_{i \neq u} H_{i,n} \cdot R_{\boldsymbol{XX}}(i,n) \cdot H_{i,n}^{*} + R_{\boldsymbol{NN}} \quad . \tag{5.286}$$

(5.286)'s equivalent-white-noise channel set thus becomes

$$\begin{aligned} \widetilde{H}_{u,n} &= \mathcal{R}_{noise}^{-1/2}(u,n) \cdot H_{u,n} \text{ with user-specific SVD} & (5.287) \\ &= F_{u,n} \cdot \Lambda_{u,n} \cdot M_{u,n}^{*} \; . & (5.288) \end{aligned}$$

The SWF energy distribution has each user satisfy

$$\mathcal{E}_{u,\ell,n} + \frac{1}{g_{u,\ell,n}} = K_u \; \forall \; u,\ell,n \text{ with } g_{u,\ell,n} = \lambda_{u,\ell,n}^{2} \; . \tag{5.289}$$

Each user has its own water level K_u. An $L_{x,u} \times L_{x,u}$ water-fill input autocorrelation matrix for user u follows from SVD's $M_{u,n}$ as

$$\left\{ R_{\boldsymbol{XX}}(u,n) \mid R_{\boldsymbol{XX}}(u,n) = M_{u,n} \cdot \text{Diag} \cdot (\boldsymbol{\mathcal{E}}_{u,n}) \cdot M_{u,n}^{*} \; \forall \; n = 0, ..., \overline{N} - 1 \; u = 1, ..., U \right\} \quad . \tag{5.290}$$

Designs compute such a set $\{R_{\boldsymbol{XX}}(u,n)\}$ through Section 2.7's iterative water-filling that determines successive user energies for MAC channel $H_{u,n}$ with noise autocorrelation matrix $\mathcal{R}_{noise}(u,n)$ in a convergent algorithm. An energy-sum MAC's maximum sum rate's achievement energizes only primary-user components; these primary components correspond to the best U^o dimensions available for use, as in Section 2.8. The all-primary energization applies now to each tone, n; the dimensions corresponding to primary components may vary with n. This exemplifies the reason for the term "user components" - different tones may have different ranks and splits between as either primary or secondary, and only users' primary components energize when the rate-sum is maximum, but the same order. Each user water-fills energy with respect to all others as noise over dimensional indices n and ℓ to water level K_u. Other water-fill-approximating energy loading[63] follows similarly and treats other users as noise in determining energy per dimension. $\mathcal{E}_u = \sum_n \mathcal{E}_{u,n}$ defines the diagonal energy matrix entries $\text{Diag}(\boldsymbol{\mathcal{E}}_x)$. When $L_{x,u} > 1$, then the block-diagonal matrix $R_{\boldsymbol{xx}}(u,n)$ generalizes to $trace\{\sum_n R_{\boldsymbol{xx}}(u,n)\} \leq \mathcal{E}_u$ for user u.

Each user's bits/symbol b_u can vary with order. But the overall rate sum is constant at

$$b = \sum_{u=1}^{U} b_u = \sum_{u=1}^{U} \sum_{n=0}^{\overline{N}-1} b_{u,n} \quad , \tag{5.291}$$

for all $U!$ orders.

Lemma 5.4.1 **[Existence of the orthogonal-division mutiplexed (ODM) point]** *When $L_x = 1$, there exists a rate-sum vector $\boldsymbol{b}_{max}$ for which all orders provide the same $\{b_u\}_{u \in \boldsymbol{u}^o}$ set - an "FDM point," as $\overline{N} \to \infty$. When $L_{x,u} > 1$, then such an ODM point also exists when $U^o = U' =\leq \varrho_H$ where $\varrho_{\widetilde{H}} = \sum_{n=0}^{\overline{N}-1} \varrho_{\widetilde{H}_n}$. When $\overline{N} < \infty$ and $U^o = U'$, there is an SWF orthogonal-dimension-multiplexing point where each primary user occupies solely its own spatial dimensions with no crosstalk from other users.*

Proof: The SWF optimization decomposes into U^o separate problems each has lowest possible (zero crosstalk) noise when they occupy a single dimension and trivially is then also SWF, as long as $U^o = U' \leq \varrho_{\widetilde{H}}$. This latter condition certainly holds if $\overline{N} \to \infty$ (at constant sampling rate). **QED.**

The Matlab SWF.m program Chapter 2 first introduced SWF.m, which this section revisits now with $\overline{N} \geq 1$.

```
function [Rxx, bsum , bsum_lin] = SWF(Eu, H, Lxu, Rnn, cb)
  Simultaneous water-filling MAC max rate sum (linear and nonlinear GDFE)
  The input is space-time domain h, and the user can specify a temporal
  block symbol size N (essentially an FFT size).
  Inputs:
  Eu  U x 1 energy/SAMPLE vector. Single scalar equal energy all users
      any (N/N+nu) scaling should occur BEFORE input to this program.
  H   The FREQUENCY-DOMAIN Ly x sum(Lx(u)) x N MIMO channel for all users.
      N is determined from size(H) where N = # used tones
  Lxu 1xU vector of each user's number of antennas
  Rnn The Ly x Ly x N noise-autocorrelation tensor (last index is per tone)
  cb  cb = 1 for complex, cb=2 for real baseband
        cb=2 corresponds to a frequency range at an sampling rate 1/T' of
        [0, 1/2T'] while with cb=1, it is [0, 1/T'].  The Rnn entered for
        these two situations may differ, depending on how H is computed.
  Outputs:
  Rxx A block-diagonal psd matrix with the input autocorrelation for each
      user on each tone. Rxx has size (sum(Lx(u)) x sum(Lx(u)) x N .
      sum trace(Rxx) over tones and spatial dimensions equal the Eu
  bsum the maximum rate sum.
  bsum  bsum_lin - the maximum sum rate with a linear receiver
      b is an internal convergence sum rate value, not output
  This program significantly modifies one originally supplied by student
  Chris Baca
```

EXAMPLE 5.4.5 (SWF Program Use) An example channel has $U = 2$ users, $L_y = L_x = 2$ with $2 \times 4 \times 2$ channel tensor H:

[63] such as Chapter 4's Levin Campello

```
>> H = [80 30; 40 -50; 30 -15; 20 25]';
>> H(:,:,2) = [20 0; -20 10; 35, 45; 10, 0]';
>> Rnn=zeros(2,2,2);
>> Rnn(:,:,1)=eye(2);
>> Rnn(:,:,2)=eye(2);
>> Eu = [1, 2];
>> cb=2;
>> [Rxx, bsum, bsum_lin] = SWF(Eu, H, [2 2], Rnn, cb)
Rxx(:,:,1) =
    1.4141    0.1921         0         0
    0.1921    1.0847         0         0
         0         0    0.6966   -0.9584
         0         0   -0.9584    1.3187
Rxx(:,:,2) =
    0.5097   -0.7109         0         0
   -0.7109    0.9915         0         0
         0         0    5.9681    0.3152
         0         0    0.3152    0.0166
bsum =   25.1071
bsum_lin =   19.9260
>> sum(diag(Rxx(1:2,1:2,1))+diag(Rxx(1:2,1:2,2))) =     4  (checks)
>> sum(diag(Rxx(3:4,3:4,1))+diag(Rxx(3:4,3:4,2))) =     8  (checks)
```

The maximum sum rate is roughly 25 bits/symbol for both users, while a linear (GLE) would instead have roughly 20 bits/symbol for those same two users. The input Eu is energy/sample, so user 1 has 2 spatial dimensions with two frequency dimensions each and thus total energy per symbol 4. Similarly user 2 has the same 4 dimensions but with 2 units of energy each, so total 8.

Energy-Sum MAC maximum rate sum: Simultaneous water-filling corresponds to the largest data rate for any given $trace\{R_{\boldsymbol{xx}}\}$; the energy-sum MAC's largest rate sum will correspond to a particular set (or possibly sets) of $R_{\boldsymbol{xx}}$. A search through all possible energy sums produces this energy-sum maximum. The energy-sum MAC however is of most interest in duality with a corresponding BC channel (which always has the input energy-sum constraint); Section 5.5 introduces duality, and eventually introduces a bcmax.m matlab program that implements Chapter 2's iterative double loop of worst-case-noise and water-filling to compute the saddle-point maximum BC rate sum. That software effectively short cuts the dual MAC's energy-sum maximum rate computation to a single calculation.

Nonethless, Chapter 2's macmax.m program also computes this maximum rate sum with $\overline{N} \geq 1$:

```
function [Rxx, bsum , bsum_lin] = macmax(Eu, h, Lxu, N , cb)
Simultaneous water-filling Esum MAC max rate sum (linear & nonlinear GDFE)
The input is space-time domain h, and the user can specify a temporal
block symbol size N (essentially an FFT size).
This program uses the CVX package
the inputs are:
Eu  The sum-user energy/SAMPLE scalar.
    This will be increased by the number of tones N by this program.
    Each user energy should be scaled by N/(N+nu)if there is cyclic prefix
    This energy is the trace of the corresponding user Rxx (u)
    The sum energy is compouted as the sum of the Eu components
    internally.
h   The TIME-DOMAIN Ly x sum(Lx(u)) x N channel for all users
Lxu  The number of antennas for each user 1 x U
N   The number of used tones (equally spaced over (0,1/T) at N/T.
cb  cb = 1 for complex, cb=2 for real baseband
the outputs are:
Rxx A block-diagonal psd matrix with the input autocorrelation for each
    user on each tone. Rxx has size (sum(Lx(u)) x sum(Lx(u)) x N .
    sum trace(Rxx) over tones and spatial dimensions equal the Eu
bsum the maximum rate sum.
bsum  bsum_lin - the maximum sum rate with a linear receiver
    b is an internal convergence (vector, rms) value, but not sum rate
```

The above example has 3 units of total energy, and the macmax.m program needs the time-domain input, so:

```
>> h=ifft(H,2,3)
h(:,:,1) =
   50.0000   10.0000   32.5000   15.0000
   15.0000  -20.0000   15.0000   12.5000
h(:,:,2) =
   30.0000   30.0000   -2.5000    5.0000
```

```
   15.0000  -30.0000  -30.0000   12.5000
>> >> [Rxx, bsum , bsum_lin] = macmax(3, h, [2 2], 2 , cb)
Rxx(:,:,1) =
    3.0005    0.0000         0         0
    0.0000    3.0005         0         0
         0         0    0.0000   -0.0000
         0         0   -0.0000    0.0000
Rxx(:,:,2) =
    1.0182   -1.4201         0         0
   -1.4201    1.9806         0         0
         0         0    2.9918    0.1577
         0         0    0.1577    0.0083
bsum =   26.1057
bsum_lin =   25.9960
trace(sum(Rxx,3)) =    12.0000  (checks)
```

these data rates are indeed higher than the energy-vector MAC as necessarily must be the case (or equal), because the energy-sum constraint subsumes the energy-vector constraint.

Column-Wise Diagonal Dominance: A GDFE special case occurs when $R_{\boldsymbol{nn}}(n) = I$ and $H_n \cdot A_n$ is column dominant. Column dominance means that each column's diagonal element is much greater than the other elements in that column, so if $\widetilde{H}_n = H_n \cdot A_n$, then

$$|\widetilde{H}_n(u,u)| >> |\widetilde{H}_n(i \neq u, u)| \; \forall \; i \neq u \quad . \tag{5.292}$$

This happens for instance with Chapter 2's massive MIMO. With diagonal dominance, R_f and thus R_b^{-1} approximate diagonal matrices, and then $G \to I$. There is no feedback in this diagonally dominant situation, and order is irrelevent in the GDFE – all orders produce the same result. Indeed each user best uses an individual water-filling energy distribution with respect to only its noise, $\mathcal{R}_{noise}(u,n) = R_{\boldsymbol{nn}}(n) \; \forall \; u$. This situation also occurs in most vectored DSL applications, and in massive-MIMO wireless[64], channels when the non-user upstream/uplink spatial noise is "white." With column-wise dominance, the GDFE feedforward matrix is solely sufficient to eliminate crosstalk on every tone.

The tonal GDFE bits/tone n for user u's l^{th} input line/antenna in any MIMO system with a GDFE is

$$b_{u,l,n} = \log_2 \left(\mathcal{E}_{u,\ell,n} \cdot S_{0,u,\ell,n} \right) \quad . \tag{5.293}$$

An example considers $U = 3$ and all 3 users have crosstalk and temporal intersymbol interference.

EXAMPLE 5.4.6 [3-user Vector Gaussian MAC with 2-dimensional output] A (complex baseband) vector ISI MAC has 3 user inputs and a two-dimensional output, essentially guaranteeing secondary user-component existence with its non-trivial ISI. The noise is white with variance $\mathcal{N}_0 = .01$. The channel is

$$H(D) = \begin{bmatrix} 1 + .9 \cdot D & -.3 \cdot D + .2 \cdot D^2 & .8 \\ .5 \cdot D - .4 \cdot D^2 & 1 - D - .63 \cdot D^2 + .648 \cdot D^3 & 1 - D \end{bmatrix} \quad . \tag{5.294}$$

The D-transform roughly allows the designer to see user 3 as roughly lowpass, user 2 as somewhat bandpass, and user 1 as highpass. This example first computes data rates for equal energies, starting with the matlab commands

```
h=cat(3,[1 0 .8 ; 0 1 1],[.9 -.3 0 ; .5 -1 -1],[0 .2 0 ; .4 -.63 0],[0 0 0 ; 0 .648 0])*10;
h(:,:,1) =
    10     0     8
     0    10    10
h(:,:,2) =
     9    -3     0
     5   -10   -10
h(:,:,3) =
         0    2.0000         0
    4.0000   -6.3000         0
h(:,:,4) =
         0         0         0
         0    6.4800         0
```

[64] Vectored DSLs [4] significantly predate "massive MIMO," but indeed are exactly the same effect from fundamental communication viewpoint.

before continuing to minPMAC use. The DFT for vector DMT with $\overline{N} = 8$ is then (recalling the earlier comment on noise whitened channels), this time now for a complex baseband channel, but again no scaling of noise-whitened channel from matlab FFT command:

```
>> N=8;
>> H = fft(h, N, 3)
H(:,:,1) =
  19.0000 + 0.0000i  -1.0000 + 0.0000i   8.0000 + 0.0000i
   9.0000 + 0.0000i   0.1800 + 0.0000i   0.0000 + 0.0000i
H(:,:,2) =
  16.3640 - 6.3640i  -2.1213 + 0.1213i   8.0000 + 0.0000i
   3.5355 - 7.5355i  -1.6531 + 8.7890i   2.9289 + 7.0711i
H(:,:,3) =
  10.0000 - 9.0000i  -2.0000 + 3.0000i   8.0000 + 0.0000i
  -4.0000 - 5.0000i  16.3000 +16.4800i  10.0000 +10.0000i
H(:,:,4) =
   3.6360 - 6.3640i   2.1213 + 4.1213i   8.0000 + 0.0000i
  -3.5355 + 0.4645i  21.6531 - 3.8110i  17.0711 + 7.0711i
H(:,:,5) =
   1.0000 + 0.0000i   5.0000 + 0.0000i   8.0000 + 0.0000i
  -1.0000 + 0.0000i   7.2200 + 0.0000i  20.0000 + 0.0000i
H(:,:,6) =
   3.6360 + 6.3640i   2.1213 - 4.1213i   8.0000 + 0.0000i
  -3.5355 - 0.4645i  21.6531 + 3.8110i  17.0711 - 7.0711i
H(:,:,7) =
  10.0000 + 9.0000i  -2.0000 - 3.0000i   8.0000 + 0.0000i
  -4.0000 + 5.0000i  16.3000 -16.4800i  10.0000 -10.0000i
H(:,:,8) =
  16.3640 + 6.3640i  -2.1213 - 0.1213i   8.0000 + 0.0000i
   3.5355 + 7.5355i  -1.6531 - 8.7890i   2.9289 - 7.0711i
```

Use of equal energy on all tones and users, assuming 1 energy unit per tone or ($\bar{\mathcal{E}}_{\boldsymbol{x}} = 0.5$ for this complex baseband channel) generates Figure 5.30 of data rate versus $\overline{N}$, with the matlab commands to follow. Section Scaling on FFT's with noise whitening. The input energy to the mu_mac program also takes energy per tone for each user, which for $\nu = 3$ offsets to $\overline{N}/(\overline{N} + \nu)$:

```
Nmax=32;
Nmax=32;
U=3;
Ly=2;
cb=1;
Usize=[1 1 1];
bsum=zeros(1,Nmax);
for index=1:Nmax
  i=2*index;    % (don't need to plot a point for every number of tones)
  H = fft(h, i, 3);
  GU=zeros(U,U,i);
  WU=zeros(U,U,i);
  S0=zeros(U,U,i);
  Bu=zeros(U,i);
  MSWMFU=zeros(U,Ly,i);
  AU=zeros(3,3,i);
  for n=1:i
    AU(:,:,n)=sqrt(i)/sqrt(i+3)*eye(3);
  end
  for n=1:i
   [Bu(:,n), GU(:,:,n) , WU(:,:,n),S0(:,:,n), MSWMFU(:,:,n)] = ...
    mu_mac(H(:,:,n), AU(:,:,n), Usize, cb);
  end
bvec=sum(Bu');
bsum(index)=sum(bvec);
end
bvec =  445.1264  412.8794  132.7477
sum(bvec) =  990.7535
N=64;
for n=1:N
Rnn(:,:,n) = eye(2);
end
[Rxx, bsum, bsumlin] = SWF(N/(N+3)*[1 1 1], H, [1  1 1], Rnn, 1);
>> bsum =   1011.1 > 990.8
>> bsumlin =  578.8502- linear loss almost 50 \%
```

The sum data rate for equal users' energy per temporal dimension approaches about 15.2 bits/tone within each symbol (longer symbols of course have higher total number of bits/symbol).

There are also two spatial dimensions, so this is about 7 bits/spatial dimension for each dimension within a symbol. For $\overline{N} \geq 64$, there is very little additional rate-sum gain. Since maximum rate sums often take a smaller $\overline{N}^*$ than full flat-bandwidth input spectra, this example will proceed with $\overline{N} = 64$. Some illustrative quantities are feedback sections and overall feedforward processing for the receiver, so by selecting some tones:

```
>> GU(:,:,23)  =
   1.0000 + 0.0000i  -1.3365 + 0.3447i  -0.3093 + 0.3130i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.8108 + 0.4218i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
>> WU(:,:,37) =
   0.0558 - 0.0000i   0.0000 - 0.0000i   0.0000 + 0.0000i
   0.0102 - 0.0066i   0.0067 - 0.0000i   0.0000 + 0.0000i
  -0.2761 - 0.0233i  -0.1830 - 0.1201i   0.1328 + 0.0000i
  >> MSWMFU(:,:,15) =
   0.0454 + 0.0341i  -0.0105 + 0.0249i
   0.0183 + 0.0179i   0.0275 - 0.0468i
   0.0589 + 0.0199i   0.0179 - 0.0416i
>> S0(:,:,29) =
   18.9071         0         0
         0  149.3696         0
         0         0    8.5326
>> 10*log10(diag(S0(:,:,29))) =
   12.7663  dB
   21.7426  dB
    9.3108  dB
```

The nonzero feedback sections illustrate best-performance's need for the nonlinear GDFE/MAC operations to obtain highest sum data rate, albeit these feedback sections were for the design with equal energy (not the SWF Rxx distribution). Inspection of a few randomly chosen tones' $R_{\boldsymbol{xx}}(n)$ values also shows that all 3 users are energized because this is for vector MAC, even though there are only two output dimensions/tone. This corresponds necessarily to secondary user components.

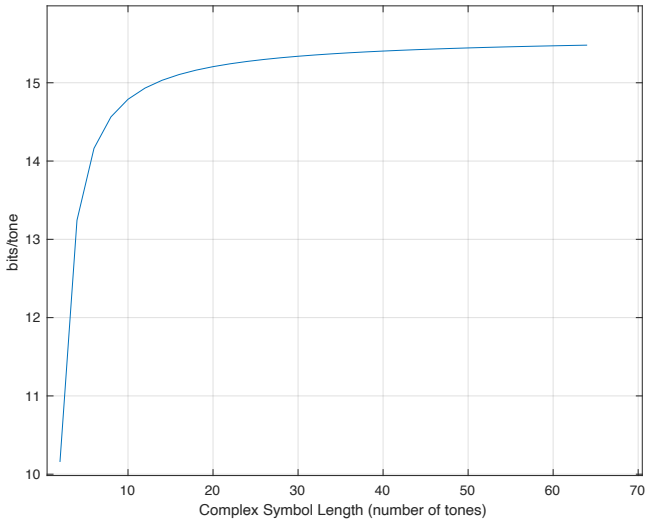


Figure 5.30: Sum data rate in bits/complex subsymbol for Example 5.4.6.

(Example 5.4.6 continued) The maximum sum rate for the energy-sum MAC (so 3 (64/67) total energy for BC follows from the macmax program.

```
[Rxx, bsum , bsum_lin] = macmax(3*(64/67), h, Usize, N , 1);
bsum = 1011.3
bsum_lin = 571.7289
```

This data rate is also higher and is a maximum when the total symbol energy for all 3 MAC users is limited to $3 \cdot 64 \cdot (64/67)$. Also, the linear-only receiver loses roughly half the data rate. Thus, for this highpass-bandpass-lowpass user-channel combination, the linear system only is very limited.

```
>> Rxx(:,:,23) =
    0.7615         0         0
         0    1.1885         0
         0         0    0.9480
>> Rxx(:,:,47) =
     1.1961         0         0
         0    1.2796         0
         0         0    0.4552
```

These particular tone-indexed spatial autocorrelation matrices have curiously 3 excited modes (not all do, but those in the transition bands where the channel well passes all 3 users are similarly excited). If $\overline{N} = 1$, this does not happen for maximum sum rate (with energy-sum-only constraint), meaning maximum of two primary user components. However, here there are 3 with $\overline{N} > 1$. Thus, this is not an FDM solution; optimum FDM solution is only guaranteed when $L_y = 1$, but of course here $L_y = 2$ so no guarantee. This example has equal-energy solution and optimum are very close (the sum rates are the same to negligible round off). With $L_y > 1$, there is a mix of vertex sharing across time and frequency dimension that can lead to solutions that necessarily need $G \neq I$, and this example illustrates that even with many tones; the GDFE feedback is nontrivial for the MIMO-GDFE multiuser system. This means some users must partition into components, and for this channel those user components that contribute to the SWF are all primary and those found already. This channel's later revisit to design for a specific $\boldsymbol{b}' \in \mathcal{C}(\boldsymbol{b})$ shows that secondary users excite and vertex-sharing becomes necessary for best design.

The dual channel (see Section 5.5 to understand the remainder of this example) maximum data rate should be the same and follows from

```
>> J2=hankel([0 1]);
J3=hankel([0 0 1]);
rxxb=zeros(2,2,64);
Hbc=zeros(3,2,64);
bbc=zeros(3,64);
for n=1:64
Hbc(:,:,n)=J3*H(:,:,n)'*J2;
rxxb(:,:,n)=(1.5/67)*eye(2);
end
[oRxx, Rwcn, bmax] = bcmax(rxxb, Hbc, 1);
bmax =  505.6484
2*bmax=1101.3
2*bmax/67 =   15.0940
```

so as expected the the max rate sum of the Esum MAC matches its dual maximum rate sum. Figure 5.31 plots the data rate versus FFT size for both the optimum with Feedback and linear without feedback. Because the number of users exceeds the number of spatial dimensions, the loss is significant, almost 50%.

Spatial Correlation of Noise Spatial correlation refers to the matrix $R_{\boldsymbol{NN}}(n)$'s possibly non-diagonal structure on any (or many/all) of a vector-DMT/OFDM system's independent tone-indexed matrix channels. Since such noise leads effectively to a crosstalking noise-whitened channel, the GDFE also canonically handles such spatially correlated MAC noise. The consequent noise-whitened channel is not necessarily diagonally dominant, even if the original channel is diagonally dominant.

Earliest use of GDFE:

EXAMPLE 5.4.7 (Vectored VDSL) VDSL is a zippered-DMT system as described in Section 4.6. The upstream direction is a vector-MAC if the DSLAM uses receiver coordination. The tone spacing is 4.3125 kHz with a cyclic extension of 640 samples on a

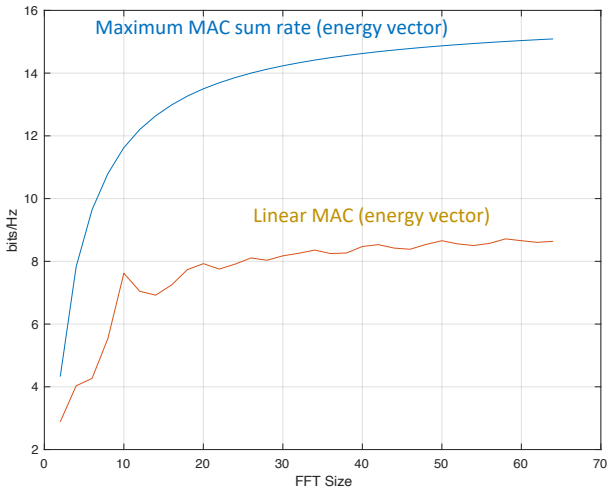


Figure 5.31: maximum sum rates 3-user example.

$1/T' = 32 \times 2.208$ MHz sampling clock. Up to 8192 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 crosstalk noise noise spectra A - (less severe) and F (more severe) are used for testing. These channels are all diagonally dominant.

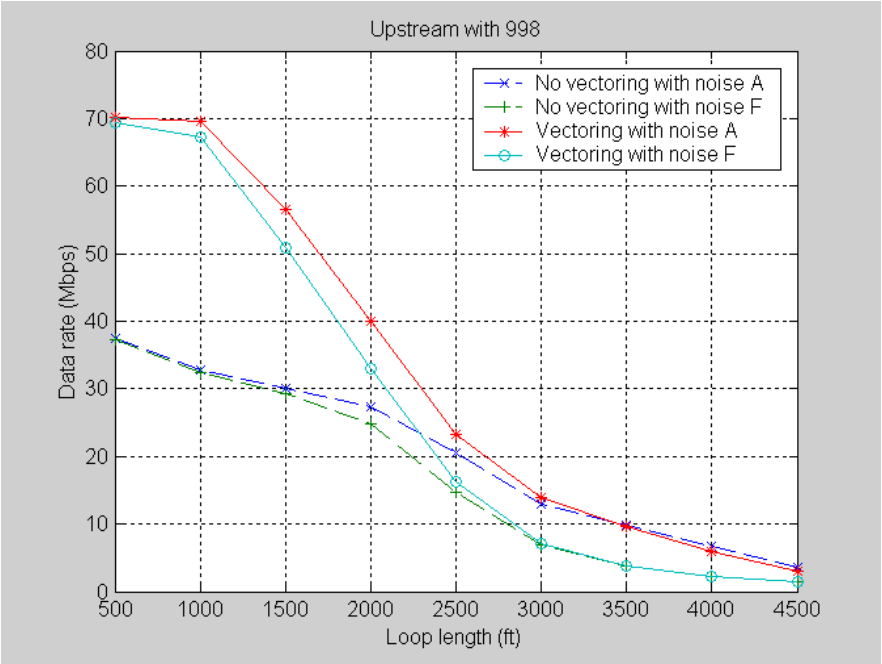


Figure 5.32: VDSL with and without vectoring - no spatial correlation of other noises.

Figure 5.32 investigates use of a vectored GDFE. In the lower curves, the upstream data rate uses water-filling in each of the FDM bands with no MAC processing nor feedback section. The upper curve is the data rate achieved with the GDFE 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 5.32. Figure 5.32's systems have $U = 25$ users and[65] $\varrho_H = 25$. However, the crosstalk is largest between immediately neighboring connections, perhaps obviously so conceptually[66]. The wires are also twisted-pairs with twisting at different number of twists per unit length on adjacent lines. Thus the crosstalk model is a random-selection process (across users, not time). For the H matrix on the tones, this means that the not only are the crosstalk contributions random, but that the diagonal spatial-matched-matrix terms are much larger in a 25×25 channel matrix than are adjacent terms. This causes diagonal dominance effect, which allows vectored VDSL to avoid the feedback section, and Figure 5.32 uses this.

Figure 5.33 illustrates the situation if the other crosstalkers are fewer in number than the number of vectored users in the upstream direction, and thus their contributions appear less random and thus spatially correlated. In this case the diagonal may still dominate because $U = 25$ is greater than the number of correlated crosstalkers significantly. However the noise correlation allows the GDFE's forward-matrix process to reduce further the crosstalk effect, again here without the feedback section. Figure 5.33's data rate averages the channel samples over users (all have same length). Again, yet an even higher data rate is achieved at short lengths. In this case the other crosstalkers overlap the upstream VDSL band. Figure 5.33 uses no frequency plan and the users simply adapt to best band use with 14.5 dBm of upstream transmit power. Figure 5.33 yet further shows that data-rate increase is large with exploitation of noise spatial correlation (the rho value in plots).

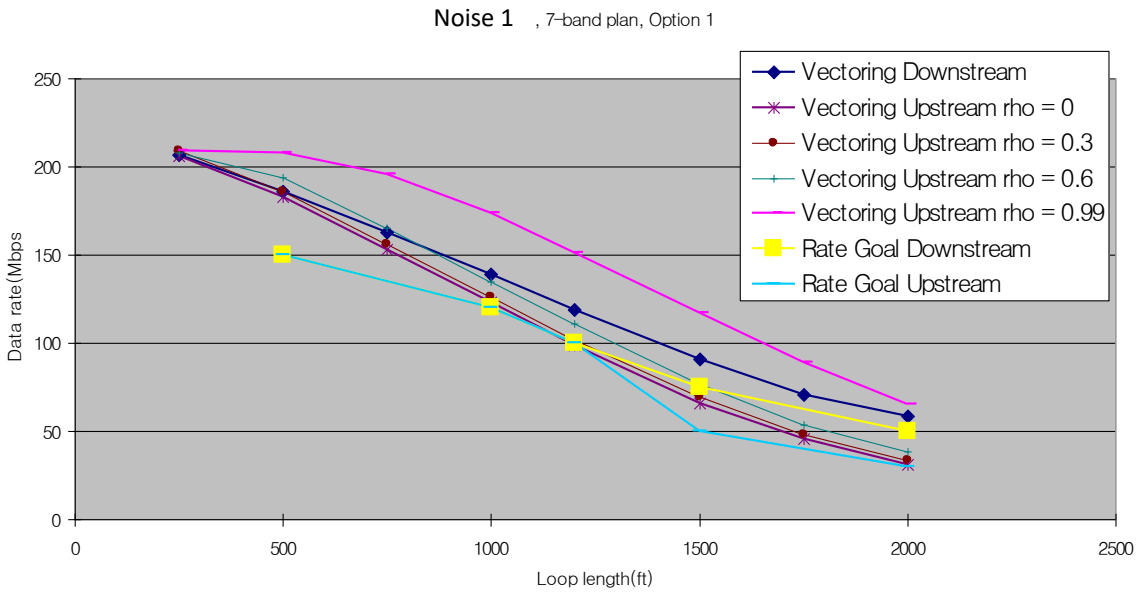


Figure 5.33: VDSL with vectoring - other noises' spatial correlation have fewer sources than the number of users.

[65] Systems with up to 384 users are in field use circa 2020.

[66] Not all wires can be physically next to one another in a cable full of such wires - so the nearest neighbor lines create most of crosstalk.

5.4.3 MAC Weighted Sums

Weighted rate and/or energy sums are an approach to MAC-input optimization. Maximum weighted sum-rate for given energy constraint $\boldsymbol{\mathcal{E}_x}$ is a potential design criterion, but does not fully control the individual user data rates (even though they are weighted presumably according to their importance). Alternately, minimum weighted energy-sum for a given desired data rate $\boldsymbol{b}$ conserves energy, but may not meet all user energy constraints. Nonetheless, a simultaneous, weighted-energy-sum minimizing and weighted-rate-sum maximizing, solution allocates resources efficiently while meeting the target $[\boldsymbol{b}, \boldsymbol{\mathcal{E}_x}]$. This subsection begins first with these two sums' formal definition and review.

Definition 5.4.2 *[Weighted Sums] The* **weighted rate sum** *for any rate vector* $\boldsymbol{b}$ *is*

$$b(\boldsymbol{\theta}) \triangleq \sum_{u=1}^{U} \theta_u \cdot b_u = \boldsymbol{\theta}^t \cdot \boldsymbol{b} \;\; , \;\; \boldsymbol{\theta} \succeq \mathbf{0} \;\; , \tag{5.295}$$

where $\boldsymbol{\theta} \in \mathbb{R}_{0+}^{U}$ *is a* $U \times 1$ *vector of nonnegative weights for individual users' bits/symbol elements* b_u*. Similarly, the* **weighted energy sum** *for any energy vector* $\boldsymbol{\mathcal{E}}$ *is*

$$\mathcal{E}(\boldsymbol{w}) \triangleq \sum_{u=1}^{U} w_u \cdot b_u = \boldsymbol{w}^t \cdot \boldsymbol{\mathcal{E}} \;\; , \;\; \boldsymbol{w} \succeq \mathbf{0} \;\; . \tag{5.296}$$

where $\boldsymbol{w} \in \mathbb{R}_{0+}^{U}$ *is a* $U \times 1$ *vector of nonnegative weights for individual users' energy elements* $\mathcal{E}_u$*.*

These weighted sums essentially assign relative user priority. Design may normalize either or both weight vectors to unity sum, effectively making the weights a user-priority probability distribution. However, any (postive-real) scaling factor is essentially superfluous and unit-sum need not be strictly observed. The setting $\boldsymbol{\theta} = \mathbf{1}$ corresponds to the users' rate sum, while $\boldsymbol{w} = \mathbf{1}$ corresponds to the energy sum, even though $\sum_{u=1}^{U} \theta_u = U = \sum_{u=1}^{U} w_u \neq 1$.

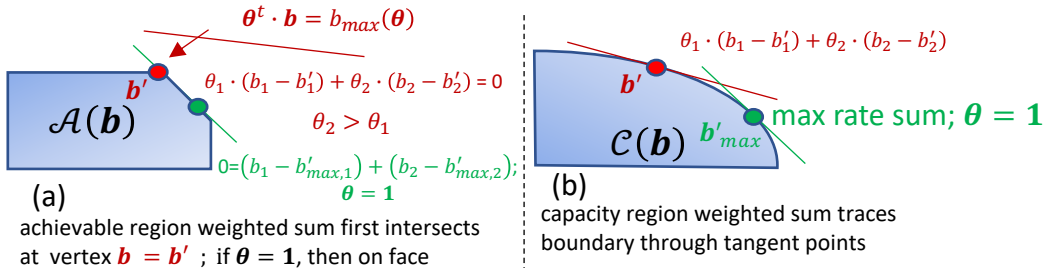


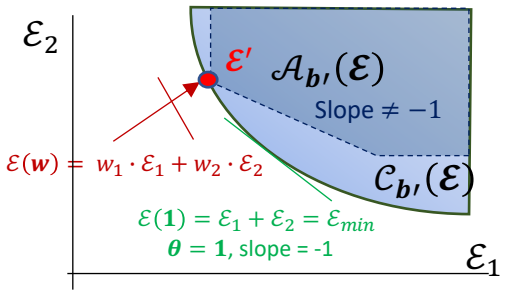
Figure 5.34: Weighted rate-sum illustration for $\mathscr{C}(\boldsymbol{b})$ tracing).

Rate region tangents: The weighted rate-difference sum $\boldsymbol{\theta}^t \cdot (\boldsymbol{b} - \boldsymbol{b}') = \boldsymbol{\theta}^t \cdot \Delta\boldsymbol{b} = 0$ describes a $(U-1)$-dimensional hyperplane that divides $\boldsymbol{b} \in \mathbb{R}^U$ into two halves with origin essentially translated to $\boldsymbol{b}'$. Figure 5.34(a)'s red line illustrates this in two dimensions. As the scalar value $b_{max}(\boldsymbol{\theta})$ decreases, this rate-sum hyperplane boundary (red line) first touches an achievable region $\mathcal{A}(\boldsymbol{b})$ at the maximum weighted rate-sum point $\boldsymbol{b}' \ni \mathbf{1}^t \cdot \boldsymbol{b}' = b_{max}(\theta)$. Figure 5.34(a)'s red line illustrates this by the pentagon's (polytope's) red vertex. Figure 5.34(a)'s red line shows a slope with magnitude less than 1, and thus in this case $\theta_2 < \theta 1$. The green rate vector represents points $\boldsymbol{b}$ on the $\mathcal{A}(\boldsymbol{b})$ boundary that sum to $b_{max}(\mathbf{1})$.

The achievable region $\mathcal{A}(\boldsymbol{b})$ corresponds to a specific $R_{\boldsymbol{xx}}$. This is visually clear in two dimensions, and Theorem 5.4.1 below proves the hyperplane first touches a polytope vertex as $b_{max}(\boldsymbol{\theta})$ decreases. For generalization to a capacity region, Figure 5.34(b) also shows the hyperplane first touches as a tangent to $\mathcal{C}(\boldsymbol{b})$ on the right. For a fixed $R_{\boldsymbol{xx}}$, $\mathcal{A}(\boldsymbol{b})$ is an polytope (pentagon in 2 dimensions) and any point on this polytope's extreme face corresponds to the same rate sum. The convex hull or union over all allowed $R_{\boldsymbol{xx}}$ such that $\boldsymbol{\mathcal{E}} =\preceq \boldsymbol{\mathcal{E}}'$, where each $\mathcal{E}_{x,u} = trace\{R_{\boldsymbol{xx}}(u)\} \leq \mathcal{E}_u$, forms the capacity region $\mathcal{C}(\boldsymbol{b})$, as also in Figure 5.34. Each $\boldsymbol{\theta}$ corresponds to a maximum weighted rate sum on the boundary $\mathscr{C}(\boldsymbol{b})$ and is tangent at that point. Thus the maximum weighted rate sums for all $\boldsymbol{\theta}$ trace $\mathscr{C}(\boldsymbol{b})$, as the next paragraph details. When $\boldsymbol{\theta} = \mathbf{1}$, this tangent point corresponds to the maximum rate sum (green and red become the same or lie in the polytope's extreme upper face in Figure 5.34(a)).

Equal weights: As always, recalling the mutual-information chain rule, all GDFE decoding orders achieve the same rate sum although with possibly different $b = \sum_{u=1}^{U} b_u$. When two or more θ_u values are identical, then the tangent intersects $\mathcal{A}(\boldsymbol{b})$ along an entire boundary or face and there are many data-rate vectors $\boldsymbol{b}'$ that all achieve the same maximum weighted rate sum, which effectively look like the face and green-line segment in Figure 5.34(a)'s. All GDFE decoding orders for those same-theta users, corresponding to all these same-theta users' order permutations, produce the same weighted rate sum - effectively making them a macro user. Indeed, more generally with arbitrary $\boldsymbol{\theta} \succeq \mathbf{0}$, the hyperplane $\boldsymbol{\theta}^t \cdot \boldsymbol{b}$ will be tangential to the capacity-region boundary at some point $\boldsymbol{b}^*_{\boldsymbol{\theta}}$ where this weighted rate sum is maximum over $\boldsymbol{b}$ for this specific $\boldsymbol{\theta}$ choice. The capacity region's (non trivial) boundary $\mathscr{C}(\boldsymbol{b})$ corresponds to enumerating and taking the convex hull of all the data rate points corresponding to different $\boldsymbol{\theta} \succeq \mathbf{0}$.

$$\mathscr{C}(\boldsymbol{b}) = \bigcup_{\boldsymbol{\theta} \succeq \mathbf{0}}^{hull} \boldsymbol{b} = \arg \max_{\boldsymbol{b}' \in \mathcal{C}(\boldsymbol{b})} \{\boldsymbol{b}'(\boldsymbol{\theta})\} \quad . \tag{5.297}$$



achievable region ; the weighted energy sum first intersects at vertex
The slope= -1 line does not (usually) bound $\mathcal{A}_{\boldsymbol{b}'}(\boldsymbol{\mathcal{E}})$

capacity-energy region
Minimized weighted energy sum traces boundary through tangent points and
$\boldsymbol{\theta} = \mathbf{1}$ for minimum energy sum

Figure 5.35: Weighted energy-sum illustration for $\mathscr{C}_{\boldsymbol{b}'}(\boldsymbol{\mathcal{E}})$ tracing) first touches.

Energy region tangents: Similarly, Figure 5.35 illustrates the energy capacity region $\mathcal{C}_{\boldsymbol{b}'}(\boldsymbol{\mathcal{E}})$, an achievable region corresponding to a specific $R_{\boldsymbol{xx}}$ within the energy-capacity region $\mathcal{C}_{\boldsymbol{b}'}(\boldsymbol{\mathcal{E}})$. A weighted (red line) energy sum $\mathcal{E}(\boldsymbol{w})$ that first touches the achievable region and is tangent to $\mathcal{E}_{\boldsymbol{b}'}$ at the point $\boldsymbol{\mathcal{E}}'$. The minimum energy-sum point where the green line with slope -1 is tangent. The polytope achievable region $\mathcal{A}_{\boldsymbol{b}'}(\boldsymbol{\mathcal{E}})$ does not have slope on its lowest face of -1 in two dimensions, or equivalently is not orthogonal to the vector $\mathbf{1}$ more generally. When there are $N_{\boldsymbol{w}}$ equal values of w_u, this corresponds to an $U_{\boldsymbol{w}}$ dimensional face where all energy sums for the corresponding users are the same. Unlike the rate sum, such a face need not correspond to a time-sharing of extreme vertices of some single underlying polytope of an achievable region. Nonetheless, through convex hull operations, the points on the face have different user energies but all such $U_{\boldsymbol{w}}$ user sets have the same energy sum. Similar to the capacity region, the capacity-energy region is traced by the weighted energy sums:

$$\mathscr{C}_{\boldsymbol{b}'}(\boldsymbol{\mathcal{E}}) = \bigcup_{\boldsymbol{w} \succeq \mathbf{0}}^{hull} \boldsymbol{\mathcal{E}} = \arg \max_{\boldsymbol{\mathcal{E}}' \in \mathcal{C}_{\boldsymbol{b}'}(\boldsymbol{\mathcal{E}})} \{\boldsymbol{\mathcal{E}}'(\boldsymbol{w})\} \quad . \tag{5.298}$$

Theorem 5.4.1 Maximum Weighted Rate Vertex Optimum:
For any given specific $\{R_{\boldsymbol{xx}}(u)\}$*, the maximum weighted rate sum, or hyperplane,*

$$b_{max} = \max_{\boldsymbol{\theta} \geq \mathbf{0}} \left\{ \sum_{u=1}^{U} \theta_u \cdot b_u \right\} \tag{5.299}$$

must occur at an achievable-region MAC vertex. Further the decoding order $\pi(u)$ *at that vertex has*

$$\theta_{\pi^{-1}}(U) \geq \theta_{\pi^{-1}}(U-1) \geq ... \geq \theta_{\pi^{-1}}(1) \ . \tag{5.300}$$

Proof: There are $U!$ GDFE-point vertices for a given block-diagonal $R_{\boldsymbol{xx}}$ that define the maximum rate-sum hyperplane of the polytope $\mathcal{A}(R_{\boldsymbol{xx}}, \boldsymbol{b})$. This proof uses induction, beginning with Figure 5.34(a). The reducing $b_{max}(\boldsymbol{\theta})$ for a fixed $\boldsymbol{\theta}$, so viewed as a function of $\boldsymbol{b}$, there has $\theta_2 > \theta_1$. This line clearly intersects the polytope (pentagon) at point $\boldsymbol{b}'$. If $\theta_1 > \theta_2$, the line slope would be steeper than 45 degrees and touch first the other vertex on the green line. When $\theta_2 = \theta_1$, the lines overlap when $b_{max}(\mathbf{1} \cdot c) = b'_1 + b'_2$. By induction, users 1 and 2 have a sum that is viewed as a macro user, and the proof repeats for the macro user as user 2 and the new user as user 3. So more generally any hyperplane $\boldsymbol{\theta}^* \cdot (\boldsymbol{b} - \boldsymbol{b}') = 0$ intersects with reducing the scalar $b_{max}(\boldsymbol{\theta}) = \boldsymbol{\theta}^* \cdot \boldsymbol{b}$, first touching at an polytope $\mathcal{A}(R_{\boldsymbol{xx}}, \boldsymbol{b})$ vertex where then the intersection point is $c' = \boldsymbol{\theta}^* \cdot \boldsymbol{b}'$. All tonal GDFE's use that order $\boldsymbol{\pi}$ for this given $R_{\boldsymbol{xx}}$ because $\boldsymbol{\theta}$ does not depend on n. This is also true then for the overall energy-minimizing $R_{\boldsymbol{xx}}$. That best order corresponds to

$$\theta_{\pi^{-1}(U)} \geq \theta_{\pi^{-1}(U-1)} \geq ... \geq \theta_{\pi^{-1}(1)} \ , \tag{5.301}$$

which follows this inductive argument. Thus, the $\boldsymbol{\theta}$, or equivalently intersection-vertex' GDFE order corresponds best position (top) corresponding to the smallest θ_u value and worst position (bottom) corresponding to largest θ_u value. **QED.**

If the max-rate-sum hyperplane is $\boldsymbol{\theta} = \mathbf{1} \cdot c$, where c is any scalar constant, then many points including all vertices on the face share this same maximum rate sum. Similarly, Theorem 5.4.1 has an equivalent statement for the weighted energy sum and the energy-capacity region, although again the face of an achievable region for the energy-capacity region need not correspond to an (unweighted) energy sum. These theorems do not depend on $\overline{N}$ if the system is vector-DMT/OFDM based.

Simultaneous Water Fill is only for maximum rate sum: This maximum rate-sum point with $\boldsymbol{\theta} = \mathbf{1}$ corresponds as in Section 5.4.2.2 to a simultaneous water-fill solution where each user sees all others as noise. This point can be achieved by iterative water-filling wthe system is Vector DMT[67]. However, when $\boldsymbol{\theta} \neq \mathbf{1}$, or equivalently not equal to some positive constant vector $c \cdot \mathbf{1}$ where $c > 0$, the best design point is not SWF. The next subsections address optimization's expansion to any point $\boldsymbol{b}' \in \mathcal{C}(\boldsymbol{b})$ (or any point $\boldsymbol{\mathcal{E}}' \in \mathcal{C}_{\boldsymbol{b}}(\boldsymbol{\mathcal{E}})$ in the capacity-energy region).

multiuser margin: Rate vectors $\boldsymbol{b} \in \mathcal{C}_{MAC}(\boldsymbol{b})$ can have multiple viable designs, even for the same energy vector $\boldsymbol{\mathcal{E}}$; however $\boldsymbol{b}$ on the capacity region border[68] $\boldsymbol{b} \in \mathscr{C}(\boldsymbol{b})$ often have only one design and correspond to a specific energy-weight vector. However, rate vectors $\boldsymbol{b}'$ in the interior $\mathcal{S}(\boldsymbol{b})$ have a gap γ_b that corresponds to to the closest border point $\boldsymbol{b}^* \in \mathscr{C}(\boldsymbol{b})$ where $\boldsymbol{b}^* = \boldsymbol{b} + \gamma_b \cdot \mathbf{1}$, as in Chapter 2. The scalar $\gamma_b \geq 0$ essentially measures the largest fixed number of bits that is equally added to all users

[67] recall Vector OFDM does NOT optimize bit/energy distributions so calling it "simultaneous water-fill OFDM" would be an oxymoron.

[68] Recalling from Chapter 2 that the capacity-region border excludes the capacity region's largest open subregion $\mathcal{S}(\boldsymbol{b}) \subset \mathcal{C}(\boldsymbol{b})$, so $\mathscr{C}(\boldsymbol{b}) \triangleq \mathcal{C}(\boldsymbol{b}) \setminus \mathcal{S}(\boldsymbol{b})$.

components in $\boldsymbol{b}'$ such that the corresponding point is in the boundary. Then, the SNR based multi-user margin gap is approximately $\gamma_m = \gamma_b \cdot 6$ dB/dimension (or 3 dB/complex subsymbol for complex baseband). The approximation becomes tight as the user bits/dimension grow, but works well for $\bar{b} \geq 1$. Section 2.6's **proportional fairness**. with QPS (queue-proportional scheduling) chooses $\boldsymbol{b}' = k \cdot \boldsymbol{q}$, where $\boldsymbol{q}$ represents the waiting queue depth (number of bits waiting in queue to be sent) and $k \in \mathbb{R}^+$ is a scalar positive constant that helps choose $\boldsymbol{b}$ while maintaining γ_b.

5.4.4 Weighted-Sum Optimization

There are two weight-sum-related MAC design criteria[69]:

1. **Minimize the weighted energy sum $\mathcal{E}_{\boldsymbol{x}}$ for a given $\boldsymbol{b}$:** This criterion minimizes weighted-energy sum with specified energy weights $\boldsymbol{w} \succeq 0$, and with $\boldsymbol{\theta} \succeq 0$ as side-constraint multipliers to minimize (maximize with negative sign) rate-constraint impact.

$$\min_{\{R_{\boldsymbol{x}\boldsymbol{x}}(u)\}} \quad \sum_{u=1}^{U} w_u \cdot \text{trace}\underbrace{\{R_{\boldsymbol{x}\boldsymbol{x}}(u)\}}_{\mathcal{E}_u} \tag{5.302}$$
$$ST: \quad \boldsymbol{b} \succeq [b_{1,min}\ b_{2,min}\ ...b_{U,min}]^* = \boldsymbol{b}^*_{min} \succeq \boldsymbol{0}$$
$$\boldsymbol{\mathcal{E}} \succeq \boldsymbol{0} \quad .$$

   The vector $\boldsymbol{w} = \boldsymbol{1}$ corresponds to the energy sum; more generally $\boldsymbol{w} \succeq \boldsymbol{0}$. Theoretically, (5.302) always has a solution for a given $\boldsymbol{b}$.

2. **Maximize the weighted rate sum** This criterion maximizes weighted-rate sum with $\boldsymbol{\theta} \succeq 0$ specified with $\boldsymbol{w} \succeq 0$ as the side-constraint multiplier vector to minimize (maximize with negative sign) energy-constraint impact, with $\boldsymbol{\mathcal{E}}_{\boldsymbol{x},u} = trace\,\{R_{\boldsymbol{x}\boldsymbol{x}}(u)\}$

$$\max_{\{R_{\boldsymbol{x}\boldsymbol{x}}(u)\}} \quad \sum_{u=1}^{U} \theta_u \cdot b_u \tag{5.303}$$
$$ST: \quad \boldsymbol{\mathcal{E}}_{\boldsymbol{x}} \preceq [\mathcal{E}_{1,max}\ \mathcal{E}_{2,max}\ ...\mathcal{E}_{U,max}]^* = \boldsymbol{\mathcal{E}}^*_{max} \succeq \boldsymbol{0}$$
$$\boldsymbol{b} \succeq \boldsymbol{0} \quad .$$

   The vector $\boldsymbol{\theta} = [11...1]^*$ corresponds to the rate sum; more generally $\boldsymbol{\theta} \succeq \boldsymbol{0}$. Theoretically, (5.303) always has a solution for a given $\boldsymbol{\mathcal{E}}_x$.

Lagrangians: There is a corresponding common Lagrangian term in the two corresponding expressions (which uses a minus sign in the second term to reverse the minimax):

$$L_{minE}(R_{\boldsymbol{x}\boldsymbol{x}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \underbrace{\min_{R_{\boldsymbol{x}\boldsymbol{x}}} \sum_{u=1}^{U} [w_u \cdot trace\,\{R_{\boldsymbol{x}\boldsymbol{x}}(u)\} - \theta_u \cdot b_u}_{\text{common term}} + \theta_u \cdot b_{min,u}] \tag{5.304}$$

$$L_{maxR}(R_{\boldsymbol{x}\boldsymbol{x}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta}) = \max_{\boldsymbol{w}} \underbrace{\min_{R_{\boldsymbol{x}\boldsymbol{x}}} \sum_{u=1}^{U} [-w_u \cdot trace\,\{R_{\boldsymbol{x}\boldsymbol{x}}(u)\} + \theta_u \cdot b_u}_{\text{common term}} + w_u \cdot \mathcal{E}_{max,u}]\,; \tag{5.305}$$

These two expressions both individually assume a zero duality gap for their respective optimizations, which will hold under developments to follow. The difference is in the interpretation of the parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$ as given constant and Lagrange multiplier (or vice versa) and in the rightmost incremental-change-constraint terms. A design that targets a minimum rate vector $\boldsymbol{b}_{min}$ for a maximum energy vector $\boldsymbol{\mathcal{E}}_{max}$ essentially optimizes both Lagrangians simultaneously. This creates interdependency between $\boldsymbol{\theta}$ and $\boldsymbol{w}$ where neither remains given, but instead the design jointly optimizes them both, finds $R_{\boldsymbol{x}\boldsymbol{x}}$, and ensures the data rate and energy constraint are both met. The minimum weighted energy-sum problem also will be more helpful in later BC-duality designs, where only all users' sum energy has a limit.

[69] Initially with capacity achieving codes presumed and thus $\Gamma = 0$ dB

5.4.4.1 Vector DMT Extension

Equation (5.302) specializes with vector DMT to

$$\min_{\{R_{\boldsymbol{XX}}(u,n)\}} \quad \sum_{u=1}^{U}\sum_{n=0}^{\overline{N}} w_u \cdot \text{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\} \tag{5.306}$$

$$ST: \quad \boldsymbol{b} = \sum_{n=0}^{\overline{N}} [b_{1,n}\ b_{2,n}\ ...b_{U,n}]^* \succeq \boldsymbol{b}_{min} \succeq \boldsymbol{0} \quad .$$

$$b_{u,n} \geq 0 \ \ \forall\ u,n$$

A similar expression holds for Equation (5.303)'s Vector DMT form. Each tone can use an independent GDFE with the set of maximum-sum energies $\boldsymbol{\mathcal{E}}_{max}$ and/or $\boldsymbol{b}_{min}$, summed over all tones.

Equation (5.304)'s, or (5.306)'s, Lagrangian is

$$L(R_{\boldsymbol{XX}}(n), \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{u=1}^{U}\left(w_u \cdot \left[\sum_{n=0}^{\overline{N}} \text{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\}\right] - \theta_u \cdot \left\{\left[\sum_{n=0}^{\overline{N}-1} b_{u,n}\right] - b_u\right\}\right) \tag{5.307}$$

(presuming no single-side-band uses of tone $n = \overline{N}$ tone if baseband real). Again a similar expression follows for the maximized weighted rate sum.

To complete the optimization, the per-tone-user bit distribution $\{b_{u,n}\}$ and corresponding energy/$R_{\boldsymbol{xx}}$ distribution $\{R_{\boldsymbol{xx}}(u)\}$ for all tones $n = 1, ., , , \overline{N}$ must each be within the corresponding tonal-MAC's matrix-AWGN ($\widetilde{H}_n$) achievable region:

$$\boldsymbol{b}_n \in \mathcal{A}_n(R_{\boldsymbol{XX}}(u,n), \boldsymbol{b}_n)\ ,\ R_{\boldsymbol{XX}}(u,n), u = 1, ..., U\ . \tag{5.308}$$

More explicitly, $\boldsymbol{b}_n$ and $R_{\boldsymbol{XX}}(n)$ must satisfy (for all $n = 0, ., , , \overline{N}-1$)

$$\boldsymbol{b}_n \in \left\{\boldsymbol{b}_n \mid 0 \leq \sum_{\boldsymbol{u}\subseteq\boldsymbol{U}} b_{u,n} \leq \log_2\left|\left(\sum_{u=1}^{U}\widetilde{H}_{u,n}\cdot R_{\boldsymbol{XX}}(u,n)\cdot\widetilde{H}^*_{u,n}\right)+I\right|\right\} = \mathcal{A}_n\left(\left\{R_{\boldsymbol{XX}}(n)\right\}, \widetilde{H}_n\right) \quad , \tag{5.309}$$

where again

$$\widetilde{H}_{u,n} = R^{-1/2}_{\boldsymbol{NN}}(n)\cdot H_{u,n} \quad . \tag{5.310}$$

Computation of tonal mutual information: The relationship between the bits per user/tone $b_{u,n}$ and tone autocorrelation $R_{\boldsymbol{XX}}(u,n)$ follows from the known best order (that is at given $\boldsymbol{\theta}$), and the tonal-GDFE/achievable-region relation, which is the difference between two chain-rule mutual information terms), becomes (with zero gap)

$$\begin{aligned} b_{u,n} &= \underbrace{\log_2\left|\sum_{i=u}^{U}\widetilde{H}_{\pi^{-1}(i),n}\cdot R_{\boldsymbol{XX}}(\pi^{-1}(i),n)\cdot\widetilde{H}^*_{\pi^{-1}(i),n}+I\right|}_{\beta_{u,n}\triangleq\sum_{i=u}^{U}b_{u,n}} \\ &- \underbrace{\log_2\left|\sum_{i=u+1}^{U}\widetilde{H}_{\pi^{-1}(i),n}\cdot R_{\boldsymbol{XX}}(\pi^{-1}(i),n)\cdot\widetilde{H}^*_{\pi^{-1}(i),n}+I\right|}_{\beta_{u+1,n}} \quad . \end{aligned} \tag{5.311}$$

The quantity $\beta_{u,n}$ is the rate sum for user u and users $i > u$ in the current order $\pi(u)$. When computing the sum $\sum_{u=1}^{U}\theta_u\cdot b_{u,n}$ from (5.295), each "log" term in (5.307) appears twice with different value of θ_i,

so with $\theta_{\pi^{-1}(U+1)} \triangleq 0$, the sum simplifies to

$$\sum_{u=1}^{U} \theta_u \cdot b_{u,n} = \sum_{u=1}^{U} \left\{ \underbrace{\left[\theta_{\pi^{-1}(u)} - \theta_{\pi^{-1}(u+1)}\right]}_{\delta_{\pi^{-1}(u)} \leq 0} \cdot \log_2 \left| \sum_{i=u}^{U} \widetilde{H}_{\pi^{-1}(i),n} \cdot R_{\boldsymbol{XX}}(\pi^{-1}(i), n) \cdot \widetilde{H}^{*}_{\pi^{-1}(i),n} + I \right| \right\} \quad . \tag{5.312}$$

Implied Order: Equation (5.312) relates the weighted rate sum to the tonal autocorrelation matrices for order $\pi(u)$. Only the order with $\theta_{\pi^{-1}(u)} \leq \theta_{\pi^{-1}(u+1)} \; \forall \; u = 1...U-1$ ensures that (5.312) remains convex for its minimization[70] over $R_{\boldsymbol{XX}}(u,n)$. This same observation occurs simply in Theorem 5.4.1's proof, specifically Equation (5.301). This particular order corresponds to the size of θ_i and thus has the smallest θ_i corresponding to the worst, or lowest, GDFE decoding position. So with a $\boldsymbol{\theta}$ (presumably optimized under (5.312)), the user with largest θ_u goes last, successively decreasing with θ_u size to the smallest-θ_u corresponding its user going first. Indeed, specification of a $\boldsymbol{\theta}$ implies an order with the smallest $\theta_u < \theta_i \; \forall i \; \neq u$ in less favored positions and so on with $\theta_u > \theta_i \; \forall \; i \neq u$ is in better positions.

Equally Important User Order: Equal $\theta_u = \theta_{u+1}$ implies order ambivalence in that these successive correspond users can be swapped (or rotated) but still produce the same rate sum for these succesive users without affecting users before or after this successive-user set within the order. Thus every tone's bit distribution ($\boldsymbol{\theta}$ is not a function of n) has a convex function and thus the entire Lagrangian is convex with this new parametrization by weighting vector $\boldsymbol{\delta}$. The case of zeroed $\boldsymbol{\delta}$ elements will shortly require some special handling to ensure constraints are met fully.

Optimizing the Order: The concavity/convexity tacitly implies that

$$\theta_{\pi^{-1}(U)} \geq \theta_{\pi^{-1}(U-1)} \geq ... \geq \theta_{\pi^{-1}(1)} \geq 0 \; \forall \; u = 1, ..., U \quad . \tag{5.313}$$

Thus, every time $\boldsymbol{\theta}$ changes in any optimizing algorithm, the user order must also change if $\boldsymbol{\theta}$'s nonnegative elements' relative sizes change; this is necessary to retain $\boldsymbol{\delta} \preceq \mathbf{0}$. Indeed, finding a best $\boldsymbol{\theta}$ implies a constant order and specific vertex. This is the vertex that is first touched by $b(\boldsymbol{\theta})$ in Figure 5.34. Instead of optimizing $\boldsymbol{\delta}$, any incremental optimization algorithm for $\boldsymbol{\theta}$ must remember to change $\boldsymbol{\pi}$ to correspond to descending order of $\boldsymbol{\theta}$. Thus the procedure determines best order when converged. Degraded MACs can have linearly dependent $\widetilde{H}_u$, or equivalently two or more user components combine effectively into a macro-user. Such degraded channels will have the above equal-$(\theta_i = \theta_{i+1})$ situation in (5.312), or a $\delta_i = 0$ for at least one $i \in \boldsymbol{U}$. This situation corresponds a need for vertex sharing. The optimizations in (5.302) and (5.303) with the restriction in (5.312) presume optima occur at a vertex. Thus, design for such (very common) situations will separately determine such vertex sharing to ensure all constraints are met.

Weight Rate Sum Maximization Comment: A similar development arises for (5.311)'s weighted-rate sum maximization where $\boldsymbol{\theta}$ is given fixed, and so the order is predetermined in the weighted rate sum, but otherwise the optimization remains concave (convex with negative sign). The vector $\boldsymbol{w}$ is instead found to ensure the energy-vector constraint is met.

Overall Design Approach: The design approach begins with the weighted rate-sum problem

$$\begin{aligned} \max_{\{R_{\boldsymbol{XX}}(u,n)\}} \quad & \sum_{u=1}^{U} \theta_u \cdot \left\{ \sum_{n=0}^{\overline{N}} b_{u,n} \right\} \\ ST: \quad & \boldsymbol{\mathcal{E}} \succeq \sum_{n} \text{trace}\left\{R_{\boldsymbol{XX}}(u,n)\right\} \succeq \mathbf{0} \quad . \end{aligned} \tag{5.314}$$

[70] Equations (5.307 - (5.311) minimize energy for a given $\boldsymbol{b}_{min}$. The sum data rate's increase assists minimization (reduction) of the Lagrangian if these terms are negative, which means the data rate grows; correspondingly if positive, rate decreases and reduces the Lagrangian.

When the resultant data rate solution $\boldsymbol{b}' \in \mathcal{C}(\boldsymbol{b})$ has user data rate components that do not meet the $\boldsymbol{b}' \succeq \boldsymbol{b}_{min}$ requirement, then the given $\boldsymbol{\theta}$ does not correspond to the desired data rate vector $\boldsymbol{b}_{min}$. Thus, the $\boldsymbol{\theta}$ needs to change. The corresponding Lagrange multipliers $\boldsymbol{w}$ for $\boldsymbol{\mathcal{E}}_{max}$ effectively also change, implying a different corresponding minimum weighted energy-sum corresponding to $\boldsymbol{b}_{min}$ has not yet been found. However, each $\boldsymbol{b}$ found is a vertex corresponding to the maximum-weighted-rate-sum $R_{\boldsymbol{xx}}$ found for the current $\boldsymbol{\theta}$. This vertex might be used in vertex-sharing to obtain the final desired $\boldsymbol{b}_{min}$. The update of $\boldsymbol{\theta}$ occurs in minimizing the weighted energy sum where it is a Lagrange multiplier that can be optimized.

5.4.4.2 Energy ($R_{\boldsymbol{xx}}$)) Minimization

The steepest descent of the Lagrangian simplifies computationally through use of tonal decomposition, which then allows individual tonal calculations of gradients and Hessian matrices for steepest descent.

Tonal Decomposition of the Lagrangian: Interchange of Equation (5.307)'s finite sums over indices u and n obtains[71]

$$L_{minE}(R_{\boldsymbol{XX}}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\theta}) = \left(\sum_{u=1}^{U} \theta_u \cdot b_u \right) + \sum_{n=0}^{\overline{N}-1} \left\{ \underbrace{\sum_{u=1}^{U} \left[w_u \cdot \text{trace}\left\{ R_{\boldsymbol{XX}}(u,n) \right\} - \theta_u \cdot b_{u,n} \right]}_{L_n(R_{\boldsymbol{XX}}(n), \boldsymbol{b}_n, \boldsymbol{w}, \boldsymbol{\theta})} \right\} \quad , \tag{5.315}$$

where $L_n(R_{\boldsymbol{XX}}(n), \boldsymbol{b}_n, \boldsymbol{w}, \boldsymbol{\theta})$ is a **tonal Lagrangian** term. For a given $\boldsymbol{\theta}$, (5.315)'s first right-side sum term is not a function of $R_{\boldsymbol{XX}}(u,n)$, because b_u is given, even though its exact constituency $b_{u,n}$ is not ($\overline{N} > 1$). $L_n(R_{\boldsymbol{XX}}(n), \boldsymbol{b}_n, \boldsymbol{w}, \boldsymbol{\theta})$ for any given $\boldsymbol{\theta}$ and $\boldsymbol{w}$ depends only on tone n quantities. Thus, the overall "max-min" (with $\boldsymbol{w}$ given and not explicitly shown and $R_{\boldsymbol{XX}}$ also absent because its value corresponds to the minimum in (5.315). is

$$L^*_{minE} = \max_{\boldsymbol{\theta}} \left\{ \left[\sum_{n=0}^{\overline{N}-1} L_{minE}(\boldsymbol{\theta}, n) \right] + \underbrace{\sum_{u=1}^{U} \theta_u \cdot b_u}_{\text{independent of } R_{\boldsymbol{XX}}(u,n),n} \right\} \tag{5.316}$$

where for each n and $\boldsymbol{\theta}$, noting $R_{\boldsymbol{XX}}(n) = \text{blkdiag}\left\{ R_{\boldsymbol{XX}}(U,n), ..., R_{\boldsymbol{XX}}(1,n) \right\}$,

$$L_{minE}(\boldsymbol{\theta}, n) \triangleq \min_{\{R_{\boldsymbol{XX}}(u,n) b_{u,n}\},} L(R_{\boldsymbol{XX}}(n), \boldsymbol{b}_n, \boldsymbol{w}, \boldsymbol{\theta}) \quad . \tag{5.317}$$

The last term in (5.316) has the given fixed b_u (from the given $\boldsymbol{b}$) values in the sum. (5.317)'s minimization is thus $\overline{N}$ independent minimizations for a given $\boldsymbol{\theta}$. The outer $\boldsymbol{\theta}$ maximization proceeds after these $\overline{N}$ inner minimizations (and is not part of the $R_{\boldsymbol{xx}}$ steepest descent optimization for each tone n).

Summing the solutions of (5.317) over the tones provides

$$L_{minE}(\boldsymbol{\theta}) = \sum_{n=0}^{\overline{N}-1} L_{minE}(\boldsymbol{\theta}, n) \quad , \tag{5.318}$$

to be mazimized over $\boldsymbol{\theta}$. This $L_{minE}(\boldsymbol{\theta})$ has maximum at θ^o, as in (5.316), for

$$L(\theta^o) = \sum_{u=1}^{U} \theta^o_u \cdot \underbrace{b_u}_{\text{known}} + \underbrace{L_{min}(\boldsymbol{\theta}^o)}_{\substack{\text{point in} \\ \mathcal{A}(R_{\boldsymbol{XX}}, \widetilde{H})}} \tag{5.319}$$

[71] The frequency-space $R_{\boldsymbol{XX}}$ notation will use non-subscription arguments to include user and tonal dimensions, e.g. $R_{\boldsymbol{XX}}(u,n)$ is an $L_{x,u} \times L_{x,u}$ smallest matrix, while $R_{\boldsymbol{XX}}(u)$ includes all tones as a 3-dimensional tensor and similarly $R_{\boldsymbol{XX}}(n)$ includes all users as a 3-dimensional tensor, while $R_{\boldsymbol{xx}}$ includes all tones and all users for a 4-dimensional tensor. $R_{\boldsymbol{XX}}$ is often a 4-dimensional array/tensor in vector MAC software programs.

Gradients and Hessians for $R_{\boldsymbol{xx}}$ optimization: Appendix C covers matrix-calculus basics that are helpful for those desiring more complete derivational details. The $R_{\boldsymbol{xx}}$ step can use a gradient-descent method independently for each tone. Basically, steepest descent minimizes the Lagrangian $L_{min}(\boldsymbol{\theta}, n)$ incrementally in the negative-gradient direction according to

$$R_{\boldsymbol{xx}}(\ell, n, k+1) = R_{\boldsymbol{xx}}(\ell, n, k) - \mu \cdot \left(\nabla^2 L_k\right)^{-1}_{n,k} \cdot \nabla L_{n,k} \tag{5.320}$$

where the gradient matrix simplifies[72] to a block-cell vector:

$$\nabla L_{n,k} \triangleq \begin{bmatrix} \nabla_{R_{\boldsymbol{XX}},U}(n,k) \\ \vdots \\ \nabla_{R_{\boldsymbol{XX}},1}(n,k) \end{bmatrix} \tag{5.321}$$

This simplification exploits the MAC's block-diagonal $R_{\boldsymbol{xx}}(n)$ requirement to compress the number of quantities into a cell-array of $L_{x,u} \times L_{x,u}$ components, reducing to the energy vector $\boldsymbol{\mathcal{E}}$ when $L_{x,u} = 1$, and a block-column vector with constant $L_{x,u} = L_x > 1$. The corresponding Hessian has cell-matrix components (necessarily square when $\ell = j$ below) that expand each cell of $\nabla_{R_{\boldsymbol{XX}}(u,n}$ into a larger row set with $\ell = 1, ..., U$ and $j = 1, ..., U$ such that

$$\nabla^2_{R_{\boldsymbol{XX}}} L(\ell, j, n, k) = \begin{bmatrix} \nabla_{R_{\boldsymbol{XX}},U,U}(\ell, j, n, k) & ... & \nabla_{R_{\boldsymbol{XX}},U,1}(\ell, j, n, k) \\ \vdots & \ddots & \vdots \\ \nabla_{R_{\boldsymbol{XX}},1,U}(\ell, j, n, k) & ... & \nabla_{R_{\boldsymbol{XX}},1,1}(\ell, j, n, k) \end{bmatrix} , \tag{5.322}$$

and which can be organized into full Hessian as a matrix of component matrices. The product $\left(\nabla^2 L_k\right)^{-1}_{\ell,j} \cdot \nabla L_k$ is of the same size as $R_{\boldsymbol{XX}}$ and updates it in a steepest-descent direction along a column-array that stacks vertically the U matrices $R_{\boldsymbol{XX}}(u, n)$ that are the nonzero components along the block-diagonal matrix $R_{\boldsymbol{XX}}(n)$.

Using (5.311), the gradient has U components (with $R^{-1}_{\boldsymbol{YY}}(0) = I$ and letting $R^{-1}_{\boldsymbol{YY}}(j)$ be the matrix inside the determinants), and presuming the order notation simplifies[73] to just j instead of the π^{-1} notation, while using the chain rule of derivatives (index n suppressed to $R_{\boldsymbol{YY}}(u) \triangleq \sum_{j=u}^{U} \widetilde{H}_u \cdot R_{\boldsymbol{XX}}(u, n) \cdot \widetilde{H}^*_u + I$).

$$\begin{aligned} \nabla L_{R_{\boldsymbol{XX}}(u)} &= \frac{1}{\ln(2)} \cdot \sum_{j=u}^{U} \theta_j \cdot \left| \widetilde{H}^*_u \cdot R^{-1}_{\boldsymbol{YY}}(j) \cdot \widetilde{H}_u \right| - w_u \cdot I + \sum_{i=1}^{L_{x,u}} \lambda^{-1}_{R_{\boldsymbol{XX}}}(i) \quad \text{where for} \quad u = 1...U \\ R^{-1}_{\boldsymbol{YY}}(u) &= R^{-1}_{\boldsymbol{YY}}(u-1) - R^{-1}_{\boldsymbol{YY}}(u-1) \cdot \widetilde{H}_u \cdot \left[R^{-1}_{\boldsymbol{XX}}(u) + \widetilde{H}^*_u \cdot R^{-1}_{\boldsymbol{YY}}(u-1) \cdot \widetilde{H}_u \right]^{-1} \cdot \widetilde{H}^*_u \cdot R^{-1}_{\boldsymbol{YY}}(u-1) \ . \end{aligned} \tag{5.323}$$

The $\sum_{i=1}^{L_{x,u}} \lambda^{-1}_{R_{\boldsymbol{XX}}}(i)$ term corresponds to a simplified non-negative user-energy side constraint in the form of individual $\sum_{\ell=1}^{L_{x,u}} \ln(\lambda^{-1}_{R_{\boldsymbol{XX}}}(\ell))$ singular values' logarithms penalizing infinitely any user's negative energy. The Hessian has U^2 components ($\ell = 1, ..., U$, $j = 1, ..., U$)

$$\nabla^2 L_u(\ell, j) = \frac{1}{\ln(2)} \cdot \left[\sum_{i=\max(u,\ell)}^{U} \theta_i \cdot trace \left\{ \widetilde{H}_u \cdot \widetilde{H}^*_u \cdot R^{-2}_{\boldsymbol{YY}}(i) \cdot \widetilde{H}_u \cdot \widetilde{H}^*_u \right\} \right] - \left\{ \lambda^{-2}_{R_{\boldsymbol{XX}}}(u) \right\} \ . \tag{5.324}$$

This Hessian's inverse will exist when the channel is non-degraded; otherwise a pseudoinverse is necessary. The pseudoinverse need occurs when the optimization becomes singular in that at least two successive θ values satisfy $\theta_u = \theta_{u-1}$, or equivalently $\delta_u = 0$. The steepest descent basically can only "see" the two

[72] Cells allow elements of what nominally appears as a vector or matrix with constant element sizes to generalize to variable sizes (like $L_{x,u}$).

[73] This reorders users indices by the current $\boldsymbol{\theta}$, which can occur after each order step. The relationship to the original under indices must be stored for eventual final $\pi(u)$.

corresponding users' rate sum as effectively a macro user. More on this singularity arises in Subsection 5.4.4.3's $\boldsymbol{\theta}$ optimization, along with a way to enumerate corresponding situations and then time-share vertices to meet the exact $\boldsymbol{b}_{min}$ constraint.

(5.320)'s positive step size μ must be less than half the inverse of the Hessian's trace to converge. The Hessian calculation is complex, but contained within a later-provided matlab subroutine. The proper Hessian calculation also includes constraints that ensure the autocorrelation matrices are positive semi-definite. The ratio $(\nabla^2 L_k^{-1}) \cdot \nabla L_k$ also must be real because L_k is real[74]. Complex gradient and Hessian matrices need to follow Appendix C's more involved relationships.

5.4.4.3 Meeting the rate target and optimization of θ:

The $\boldsymbol{\theta}$ update occurs as a weighted-rate-sum-minimizing Lagrange multiplier, where $\boldsymbol{w}$ is instead a given parameter. Update of $\boldsymbol{\theta}$ can also alter (5.313)'s implied order. The variable $\boldsymbol{\theta}$'s iterative optimization can use an elliptical-descent algorithm. The $\boldsymbol{\theta}$ optimization is for the current input $R_{\boldsymbol{XX}}(u,n,k)$, on possible some iteration k, which auguments here temporarily yet another index. This $R_{\boldsymbol{XX}}$ may not yet meet the $\boldsymbol{b}_{min}$ constraint with the corresponding set of $b_{u,n}(k)$ because the weighted rate-sum was instead maximized for a given energy vector (or sum-energy) constraint. The energy-sum Lagrangian is a function of $\boldsymbol{\theta}$ as

$$L^o_{minE}(\boldsymbol{\theta}) = \sum_{n=1}^{\overline{N}} \sum_{u=1}^{U} L_{minE,n}(R_{\boldsymbol{XX}}(u,n,k), b_{u,n}(k), \boldsymbol{w}, \boldsymbol{\theta}) \quad . \tag{5.325}$$

The $\boldsymbol{\theta}$ optimization evaluates (5.325) at an offset from the previous fixed $\boldsymbol{\theta}$ (used in $R_{\boldsymbol{XX}}$ optimization), which is $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} + \boldsymbol{\Delta\theta}$. That offset has $L^o_{minE}(\boldsymbol{\theta} + \boldsymbol{\Delta\theta})$ as

$$\begin{aligned} L_{minE}(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}) \leq L^o(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}) &= L_{min}(\boldsymbol{\theta}) + \sum_{u=1}^{U} \Delta\theta_u \cdot \left[b_u - \sum_{n=1}^{\overline{N}} b_{u,n} \right] \\ &= L_{min}(\boldsymbol{\theta}) + \boldsymbol{\Delta\theta}^* \cdot \Delta\boldsymbol{b} \quad , \end{aligned} \tag{5.326}$$

where $L_{minE}(\boldsymbol{\theta})$ is the value from (5.317). If $\Delta\boldsymbol{b} = 0$, then the best $\boldsymbol{\theta}$ is already known because the data-rate constraint is met. This $\boldsymbol{\theta}$ value equals the one used in $R_{\boldsymbol{xx}}$ optimization[75] in the immediately preceeding $R_{\boldsymbol{XX}}$ optimization, and the algorithm has found a vertex with the desired rate vector $\boldsymbol{b}_{min}$. The $\boldsymbol{\theta}$ optimization determines a $\boldsymbol{\Delta\theta}$ that maximizes $L^o_{minE}(\boldsymbol{\theta} + \boldsymbol{\Delta\theta})$, and then the algorithm returns with that new $\boldsymbol{\theta}$ to the weighted sum-rate optimization. Essentially, the components of $\Delta\boldsymbol{b}$, sign and magnitude, indicate desirable $\boldsymbol{\theta}$ values that a subsequent $R_{\boldsymbol{XX}}$ optimization can use to adjust $\Delta\boldsymbol{b}$, with any consequent change in order, to zero. (5.326) subtly hides that two equal (and thus successive) $\theta_i = \theta_{i+1}$ values (or more than 2) essentially permit a degree of freedom in adjacent users b_i values so that the optimization will only address the sum of user data rates for these two users. This is inevitable consequence of the original non-convexity of the weighted rate sum. However, such situations happily correspond to a set of solutions on the capacity–region's boundary that vertex-sharing will return to meet the constraints. All these points have the same energy sum, so the corresponding vertices' different data rates can be shared/averaged to the desired rate vector $\boldsymbol{b}$.

Ellipsoid Algorithm: The $\boldsymbol{\theta}$ update uses an ellipsoid-update algorithm to converge to the optimum $\boldsymbol{\theta}^o$ value. The ellipsoid method must initialize with an ellipsoid $O(\boldsymbol{\theta})$ that contains $\boldsymbol{\theta}^o$. The algorithm iteration index is k with estimate of $\boldsymbol{\theta}^o$, $\boldsymbol{\theta}_k$ is the center of an ellipsoid, $O_k(\boldsymbol{\theta})$. The algorithm recursively finds a smaller-volume ellipsoid, $O_{k+1}(\boldsymbol{\theta})$, at iteration $k+1$ as the smallest containing the intersection of O_k and the half-space $(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^* \cdot \Delta\boldsymbol{b} \geq 0$, because $\boldsymbol{\theta} - \boldsymbol{\theta}_k$'s positive projection on $\Delta\boldsymbol{b}$ means the surviving planar half space contains $\boldsymbol{\theta}$.

[74] Thus, good finite-precision software implementation should ensure this.

[75] If $\boldsymbol{\Delta\theta}^* \cdot \Delta\boldsymbol{b} < 0$, then $L_{minE}(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}) \leq L_{minE}(\boldsymbol{\theta})$ 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_{minE}.

Each ellipsoid has mathematical description:

$$O_k(\boldsymbol{\theta}) \triangleq \left\{\boldsymbol{\theta} \mid (\boldsymbol{\theta}-\boldsymbol{\theta}_k)^* \cdot A_k^{-1} \cdot (\boldsymbol{\theta}-\boldsymbol{\theta}_k) \leq 1\right\} \quad , \tag{5.327}$$

where positive-definite A_k (this A has no direct relation to Chapter 2's special square root discrete modulator A and is known to be the Hessian matrix for the ellipsoid) specifying the ellipsoid axes. A has eigenvectors $\boldsymbol{v}_{A,u}$. The matrix A's eigenvalues $\lambda_{A,u}$ specify the semi-axis lengths as $\sqrt{\lambda_{A,u}}$. Defining $\boldsymbol{\Delta\theta}_k \triangleq (\boldsymbol{\theta}-\boldsymbol{\theta}_k)$, the new ellipsoid is then

$$O_{k+1}(\boldsymbol{\theta}) = \left\{\boldsymbol{\theta} \mid O_k(\boldsymbol{\theta}) \bigcap \boldsymbol{\Delta\theta}_k^* \cdot \Delta\boldsymbol{b}_k \geq 0\right\} \quad . \tag{5.328}$$

When $U' > U$, equivalently at least one $L_{x,u} > 1$, the matrix $\widetilde{H}_u = F_u \cdot \Lambda_u \cdot M_u^*$ redefines as $\widetilde{H}_u \rightarrow \widetilde{H}_u \cdot M_u$ where user u has autocorrelation matrix $M_u \cdot \mathrm{Diag}(\boldsymbol{\mathcal{E}}_u) \cdot M_u^*$. While this is an additional MAC-input restriction, it does not reduce the MAC capacity region, nor preclude any $\boldsymbol{b}' \in \mathcal{C}(\boldsymbol{b})$ from realization.

The ellipsoid-normalized rate-difference sub-gradient vector is

$$\Delta\tilde{\boldsymbol{b}}_k \triangleq \frac{\Delta\boldsymbol{b}_k}{\sqrt{\Delta\boldsymbol{b}_k^* \cdot A_k \cdot \Delta\boldsymbol{b}_k}} \quad , \tag{5.329}$$

which simplifes (5.328) and (5.327) to produce a new ellipsoid that covers the intersection of the half plane and the old ellipsoid:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{1}{U+1} \cdot A_k \cdot \Delta\tilde{\boldsymbol{b}}_k \text{ new center} \tag{5.330}$$

$$A_{k+1} = \frac{U^2}{U^2-1}\left[A_k - \frac{2}{U+1} \cdot A_k \cdot \Delta\tilde{\boldsymbol{b}}_k \cdot \Delta\tilde{\boldsymbol{b}}_k^* \cdot A_k\right] \text{ new axes} \quad . \tag{5.331}$$

These steps may execute multiple times to ensure any $\boldsymbol{\theta} \succeq \mathbf{0}$, which it must be as a Lagrange multiplier. A $\theta_u < 0$ requires update until positive, which can occur by setting the corresponding $\Delta\tilde{b}_{u,k} = 1$ for the successive updates using (5.330) until $\theta_u > 0$. The first step in (5.330) is a steepest descent (with A as the inverse of the Hessian matrix A^{-1} weighting the direction by the ellipsoid axes' lengths together) with step-size $1/(U+1)$ that moves so that $\Delta\boldsymbol{\theta}_k$ moves into the half-plane where $\Delta\boldsymbol{\theta}_{k+1} \cdot \Delta\boldsymbol{b}_k \geq 0$. The second update in (5.331) follows through insertion of (5.330) into (5.327) and solving for A_{k+1} in terms of A_k, U, and $\Delta\boldsymbol{b}_k$. The new ellipsoid's volume decreases as

$$\mathrm{vol}\left(O_{k+1}\right) \leq e^{-\frac{1}{2U}} \cdot \mathrm{vol}\left(O_k\right) \quad , \tag{5.332}$$

implying exponential convergence.

The earlier mentioned equal successive (after reordering) $\delta = 0$ issue corresponds to an A_k matrix in (5.330) that is singular and for which $\Delta\tilde{\boldsymbol{b}}_k \neq \mathbf{0}$ but lies in A_k's null space. The ellipsoid effectively has two identical axes/dimensions that appear the same, so that any corresponding users' sum rate satisfies (stops) the update. A final step is necessary, recognizing that the two steepest descents (Hessian for autocorrelation matrix and ellipsoid for $\boldsymbol{\theta}$) have converged, but with $\boldsymbol{b} \neq \boldsymbol{b}_{min}$ but $\sum_u b_u = \sum_u b_{min,u}$. The user pairs corresponding to successive equal $\theta_u = \theta_{u-1}$ correspond to two vertices, basically with the corresponding $R_{\boldsymbol{xx}}(u) \leftrightarrow R_{\boldsymbol{xx}}(u-1)$. Since the converged algorithm outputs the best $\{R_{\boldsymbol{xx}}(u)\}$, the two corresponding data rate vectors follow. These will have the same $b_{i\neq u}$ nor $_{u-1}$ values for the other (non-equal-theta) users, but differ in that $b_{opt,u} \neq b_{min,u}$ and $b_{opt,u-1} \neq b_{min,u-1}$ but $b_{opt,u} + b_{opt.u-1} = b_{min,u} + b_{min,u-1}$. There is a fraction $\boldsymbol{f} = [f_u \; f_{u-1}]^t$ with $f_u + f_{u-1} = 1$ such that

$$\boldsymbol{b}_{min} = \underbrace{\begin{bmatrix} b_{v1,u} & b_{f2,u-1} \\ b_{v2,u} & b_{v2,u-1} \end{bmatrix}}_{B} \cdot \boldsymbol{f} \quad , \tag{5.333}$$

which has solution then

$$\boldsymbol{f} = B^{-1} \cdot \boldsymbol{b}_{min} \quad . \tag{5.334}$$

The special case $|B| = 0$, which means the two vertices are the same; so with $\theta_u = \theta_{u-1}$. Consequently, incremental change of $\boldsymbol{b}_{min}$'s corresponding elements of cause the same incremental energy offsets with $b_{min,u} = b_{min,u-1}$. In this special case the problem is already solved with the converged algorithm's output $\boldsymbol{b}$. (This special case may need exception handling in software implementations.)

Initialize Ellipsoid: To determine the initial ellipsoid that contains $\boldsymbol{\theta}^o$ in general, the algorithm may select a random order and a user-bit/symbol distribution b_i^o such that

$$b_{i\neq u} = b_i^o \tag{5.335}$$
$$b_u = b_u^o + 1 \quad . \tag{5.336}$$

Initialization makes a single-pass[76] through Chapter 4's fixed-margin iterative water-filling, for the selected order and bit distribution in (5.335) and (5.336), generates the bit distribution $\boldsymbol{b}^o$, thus generating a set of $\{R_{\boldsymbol{XX}}(u,n)\}$. This set of autocorrelation functions substitutes into the Lagrangian equation, which must always be non-negative for $\boldsymbol{\theta}^o \succeq 0$, so

$$0 \leq L_{min}(\boldsymbol{\theta}^o) \leq \left[\sum_{u=1}^{U}\sum_{n} w_u \cdot \text{trace}\left(R_{\boldsymbol{XX}}(u,n)\right)\right] + \left[\sum_{u=1}^{U}\left(b_u - \sum_{n} b_{u,n}\right)\right] \cdot \theta_u^o \tag{5.337}$$

must hold. Rearranging this equation using (5.336) leads to

$$0 \leq \theta_u^o \leq \sum_{u=1}^{U}\sum_{n} w_u \cdot \text{trace}\left(R_{\boldsymbol{XX}}(u,n)\right) = \theta_{u,max} \quad . \tag{5.338}$$

The step in (5.336) repeats for each user (that is incrementing each user successively by one bit as in (5.335) and (5.336)) to generate a $\theta_{u,max}$ for each, as in (5.338). 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 eigenvector-axised ellipsoid that covers this box has diagonal Hessian with squared lengths at least $\theta_{u,max}^2$ on each of these axes.

$$A_0^{-1} = \begin{bmatrix} \left(\frac{1}{\theta_{1,max}}\right)^2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \left(\frac{1}{\theta_{U,max}}\right)^2 \end{bmatrix} \quad . \tag{5.339}$$

The updates in (5.330) and (5.331) run until $\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k < \epsilon$.

Alternatively, the process' initialization with a first maximum-equal-weighted rate sum with $\boldsymbol{\theta} = \mathbf{1}$ ensures that the initial ellipsoid contains $\boldsymbol{\theta}^o$ because essentially any scaling by a positive constant only scales by the same factor the weighted rate sum[77], and thus an initial hypersphere of for instance $A = U \cdot I$ is sufficient.

5.4.4.4 Maximum weighted rate-sum programs

Matlab maxRMAC_cvx.m : The matlab program for weighted-sum-rate maximization, maxRMAC_cvx.m , uses CVX for a given $\boldsymbol{\theta}$ and input energy vector $\boldsymbol{\mathcal{E}}_{max}$. This first program listing accepts inputs with only $L_{x,u} = 1$ and thus runs faster. A vector version next appears after the scalar version. Yun Liao provided these two programs, based on a final non-CVX version that also appears, thanks to former student Dr. Mehdi Mohseni.

```
>> help maxRMAC_cvx
  function [Eun, w, bun] = maxRMAC_cvx(H, Eu, theta, cb)
  maxRMAC_cvx Maximizes weighted rate sum subject to energy constraint, each
  user has ONLY ONE transmit antenna. It uses CVX and mosek. It only works
  for Lxu=1 on all users.
    INPUTS:
      H(:,u,n): Ly x U x N channel matrix. Ly = number of receiver antennas,
      U = number of users and N = number of tones.
      If the channel is real-bbd (cb=2), maxPMAC_cvx realizes user data rates
      over all the tones (or equivalently positive and negative
```

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

[77] This is simpler than the case when initialization starts with minimum energy-sum minimization for a give $\boldsymbol{w}$ that leads to the initial fixed-margin water filling above

```
      freqs). This means the input H must be conjugate symmetric H_n =
      H_{N-n}^*.  The program will reduce N by 2 and focus energy on the
      lower half of frequencies. Thus N>1 must be even for real channels,
      with a special exception made for N=1.
      By constast if cb=1 (cplx bbd), maxRMAC_cvx need not have a conjugate
      symmetric H and N is not reduced.
  Eu: 1 x U Energy constraint for each user as total energy per symbol.
      For N=1 channel, the program adjusts energy to be per complex (cb=1)
      symbol at the beginning and then restores at end.  cb=2 has no
      change.  So for N=1, the input Eu should be u's energy/symbol.
  theta: weight on each user's rate, length-U vector.
  cb: =1 if H is complex baseband, and cb=2 if H is real baseband.
    OUTPUTS:
  Eun:    U x N energy distribution. Eun(u,n) is user u's energy allocation
          on tone n.
  w:      U x 1 Lagrangian multiplier w.r.t. energy constraints
  bun:    U by N bit distributions for all users.
```

Matlab maxRMACMIMO.m : The more general slower-running program is maxRMACMIMO.m permits variable number of antennas.

```
function [Rxxs, Eun, w, bun] = maxRMACMIMO(H, Lxu, Eu, theta , cb)
  maxRMAC_vector_cvx Maximize weighted rate sum subject to energy
  constaint. This uses cvx and mosek.
    INPUTS:
      H(:,u,n): Ly x U x N channel matrix. Ly = number of receiver antennas,
      U = number of users and N = number of tones.
      If the channel is real-bbd (cb=2), maxPMAC_cvx realizes user data rates
      over all the tones (or equivalently positive and negative
      freqs). This means the input H must be conjugate symmetric H_n =
      H_{N-n}^*.  The program will reduce N by 2 and focus energy on the
      lower half of frequencies. Thus N>1 must be even for real channels,
      with a special exception made for N=1.
      By constast if cb=1 (cplx bbd), maxRMAC_cvx need not have a conjugate
      symmetric H and N is not reduced.
  Eu: 1 x U Energy constraint for each user as total energy per symbol.
      For N=1 channel, the program adjusts energy to be per complex (cb=1)
      symbol at the beginning and then restores at end.  cb=2 has no
      change.  So for N=1, the input Eu should be u's energy/symbol.
  theta: weight on each user's rate, length-U vector.
  cb: =1 if H is complex baseband, and cb=2 if H is real baseband.
    OUTPUTS:
  Rxxs:  U-by-N cell array containing Rxx(u,n)'s if Lxu is a length-U
         vector; or Lxu-by-Lxu-by-U-by-N tensor if Lxu is a scalar.
  Eun:    U x N energy distribution. Eun(u,n) is user u's energy allocation
          on tone n.
  w:      U x 1 Lagrangian multiplier w.r.t. energy constraints
  bun:    U by N bit distributions for all users.
  bun: U by N bit distributions for all users.
```

Matlab maxRMAC.m : A non-CVX version is provided also, with the 3 called subroutines listed in Appendix G. This program will execute more quickly but also will produce solutions equivalent to, but not exactly the same as the CVX versions. (Basically, same best weighted sum rate, but with different user constituent rates)

```
>> help maxRMAC
 function [Eun, w, bun] = maxRMAC(H, Eu, theta, cb)
  maxRMAC Maximizes weighted rate sum subject to energy constraint, each
  user has ONLY ONE transmit antenna (Lxu=1). It does not use CVX.
    INPUTS:
      H(:,u,n): Ly x U x N channel matrix. Ly = number of receiver antennas,
      U = number of users and N = number of tones.
      If the channel is real-bbd (cb=2), maxPMAC_cvx realizes user data rates
      over all the tones (or equivalently positive and negative
      freqs). This means the input H must be conjugate symmetric H_n =
      H_{N-n}^*.  The program will reduce N by 2 and focus energy on the
      lower half of frequencies. Thus N>1 must be even for real channels,
      with a special exception made for N=1.
      By constast if cb=1 (cplx bbd), maxRMAC_cvx need not have a conjugate
      symmetric H and N is not reduced.
  Eu: 1 x U Energy constraint for each user as total energy per symbol.
      For N=1 channel, the program adjusts energy to be per complex (cb=1)
      symbol at the beginning and then restores at end.  cb=2 has no
      change.  So for N=1, the input Eu should be u's energy/symbol.
  theta: weight on each user's rate, length-U vector.
  cb: =1 if H is complex baseband, and cb=2 if H is real baseband.
```

```
  OUTPUTS:
Eun:    U x N energy distribution. Eun(u,n) is user u's energy allocation
        on tone n.
w:      U x 1 Lagrangian multiplier w.r.t. energy constraints
bun:    U by N bit distributions for all users.
Subroutines called
    startEllipseRate
    Lag_dual_f_rate
    minPtonerate
```

EXAMPLE 5.4.8 *[Maximum Rate Sum]* A check of maxRMAC_cvx on a familiar channel is

```
> H=zeros(1,2,1);
>> H(1,1,1)=80;
H(1,2,1)=60;
>> Eu=[1 1];
>> theta=[1 1];
>> [Eun, w, bun] = maxRMAC_cvx(H, Eu, theta)
Eun =
    1.0000
    1.0000
w =
    0.6394
    0.3597
bun =
    6.3220
    0.3219
>> sum(bun) =    6.6439
```

If this channel were complex baseband, then

```
>> [Eun, w, bun] = maxRMAC_cvx(H, Eu, theta, 1);
>> Eun' =    1.0000    1.0000
>> w'  =1.2789    0.7194
>> bun =   11.6443    0.6437
>> sum(bun) =   12.2880
```

and the data rates nearly double because this channel now has two real dimensions and thus spreading energy over more dimensions increases data rate as per Chapter 1's fair comparisons (here with asymptotically zero error probability with $\Gamma = 0$ dB). The Eu input is the energy/symbol for each user, or $\mathcal{E}_{\boldsymbol{x}}$. The program interprets energy for complex baseband for each user per symbol. When $\overline{N} = 1$, the program works for real channels also and the energy is also per (real) symbol. Otherwise for real channles, $\overline{N} = N \in 2\mathbb{Z}^+$.

Rerunning this with the vector CVX program trivally sets Lxu = 1 and produces the same result.

```
>> [Rxxs, Eun, w, bun] = maxRMACMIMO(H, [1 1], Eu, theta, 2)
Rxxs =
  2x1 cell array
    {[     1]}
    {[1.0000]}
Eun =
    1.0000
    1.0000
w =
    0.6394
    0.3597
bun =
    6.3220
    0.3219
>> sum(bun) =    6.6439
```

However, running the more direct non-CVX version produces the same rate sum

```
>> [Eun, w, bun] = maxRMAC(H, Eu', theta', 2)
Eun =
    1.0000
    1.0000
w =
    4.8193
```

```
    0.1800
bun =
    6.3220
    0.3219
>> sum(bun) =    6.6439
```

This program often runs faster than CVX but can deviate slightly on more complicated channels.

multitone extensions: Accepting channel inputs in frequency domain ($\overline{N} \geq 2$ requires some understanding of these programs' use as the next example illustrates.

EXAMPLE 5.4.9 *[Multitone Examples]* Repeating this channel's gains on each of 4 tones, and providing total energy per user thus 4x larger, depends on whether the channel is to be interpreted as complex or real (complex baseband channels again can have all real transfer values).

```
> H4
H4(:,:,1) =    80    60
H4(:,:,2) =    80    60
H4(:,:,3) =    80    60
H4(:,:,4) =    80    60
[Eun, w, bun] = maxRMAC_cvx(H4, 4*Eu', theta', 1)
Eun =
    1.0000    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000    1.0000
w =
    0.6396
    0.3598
bun =
   12.6441   12.6441   12.6441   12.6441
    0.6438    0.6438    0.6438    0.6438 ,
```

which is the complex result repeated 4 times as expected. However, for real baseband interpretation

```
>> [Eun, w, bun] = maxRMAC_cvx(H4, 2*Eu', theta', 2)
Eun =
    1.0000    1.0000
    1.0000    1.0000
w =
    0.6396
    0.3598
bun =
    6.3220    6.3220
    0.3219    0.3219
>> sum(bun,'all') =   13.2879
```

The channel is Hermitian symmetric, which it must be for real baseband channels as an input. There are now only two tones that carry (non-redundant) information, and so the energy/symbol was only doubled. Tacitly, there are another 2 negative-frequency tones carrying the same energy and data rate, but not output because the channel is real baseband. The data rates remain the same as the real-baseband case.

As a check on the 2-tone real baseband example, SWF should provide the same results if correctly used. SWF accepts input energy per sample, so the energy vector remains the same for the two users. The channel H4 should be reduced to just its first two tones. Both Lxu=1. Finally the noise is while on each of the two dimensions (SWF does noise whitening internally).

```
[Rxx, bsum , bsum_lin] = SWF([1 1], H4(:,:,1:2), [1 1], ones(1,1,2), 2)
bsum =   13.2879
Rxx(:,:,1) =
     1     0
     0     1
Rxx(:,:,2) =
     1     0
     0     1
bsum =   13.2879
bsum_lin =    2.1174
```

The rate sum is the same, which it must be. Each of the two tones has one unit of energy on each dimension. For this channel, the DC and Nyquist tones are combined and have the same gain. In many practical cases, both DC and Nyquist have zero gain and this is irrelevant. For a real baseband channel that does not have this symmetry, cb can be set to 1 and the conjugate symmetric channel can entered into SWF or maxRMAC's programs essentially both positive and negative frequencies. The input energy constraint should be doubled. The output user bit distribution should be divided by 2. The Nyquist frequency will carry all those 1/2 bits on on its inphase component, while the DC will carry it's one-half bits as PAM.

For a channel with different gains - necessarily implying here complex baseband because the channel does not have Hermitian symmetry (or divide ultimate data rates by 2 after doubling desired energy and repeating the channel to double its dimensionality):

```
>> H4x=zeros(1,2,4);
>> H4x(:,:,1)= [80 60];
>> H4x(:,:,2)= [40 30];
>> H4x(:,:,3)= [50 50];
>> H4x(:,:,4)= [30 40];
>> [Eun, w, bun] = maxRMAC_cvx(H4x, 4*Eu, theta,1)
Eun =
    1.9984    1.9978    0.0037    0.0000
    0.0000    0.0000    1.9980    2.0020
w =
    0.5004
    0.4995
bun =
   13.6428   11.6427    3.3711    0.0023
    0.0000    0.0000    8.9181   11.6435
>> sum(bun'*theta') =   49.2206
```

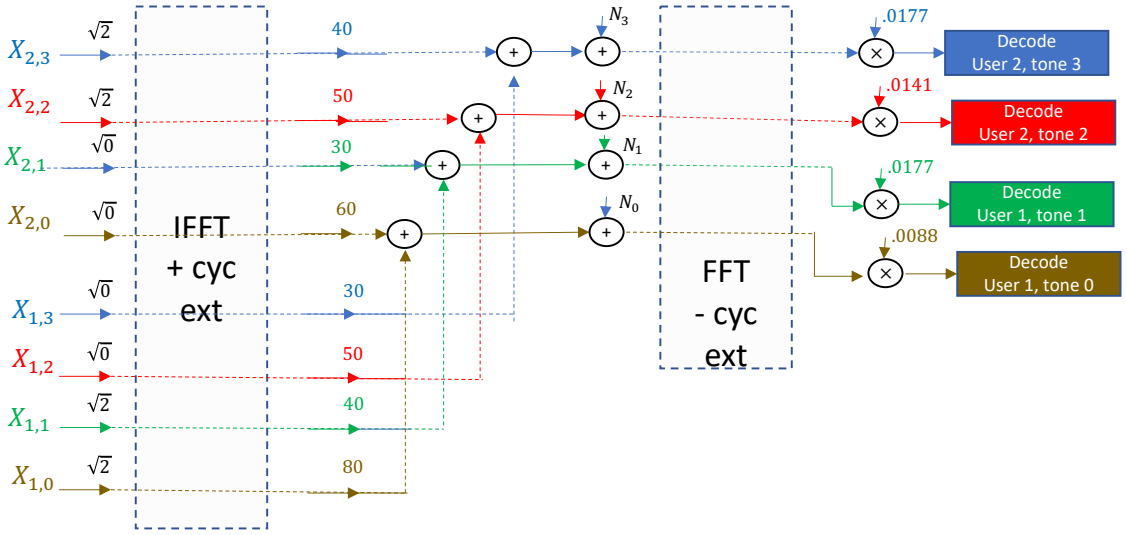


Figure 5.36: Design Example for H4x.

The corresponding MAC design uses the mu_mac.m program on each of the user tones, here shown for $n = 3, ..., 0$ to match Figure 5.36

```
[b, GU, WU, SO, MSWMFU] = mu_mac([30 40], [0 0 ; 0 sqrt(2)], [1 1] , 1);
b = 0   11.6443
GU =   1     0
             0     1
MSWMFU =0.0236
```

```
       0
0.0177
[b, GU, WU, S0, MSWMFU] = mu_mac([50 50], [sqrt(2) 0 ; 0 0], [1 1] , 2);
b =    0   12.2880
GU =   1     0
             0     1
MSWMFU =0.0236
       0
 0.0141
 [b, GU, WU, S0, MSWMFU] = mu_mac([40 63], [sqrt(2) 0 ; 0 0], [1 1] , 2);
b =    11.6443         0
GU =   1     0
             0     1
MSWMFU =0.0236
    0.0177
         0
[b, GU, WU, S0, MSWMFU] = mu_mac([80 60], [sqrt(2) 0 ; 0 0], [1 1] , 2);
b =    6.8220         0
GU =   1     0
             0     1
MSWMFU =
    0.0088
         0


>> [Rxx, bsum , bsum_lin] = SWF(Eu, H4x(:,:,1:4), [1 1], ones(1,1,4), 1)
Rxx(:,:,1) =
    2.0001         0
         0         0
Rxx(:,:,2) =
    1.9996         0
         0         0
Rxx(:,:,3) =
    0.0003         0
         0    2.0000
Rxx(:,:,4) =
         0         0
         0    2.0000
bsum =   49.2206
bsum_lin =   48.4228
```

The SWF solution appears largely FDM on these 4 tones for maximum rate sum, while the maxRMAC is also close to this, with presumably Tone 2 (from right) in maxRMAC_cvx apparently seeing some internal CVX finite-precision issue. Note that with this channel (unlike the channel that has 80 60] on all n), the FDM solution allows the linear solution to approximate well the nonlinear solution.

All 3 have the same rate sum to 6-digit accuracy; however individual user rates vary slightly. Tone 2 appears sensitive to errors. The SWF program uses no CVX and specifically designed for this application, and probably most trustworthy when it can be used. The weight values are nearly equal, which is a sign that the system borders singularity and that either user's energy increments equally affect the sum, allowing equal and opposite offsets to zero in altering the individual energy contributions.

An unequal theta version (which only can use maxRMAC programs and not SWF):

```
>> theta = [    10     1  ];
>> [Eun, w, bun] = maxRMAC_cvx(H4x, [1 1], theta ,1)
Eun =
    0.2563    0.2560    0.2467    0.2410
    0.1173    0.1177    0.3266    0.4384
w =
   38.1615
    1.7429
bun =
   10.6807    8.6818    9.2708    7.7674
    0.3303    0.3310    1.2152    2.0772
>> sum(bun') =   36.4006    3.9537
>> sum(bun,'all') =   40.3543
```

User 2 (top) is heavily favored. The rate sum reduces, but the maximum weighted rate sum depends of course on $\boldsymbol{\theta}$ and its scaling, but by letting both $\boldsymbol{\theta}^* \cdot \mathbf{1}$ be equal, the weighted sum is very large because it favors one user, which the bit distribution also reflects. The

energy weights also indicate the user 2 preference. Reversing the weighting to favor user 1 also reduces rate sum. Thus, the weighted rate sum may be optimized, but the (unweighted) rate sum reduces.

```
>> theta=[1 5]';
>> [Eun, w, bun] = maxRMAC_cvx(H, [1 1]', theta,1)
Eun =
    0.2371    0.2363    0.2499    0.2766
    0.3691    0.3689    0.2523    0.0096
w =
    1.9903
   17.9654
bun =
    1.9123    1.9115    1.0057    0.0279
    9.7392    7.7395    9.2897    8.7929
>> sum(bun') =    4.8575   35.5613
>> sum(bun,'all') =   40.4188
```

MIMO Example(s) The maxRMAC program permits variable $L_{x,u} \geq 1$, as the following example illustrates:

EXAMPLE 5.4.10 *[Variable $L_{x,u}$ weighted rate maximization]* The noise-whitened 2×4 channel H2 has two users, each with $L_{x,u} = 2$, each with energy of 5 and 6 units per complex symbol respectively. The following run illustrates that each user has a 2×2 input autocorrelation matrix $R_{\boldsymbol{xx}}(u)$ that maximizes the corresponding rate sum:

```
>> H2 =
     4     3     2     1
     5     6     7     8
[Rxxs, Eun, w, bun] = maxRMACMIMO(H2, [2 2], [5 6], [1 1] , 1)
Rxxs = 2x1 cell array
Eun =
    5.0000
    6.0000
W' =    0.3881    0.3275
bun =
    7.6484
    5.8337
>> Rxxs{:,:}
=    3.5983    2.2458
       2.2458    1.4017
=    1.6863    2.6971
       2.6971    4.3137
```

The $R_{\boldsymbol{xx}}(u)$ are not diagonal, but their traces match the specified energies. The two users have the data rates 7.6 and 5.8 respectively.

If user 1 had instead only 1 antenna, then the corresponding use of maxRMACMIMO is

```
[Rxxs, Eun, w, bun] = maxRMACMIMO(H2(:,1:3), [2 1], [5 6], [1 1] , 1)
Rxxs =  2x1 cell array
Eun =
    5.0000
    6.0000
W' =    0.3811    0.3144
bun =
    7.5867
    4.1392
>> Rxxs{:,:}
 =
    3.9149    2.0611
    2.0611    1.0851
 =
    6.0000
```

In this case, user 1 has only a scalar $R_{\boldsymbol{xx}}(1) = 6$, while user 2 has a 2×2 autocorrelation matrix, which is not equal to to the case where H2 is 2×4 even though user 1's energy is the same and user 2's channel submatrix is the same. The extra antenna for user 1 actually improves both users' data rates because the extra antenna reduces uncanceled interference between the users.

Matlab maxRESMAC_cvx.m : This program is similar or maxRmac_cvx , but the "ES" is energy sum. This program is very useful in Section 5.5's duality where the BC dual channel will naturally have a single transmitter's energy constraint.

```
function [E, w, b] = maxRMAC(H, Eu, theta, cb)
maxRMAC Maximizes weighted rate sum subject to energy constraint, each
user has ONLY ONE transmit antenna (Lxu=1). It does not use CVX.
  INPUTS:
H(:,u,n): Ly x U x N channel matrix. Ly = number of receiver antennas,
    U = number of users and N = number of tones.
    If the channel is real-bbd (cb=2), maxPMAC_cvx realizes user data rates
    over all the tones (or equivalently positive and negative
    freqs). This means the input H must be conjugate symmetric H_n =
    H_{N-n}^*.  The program will reduce N by 2 and focus energy on the
    lower half of frequencies. Thus N>1 must be even for real channels.
    For real N=1 channel, the program runs and will produce half the data
    rates for the same complex N=1 channel.
    By constast if cb=1 (cplx bbd), maxRMAC_cvx need not have a conjugate
    symmetric H and N is not reduced.
Eu: 1 x U Energy constraint for each user as total energy per symbol.
theta: weight on each user's rate, length-U vector.
cb: =1 if H is complex baseband, and cb=2 if H is real baseband.
  OUTPUTS:
E:    U x N energy distribution. Eun(u,n) is user u's energy allocation
        on tone n.
w:   U x 1 Lagrangian multiplier w.r.t. energy constraints
b: U by N bit distributions for all users.
Subroutines called
    startEllipseRate
    Lag_dual_f_rate
    minPtonerate
```

This routine calls 3 subroutines startEllipseRate, Lag_dual_f_rate , and minPtonerate that all appear in Appendix G.

There is also a CVX-using version with variable number of antennas with call

```
function [Rxxs, Eun, w, bun] = maxRMACMIMO(H, Lxu, Eu, theta , cb)
 See Appendix G.
```

maxRMACMIMO a minPtoneMIMO routine

> *[Energy-Sum-Based weighted-rate-sum-maximization example]* Returnring to this Examples 4-tone complex channel, the energy-sum-only constraint relaxes energies so the equal-rated rate-sum should (and does slightly beyond precisional display) increase:
>
> ```
> Eun, w, bun] = maxRESMAC_cvx(H, 8, [ 1 1]',1)
> Eun =
>     2.0000    1.9996   -0.0000    0.0000
>     0.0000    0.0000    2.0007    1.9997
> w =    0.4998    0.4998
> bun =
>    13.6439   11.6440   -0.0000    0.0005
>     0.0000    0.0000   12.2885   11.6436
> >> sum(bun,'all') =   49.2206
> >> sum(Eun,'all') =    8.0000
> >> sum(bun') = 25.2884   23.9322
> ```
>
> The sum data rate increases as expected, by almost 1 bit per symbol. The solution is exactly FDM in nature with each user using its best two tones. There should only be one weight vector element, and this remains a not yet understood (by this author) quirk of CVX in that there should be only one element. However, it is the same value so likely a CVX issue related to the energy having two user components that are jointly optimized.

5.4.4.5 Minimum weighted energy-sum programs

Similarly, designers may target a minimum (possibly weighted) energy sum for a given $\boldsymbol{b}$. The set of all such weighted energy sums for $\boldsymbol{w} \succeq \mathbf{0}$ traces the boundary $\mathcal{C}_{\boldsymbol{b}}(\boldsymbol{\mathcal{E}})$.

Matlab minPMAC.m : This program and its 6 called subroutines were intially written by former student Dr. M. Mohseni. They avoid the use of CVX and run quickly, but force $L_{x,u} = 1$. They were modified by the author substantially to handle equal-theta issues that the original programs ignored. This program program will set a FEAS_FLAG=2 when there are equal-theta users. Vertex-sharing ratios for each cluster of equal-theta users are also provided, as entries in a table info. Simple use with full-rank channels can ignore the last 3 outputs; these have additional information only when the MAC is degraded and must consequently use vertex sharing. The called subroutines are in Appendix G.

```
 function [Eun, theta, bun, FEAS_FLAG, bu_a, info] = minPMAC(H, bu, w, cb)
This main function contains ellipsoid method part and calls direclty or
  indirectly 6 other functions.  minPMAC uses no CVX.  Another routine
  minPMAC_cvx uses cvx and usually runs longer.  However, these scalar
  minPMAC programs assume each user has Lxu = 1;
  There is a CVX-using program minPMACMIMO that allows variable Lxu.
INPUTS:
H(:,u,n): Ly x U x N channel matrix. Ly = number of receiver antennas,
    U = number of users and N = number of tones.
    If the channel is real-bbd (cb=2), minPMAC realizes user data rates
    over the lower half of the tones (or equivalently, the positive
    freqs), and directly corresponds to the input bu user-data rate vector.
    N is the number of tones actually transmitted (so corresponds to a 2N
    size FFT when cb=2).
    By constast if cb=1 (cplx bbd), minPMAC realizes user data rates
    over all tones, but uses the same core optmization, so the data rate
    is halved internal to the program, realizing that half bu rate on the
    lower tones, because the designer (cb=1) wants it over all tones.
    N is the number of tones actually transmitted (so corresponds to a N
    size full-complex FFT when cb=1).
bu_min: U x 1 vector containing the target rates for all the users.
w:  U x 1 vector containing the weights for each user's power.
cb: =1 if H is complex baseband, and cb=2 if H is real baseband.
OUTPUTS:
Eun:    U by N energy distribution that minimizes the weighted-sum energy.
      E(u,n) is user u's energy allocation on tone n.
theta:the optimal U by 1 dual variable vector containing optimal weights
    of rates. Theta determines the decoding order. Largest theta is
    decoded last, and smallest first.
bun,   U by N bit distributions for all users.
FEAS_FLAG: indicator of achievability.
          FEAS_FLAG=1 if the target is achieved by a single ordering;
          FEAS_FLAG=2 if the target is achieved by time-sharing
bu_a: U-by-1 vector showing achieved sum rate of each user.
info: various length output depending on FEAS_FLAG
          --if FEAS_FLAG=1: 1 x 4 cell array containing
              {Rxxs, Eun, bun, theta} corresponds to the single vertex
              there are no equal-theta user sets in this case
          --if FEAS_FLAG=2: 1-x 6 cell array, with each row representing
              a time-shared vertex {Rxxs, Eun, bun, theta, frac}
              there are numclus equal-theta user sets in this case.
 info's row entries in detail (one row for each vertex shared
      - Eun:  U-by-N matrix showing users' transmit energy on each tone.
          If infeasible, output 0.
      - bun: U-by-N matrix showing users' rate on each tone. If
          infeasible, output 0.
      - theta: U-by-1 Lagrangian multiplier w.r.t. target rates
      - order: produces the order from left(best) to right for vertex
      - frac: fraction of dimensions for each vertex in time share (FF =2
          ONLY)
      - cluster: index (to which cluster the user belongs; 0 means no
          cluster)
Subroutines called directly are
      startEllipse.m
      Lag_dual_f.m
  and indirectly
      minPtone.m
      Hessian.m
      eval_f.m
      fmwaterfill_gn.m
```

Matlab minPMACMIMO.m : This version allows variable $L_{x,u} \geq 1$ and CVX. It modifies two of the subroutines above. These also appear in Appendix G.

```
function [FEAS_FLAG, bu_a, info] = minPMACMIMO(H, Lxu, bu_min, w, cb)
INPUTS:
 H(:,u,n): Ly x U x N channel matrix. Ly = number of receiver antennas,
           U = number of users and N = number of tones.
```

```
    If the channel is real-bbd (cb=2), minPMAC realizes user data rates
    over the lower half of the tones (or equivalently, the positive
    freqs), and directly corresponds to the input bu_min user-data rate vector.
    By constast if cb=1 (cplx bbd), minPMAC realizes user data rates over
    all tones, but uses the same real-bbd core optmization that doubles the
    number of tones with an equivalent-gain set, so this program halves
    the data rate internally and realizes that half bu_min rate on the
    lower tones, which is the input set for which results are reported.
Lxu:  1 x 1 or 1 x U vector containing each users' number of
      transmit antennas. If Lxu is 1 x 1, each user has Lxu antennas.
bu_min: U x 1 vector containing the target rates for all the users.
w:  U x 1 vector containing the weights for each user's energy.
cb: =1 if H is complex baseband, and cb=2 if H is real baseband.
  Outputs:
      - FEAS_FLAG: indicator of achievability.
          FEAS_FLAG=1 if the target is achieved by a single ordering;
          FEAS_FLAG=2 if the target is achieved by time-sharing
      - bu_a: U-by-1 vector showing achieved sum rate of each user.
      - info: various length output depending on FEAS_FLAG
          --if FEAS_FLAG=1: 1 x 4 cell array containing
              {Rxxs, Eun, bun, theta} corresponds to the single vertex
              there are no equal-theta user sets in this case
          --if FEAS_FLAG=2: 1-x 6 cell array, with each row representing
              a time-shared vertex {Rxxs, Eun, bun, theta, frac}
              there are numclus equal-theta user sets in this case.
     info's row entries in detail (one row for each vertex shared
      - Rxxs: U-by-N cell array containing Rxx(u,n)'s if Lxu is a
          length-U vector; or Lxu-by-Lxu-by-U-by-N tensor if Lxu is a
          scalar. If the rate target is infeasible, output 0.
      - Eun:  U-by-N matrix showing users' transmit energy on each tone.
          If infeasible, output 0.
      - bun: U-by-N matrix showing users' rate on each tone. If
          infeasible, output 0.
      - theta: U-by-1 Lagrangian multiplier w.r.t. target rates
      - order: produces the order from left(best) to right for vertex
      - frac: fraction of dimensions for each vertex in time share (FF =2
          ONLY)
      - cluster: index (to which cluster the user belongs; 0 means no
          cluster)
Functions called are
      startEllipse_var_Lxu
      minPtoneMIMO
```

EXAMPLE 5.4.11 [simple scalar MAC with single and multiple tones] This first example returns to Example 2.7.1's simple scalar MAC with gains $h_2 = 0.8$ and $h_1 = 0.6$. For this channel the number of tones can be set as $\bar{N} = 1$, while $U = 2$ and $L_y = 1$. The input energies were $\mathcal{E}_1 = \mathcal{E}_2 = 1$, but not of use in the minPMAC software. This example attempts $b_1 = b_2 = 3$ bits/dimension for each user. 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];
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H, b', w', 2);
Eun =
    0.6300
    0.0175
theta =
    1.2800
    1.2956
bun =
    3.0000
    3.0000
FEAS_FLAG =     1
bu_a =
    3.0000
    3.0000
info =  1x4 table
     bu_v         Eun             bun             theta
    ______    ____________    ____________    ____________
    3    3    {2x1 double}    {2x1 double}    {2x1 double}
```

The $\boldsymbol{\theta}$ elements are not equal, so this simple channel's minimum energy sum occurs at a vertex (as previously known). The smaller $\theta_2 < \theta_1$ implies[78] user 2 is first in MAC decoding order. The reader can verify that reversing the 80 and 60 causes all the values above to reverse and then the MAC receiver decodes user 2 last. The data rates are equal to $\boldsymbol{b}$. The minimum energy is $\mathcal{E}_x = .6475$. Again, progressing to 4 tones (really 2 if viewed as real baseband with cb=2):

```
>> H=zeros(1,2,4);
>> H(1,1,:)=80 ;
>> H(1,2,:)=60 ;
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H, 4*b', w', 2)
theta =
    1.2800
    1.2956
Eun =
    0.6300    0.6300    0.6300    0.6300
    0.0175    0.0175    0.0175    0.0175
bun =
    3.0000    3.0000    3.0000    3.0000
    3.0000    3.0000    3.0000    3.0000
FEAS_FLAG =     1
bu_a =
   12.0001
   12.0000
```

The CVX version minPMACMIMO produces an identical result, but runs an order of magnitude slower. Increasing $\boldsymbol{b}$ at the same bandwidth eventually causes the energy limit to be exceeded because minPMAC only addresses minimum energy sum for the given data rate:

```
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H, 5*b', w', 2)
theta =
   10.2400
   10.2840
Eun =
    5.0917    5.0917    5.0917    5.0917
    0.0500    0.0500    0.0500    0.0500
theta =
   10.2400
   10.2840
bun =
    3.7500    3.7500    3.7500    3.7500
    3.7500    3.7500    3.7500    3.7500
FEAS_FLAG =     1
bu_a =
   15.0000
   15.0000
```

In this case, the attempted data rate is larger, so the minPMAC output energies would violate an energy constraint of 1 unit/per user.

MAC Receiver Equal-Theta Sharing: Generally speaking when there are equal thetas, the MAC receiver must alternate order according to the dimensional-use fraction (info.frac in Example 5.4.11 above). This is an example of subuser components. The users' $R_{\boldsymbol{xx}}(u)$ do not change. In practice with good codes ($\Gamma = 0$ dB).

Continguing the same example with $\overline{N} = 4$ obtains (note that the rate vector is multiplied by 4 since there are now 12 bits over a subsymbol that is 4 times longer for each user):

[Example 5.4.11 continued]

Equal channel gains instead creates a singular channel condition where the two theta values are equal:

```
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC([80 80], b', w', 2)
Eun =
    0.3200
    0.3199
```

[78] Recalling that Matlab reverses this text's MAC-indexing convention.

```
theta =
    1.2800
    1.2800
bun =
    0.4997
    5.5003
FEAS_FLAG =     2
bu_a =
    3.0000
    3.0000


info = 2x6 table
           bu_v               Eun             bun            theta         frac     clusterID
    ___________________    ____________    ____________    ____________    _____    _________
    0.49971     5.5003     {1x2 double}    {1x2 double}    {1x2 double}    0.49998      1
     5.5003    0.49971     {1x2 double}    {1x2 double}    {1x2 double}    0.50002      1
 >> info.bu_v'*info.frac =
    3.0000
    3.0000   (vertex sharing works within the cluster)
```

In this $\theta_1 = \theta_2$ case, vertex-sharing is necessary and the info table lists the two vertices and the fraction of time they are shared. The energy sum is minimum and the same for both vertices. The 50/50 vertex share achieves the target $b_1 = b_2 = 3$ target value, other target rates might provide different vertex-sharing fractions, as here:

```
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC([80 80], [4 2], w', 2)
Eun =
    0.3201
    0.3197
theta =
    1.2800
    1.2800
bun =
    0.5002
    5.4998
FEAS_FLAG =     2
bu_a =
    4.0000
    2.0000


info =  2x6 table
           bu_v               Eun             bun            theta          frac      clusterID
    ___________________    ____________    ____________    ____________    _______    _________
     5.5006    0.49943     {1x2 double}    {1x2 double}    {1x2 double}    0.69991        1
    0.50022     5.4998     {1x2 double}    {1x2 double}    {1x2 double}    0.30009        1
```

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," but thus the design may not satisfy an energy-vector constraint. The next Subsection's admMAC program addresses solution admissibility when there is an energy constraint. When $U > \varrho_{\widetilde{H}}$ with minPMAC, there will always be equal-theta users when $\overline{N} = 1$. However, as $\overline{N} >$ increases, the situation may depend on the channel and if the design over tones can emulate the equivalent vertex-sharing by varying each user's energy over n.

[illustration of variable $\overline{N}$] A 2×3 channel (with $\overline{N} = 1$) must have the equal-theta situation because $\varrho_H < U$:

```
>> Htemp = [
    80    40    30
    30   -50   -15 ];
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(Htemp, [7 5 6], [1 1 1], 1)
Elapsed time is 0.823752 seconds.
Eun =
    0.0560
    0.0891
    0.0967
theta =
    0.2892
    0.1968
```

```
    0.1968
bun =
    6.0000
    3.4106
    8.5895
FEAS_FLAG =     2
bu_a =
    7.0000
    5.0000
    6.0000


info =  2x7 table
               bu_v                  Eun             bun             theta           order          frac      clusterID
    ____________________________    ____________    ____________    ____________    ____________    _______    _________
    5.8732    6.1269         6    {1x3 double}    {1x3 double}    {1x3 double}    {1x3 double}    0.58514        1
    8.5895    3.4106         6    {1x3 double}    {1x3 double}    {1x3 double}    {1x3 double}    0.41485        1
>> info.bu_v'*info.frac =
    7.0000
    5.0000
    5.9999
    >> info.order{:} =
     1     2     3
     2     1     3
```

Example extension to $\overline{N} > 1$ may or may not exhibit vertex-sharing over the different tones. The H4 channel below is complex baseband and a first attempt is at a fairly high data rate (number of bits/dimension). Because this channel has no conjugate symmetry with n, it must be complex baseband, which is why cb=1:

```
H4=zeros(2,3,4);
>> H4(:,:,1) =[80    40    30
    30   -50   -15
 ];
>> H4(:,:,2) =[100    30    75
   -10    80    15 ];
>> H4(:,:,3) =[10    40   -60
    30    70   -10 ];
>> H4(:,:,4) =[20   -30  -100
    50    60    70];
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H4, 4*[7 5 6], [1 1 1], 1)
Elapsed time is 3.616063 seconds.
Eun =
    0.0914    0.0886    0.0044    0.0913
    0.0038    0.0068    0.0911    0.0915
    0.0863    0.0866    0.0861    0.0000
theta =
    0.1832
    0.1832
    0.1744
bun =
    8.7054    6.6123    2.1415    7.9765
    2.4049    5.3436    8.4013   10.4148
    6.6106    7.1161    6.2731    0.0000
FEAS_FLAG =
     2
bu_a =
   28.0000
   19.9998
   24.0000


info =2x7 table
               bu_v                  Eun             bun             theta           order          frac      clusterID
    ____________________________    ____________    ____________    ____________    ____________    _______    _________
    29.668    20    22.332    {1x3x4 double}    {1x3x4 double}    {1x3 double}    {1x3 double}    0.60595        1
    25.436    20    26.564    {1x3x4 double}    {1x3x4 double}    {1x3 double}    {1x3 double}    0.39405        1
 >> info.bu_v'*info.frac
ans =
   28.0000
   19.9997
   24.0000
>>  info.order{:}  =
     2     3     1
     2     1     3
```

This first 4-tone example retains the equal-theta vertex sharing. However, when running this same channel with a lower data rate, which essentially encourages independent use of available dimensions, the vertex-sharing does not occur (see FEAS_FLAG =1), and evident with matlab's long-format display of more significant digits.

```
 >> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H4,[4 5 3], [1 1 1], 1)
>> theta =
    0.0009
    0.0011
    0.0009
Elapsed time is 1.780554 seconds.
Eun =
   1.0e-03 *
    0.3003    0.3406    0.0001    0.0798
    0.2921    0.4070    0.3830    0.0114
    0.0001    0.0002    0.0814    0.3887
theta =
    0.0009
    0.0011
    0.0009
bun =
    1.6214    2.0935    0.0001    0.2848
    1.1360    1.9895    1.8031    0.0719
    0.0001    0.0016    0.2814    2.7169
FEAS_FLAG =     1
bu_a =
    3.9998
    5.0004
    3.0000
info =
  1x4 table
                bu_v                    Eun             bun             theta
    ___________________________    ____________    ____________    ____________
    3.9998    5.0004         3    {3x4 double}    {3x4 double}    {3x1 double}
>> format long
>> info.theta{:} =
   0.000889504590608
   0.001099533276759
   0.000917771187035
```

With different gains, the energy/bit distributions correspondingly change, and reduces/zeros some users' energies on some dimensions. The theta values are (slightly) different, so the long-form matlab format was used to illustrate.

Figure 5.37 illustrates the use of clusters with colors for different clusters shown (instead of inserting the commands directly here from matlab):

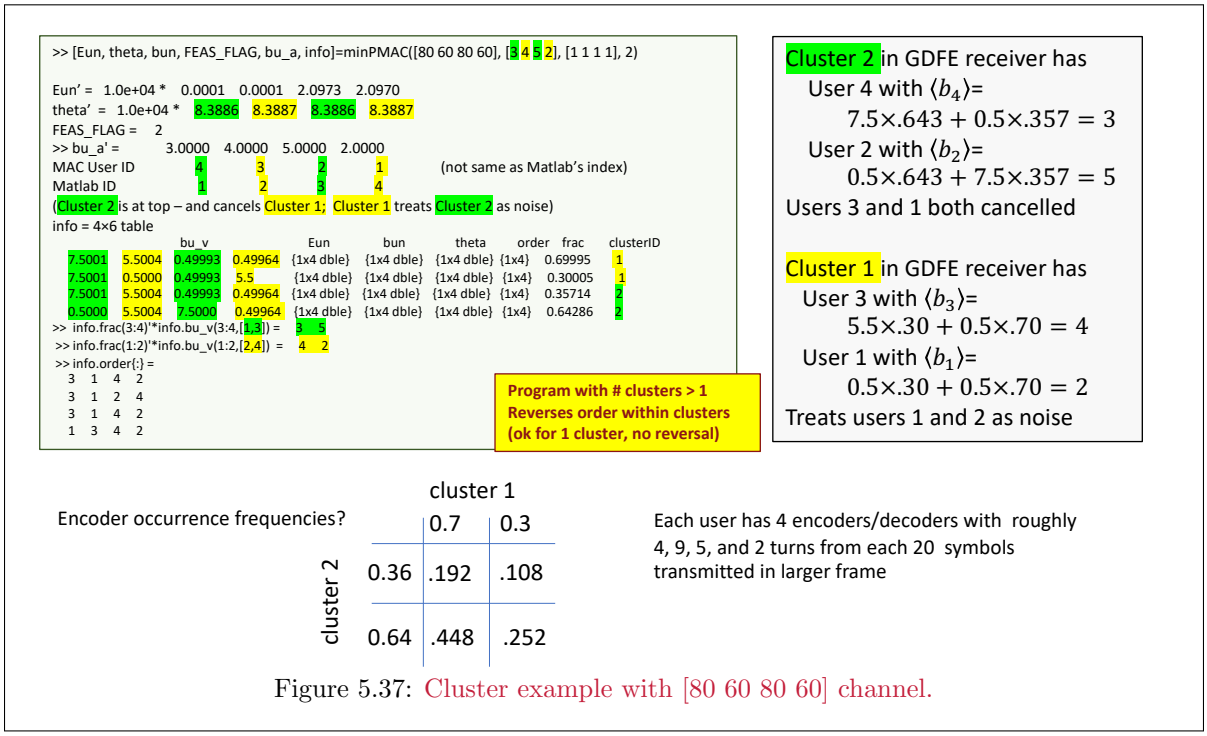


Figure 5.37: Cluster example with [80 60 80 60] channel.

Finally the 3-user 64-tone channel is revisted, and despite $\overline{N} = 64$, this channel does have equal-theta for 2 of the 3 users (H repeated from Example 5.4.6).

```
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H, [445 412 132]', [1 1 1]',2);
Elapsed time is 215.045054 seconds.
>> theta =
   547.1836
   547.1836
   538.4958
FEAS_FLAG =     2
bu_a =
  445.0000
  412.0000
  132.0000
```

% bu_v			Eun	bun	theta	frac	clusterID
445.81	411.19	132	{1x3x64 double}	{1x3x64 double}	{1x3 double}	0.95968	1
425.68	431.32	132	{1x3x64 double}	{1x3x64 double}	{1x3 double}	0.040321	1

The minimum energy sum in this case, even with 64-tones and different energy and bit distributions over these tones, favors additionally time-sharing. However, one vertex is heavily favored with the other at roughly 4%. This small fraction complicates the attempt to "vertex share over the tones" for this channel.

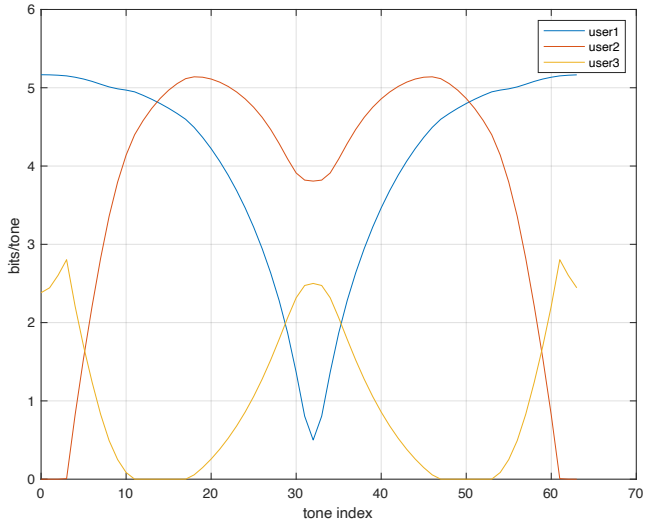


Figure 5.38: Bit Distributions for Example 5.4.1.

Example 5.4.6 shows that a single feedback section is non-trivial on this channel (Matlab's tone 22 corresponds to $n = 23$). Figure 5.38 shows the corresponding bit distributions. These are for a complex channel, and minPMAC works in bits/real-dimension. Thus, the design uses $\boldsymbol{b} = \frac{1}{2} \cdot [520 \quad 480 \quad 130]$, which is clearly admissible because each user is less than the simultaneous water-filling bits/user elements.

A design that minimizes the 3 users' energy sum follows through minPmac. The program minPMAC's bu_min input is in bits/user/real-dimension; thus the design below includes the factor $1/2 \cdot \log_2$.

The output bit distribution is similarly in bits/real-dimension. So the input bit-rate target $\boldsymbol{b}$ is scaled by 1/2 if the channel is baseband complex and the output bits/dimension can be doubled for complex baseband. The energy outputs are also in bits/real-dimension, so for instance if a complex baseband input channel has $\overline{N}$ tones, then to get the average energy/dimension, a users' total energy can be added over the tones (the minPMAC output E distribution) and then divided by $N = 2\overline{N}$; however the division would be by $N = \overline{N}$ in the case of a real baseband channel. This example is complex, and so the following steps illustrate the correct use of minPMAC when $\overline{N} = 64$:

```
bu_min=0.5*[440 410 130]'; \% slightly < flat-energy earlier,  times 1/2
w=[1 ; 1 ; 1];
Lxu=1;
>> [E, theta, b] = minPMAC(H, bu_min, w);
>> sum(b') = 219.9070  204.9593   64.9617
>> sum(E') =   1.4910    1.2218    0.4903
>> theta' =     0.0812    0.0801    0.0548
>> sum(sum(E')) =    3.2030  (> 3)
```

which does have energies that are above those for the equal-energy case of 1 unit on each user. The data rates are very close to the targets.

The margin would be determined by the ratio.

```
Gamma=6*((2^(sum(sum(b))/67)-1)/(2^(2*sum(sum(Bu))/67)-1)-1) = -6.0000 dB
```

or effectively about 1 of a bit/real-dimension below fundamental limit on a radial line through the point to $\mathscr{C}(\boldsymbol{b})$. The corresponding MAC receiver design for this alternative data rate close to maximum rate sum is then given by:

```
cb=1;
Usize=[1 1 1];
Bu=zeros(64,3);
Aopt=zeros(3,3,64);
for n=1:N
Aopt(:,:,n)=N*diag(E(:,n));
end
for n=1:N
[ Bu(n,:) GU(:,:,n), WU(:,:,n), S0(:,:,n), MSWMFU(:,:,n)] = ...
mu_mac(H(:,:,n), Aopt(:,:,n), Usize , cb);
end
sum(sum(Bu')) =   1044.9
>> GU(:,:,23) =
   1.0000 + 0.0000i  -2.1570 + 0.5562i  -0.0704 + 0.0712i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.1145 + 0.0597i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
```

The feedback section for the same tone changes slightly because the input energies changed slightly. Also, since minPMAC found an energy set that violates at least one user energy level, the sum rate of 1045 (or 15.6 bits/tone) exceeds the previous data rates because those were under strict adherence to energy constraints.

As an example of the commands that follow from the optimized input energies to design the MAC GDFE, with $\overline{N} = 8$ so that display of results is easier - the lower $\overline{N}$ value will cause minPMAC to exceed the allowed energy limit (which does not happen with $bN = 64$). The attempted data rates are reduced by a factor of 8 also from the 64-tone case, and the energy must be increased by 9/8 for the cyclic prefix, which causes the minPMAC data rates in the MAC-GDFE design to be missed.

```
N=8;
U=3;
Ly=2;
cb=1;
Lxu=[1 1 1];
bsum=zeros(1,N);
H8 = fft(h, N, 3);
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H8, [54 51 16]', [1 1 1]',1)
GU=zeros(U,U,N);
WU=zeros(U,U,N);
S0=zeros(U,U,N);
Bu=zeros(U,N);
MSWMFU=zeros(U,Ly,N);
AU=zeros(3,3,N);
  for n=1:N
    AU(:,:,n)= sqrtm(diag(Eun(:,n)));
  end
  for n=1:N
   [Bu(:,n), GU(:,:,n) , WU(:,:,n),S0(:,:,n), MSWMFU(:,:,n)] = ...
    mu_mac(H(:,:,n), AU(:,:,n), Lxu, cb);
  end
bvec=sum(Bu');
Bsum(index) = sum(bvec);
FEAS_FLAG =     1
 bvec =   54.7544   51.8108   16.1050
bu_a =   54.0000   51.0000   16.0000
>> GU
GU(:,:,1) =   1.0e+04 *
   0.0001 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0001 + 0.0000i  -1.7799 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0001 + 0.0000i
GU(:,:,2) =
   1.0000 + 0.0000i  -0.2004 + 0.0132i   0.1823 + 0.2030i
   0.0000 + 0.0000i   1.0000 + 0.0000i   1.3125 - 0.1839i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,3) =
   1.0000 + 0.0000i  -0.8690 + 0.1232i  -0.0108 + 0.0889i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.1887 + 0.0287i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,4) =
   1.0000 + 0.0000i  -1.7737 + 0.5843i  -0.4677 + 0.3006i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.8075 + 0.5128i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,5) =   1.0e+03 *
   0.0010 + 0.0000i  -1.7618 + 0.0000i  -9.6891 + 0.0000i
   0.0000 + 0.0000i   0.0010 + 0.0000i   0.0024 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0010 + 0.0000i
```

```
GU(:,:,6) =
   1.0000 + 0.0000i  -1.7737 - 0.5843i  -0.4677 - 0.3006i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.8075 - 0.5128i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,7) =
   1.0000 + 0.0000i  -0.8690 - 0.1232i  -0.0108 - 0.0889i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.1887 - 0.0287i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,8) =
   1.0000 + 0.0000i  -0.2004 - 0.0132i   0.1823 - 0.2030i
   0.0000 + 0.0000i   1.0000 + 0.0000i   1.3125 + 0.1839i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
>> MSWMFU
MSWMFU(:,:,1) =   1.0e+03 *
   0.0000 + 0.0000i   0.0000 + 0.0000i
  -2.2006 + 0.0000i   4.6158 + 0.0000i
   0.0001 + 0.0000i  -0.0003 + 0.0000i
MSWMFU(:,:,2) =
   0.0397 + 0.0155i   0.0086 + 0.0183i
   0.0584 + 0.0483i  -0.0191 - 0.1594i
   0.0481 + 0.0390i   0.0017 - 0.1107i
MSWMFU(:,:,3) =
   0.0406 + 0.0365i  -0.0162 + 0.0203i
   0.0135 + 0.0147i   0.0294 - 0.0304i
   0.2279 + 0.0713i   0.0673 - 0.1243i
MSWMFU(:,:,4) =
   0.0666 + 0.1166i  -0.0648 - 0.0085i
   0.0123 + 0.0189i   0.0479 + 0.0042i
   0.0819 + 0.0380i   0.0158 - 0.0243i
MSWMFU(:,:,5) =   1.0e+02 *
   7.7990 + 0.0000i  -7.7990 + 0.0000i
   0.0006 + 0.0000i   0.0009 + 0.0000i
  -0.0013 + 0.0000i   0.0010 + 0.0000i
MSWMFU(:,:,6) =
   0.0666 - 0.1166i  -0.0648 + 0.0085i
   0.0123 - 0.0189i   0.0479 - 0.0042i
   0.0819 - 0.0380i   0.0158 + 0.0243i
MSWMFU(:,:,7) =
   0.0406 - 0.0365i  -0.0162 - 0.0203i
   0.0135 - 0.0147i   0.0294 + 0.0304i
   0.2279 - 0.0713i   0.0673 + 0.1243i
MSWMFU(:,:,8) =
   0.0397 - 0.0155i   0.0086 - 0.0183i
   0.0584 - 0.0483i  -0.0191 + 0.1594i
   0.0481 - 0.0390i   0.0017 + 0.1107i
```

5.4.5 Admissible optimal MAC Designs for $\boldsymbol{b} \in \mathcal{C}(\boldsymbol{b})$

Any rate vector $\boldsymbol{b}_{min} \in \mathcal{C}(\boldsymbol{b})$ corresponds to a maximum weighted rate sum $\mathbf{1}^* \cdot \boldsymbol{b}_{min}$ with weights $\boldsymbol{\theta} \succeq \mathbf{0}$, although the designer may not have control of the specific $\boldsymbol{\theta}$. That rate vector $\boldsymbol{b}_{min}$ also corresponds to an energy vector $\boldsymbol{\mathcal{E}}' \in \mathcal{C}_{\boldsymbol{b}_{min}}(\boldsymbol{\mathcal{E}})$ that minimizes a weighted energy sum with weights $\boldsymbol{w} \succeq \mathbf{0}$. Again, the designer may not control the $\boldsymbol{w}$. Such a $\boldsymbol{b}_{min}$ is admissible because it is in the capacity region.

Design Specification Options: If the designer specifies the $\boldsymbol{b}_{min}$ and $\boldsymbol{\mathcal{E}}_{max}$, then presumably those are the targets. The corresponding best weight vectors $\boldsymbol{\theta}$ and $\boldsymbol{w}$ are consequent to the $[\boldsymbol{b}_{min}\ \boldsymbol{\mathcal{E}}_{max}]$ specification. The previous subsection (5.4.3) instead either specified $[\boldsymbol{b}_{min}\ \boldsymbol{w}]$ for minimum weighted sum-energy (including minimum energy sum when $\boldsymbol{w} = \mathbf{1}$ or specified $[\boldsymbol{\mathcal{E}}_{max}\ \boldsymbol{\theta}]$ for maximum weighted rate sum (including maximum sum rate when $\boldsymbol{\theta} = \mathbf{1}$). This subsection develops the direct specification $[\boldsymbol{b}_{min}\ \boldsymbol{\mathcal{E}}_{max}]$. Such a point is admissible on a given channel if $\boldsymbol{b}_{min} \in \mathcal{C}(\boldsymbol{b})$, and consequently $\boldsymbol{\mathcal{E}}_{max} \in \mathcal{C}_{\boldsymbol{b}_{min}}(\boldsymbol{\mathcal{E}})$.

A direct design's development recognizes that both the weighted rate-sum and the weighted energy-sum are individually convex[79]. Subsection (5.4.3)'s iterative $\boldsymbol{\theta}$-ellipsoid optimization updates $\boldsymbol{\theta}$ so that corresponding rate vectors $\boldsymbol{b}$ converge to the best Lagrange multipliers $\boldsymbol{\theta}$ for $\boldsymbol{b}_{min}$ for some given $\boldsymbol{w}$ and corresponding minimum weighted energy sum. Similarly, the maximum weighted rate sum for given $\boldsymbol{\theta}$ produces a $\boldsymbol{w}$ as Lagrange multipliers for the users' energy contsraints. Alternation between these two

[79] With the order indicated by ranking the values in $\boldsymbol{\theta}$.

objectives to find a pair $[\boldsymbol{\theta}\ \boldsymbol{w}]$ that jointly satisfies both will lead to a zero duality gap for admissible designs in (5.304) and (5.305), as Figure 5.39 shows.

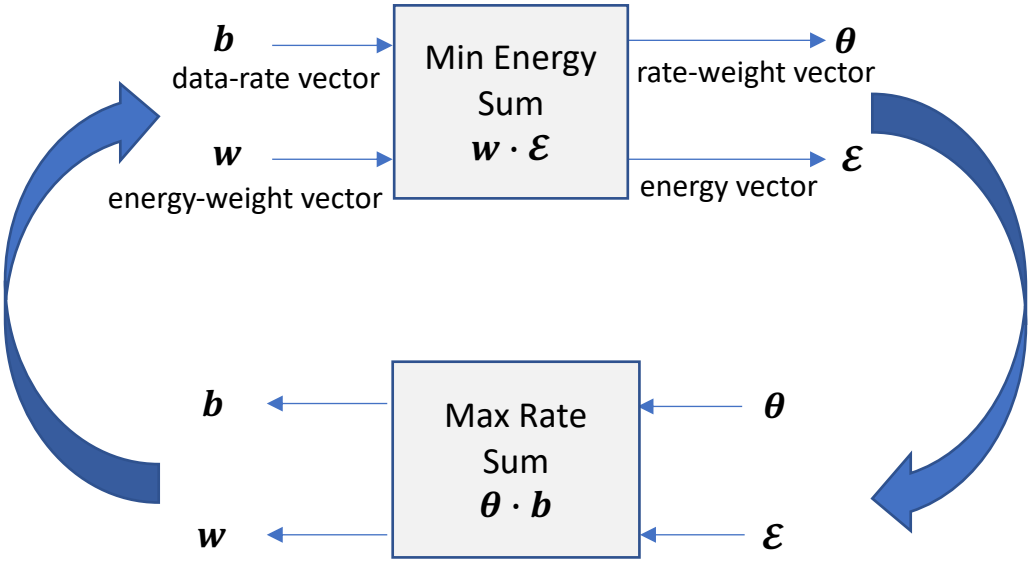


Figure 5.39: Iterative design given data-rate and energy vectors.

If initially $\boldsymbol{\theta} = \mathbf{1}$, then the weighted rate-sum maximization (e.g., maxRMAC programs) meet energy constraints and produce the maximum rate sum. If $\mathbf{1} \cdot \boldsymbol{b} \geq \mathbf{1} \cdot \boldsymbol{b}_{min}$, then design can continue. (If not, $\boldsymbol{b}_{min}$ is clearly inadmissible). If any user rate does not meet $b_i \geq b_{i,min}$, then the design needs a new updated $\boldsymbol{\theta}$; otherwise the design is complete. If not complete, the ellipsoid update iteratively updates $\boldsymbol{\theta}$ so that the current $\boldsymbol{b}$ solution moves closer to the desired $\boldsymbol{b}_{min}$. In particular θ_u values corresponding to $b_u < b_{min,u}$ need to increase relative to other users. Their θ_u values also should increase relative at least to users with excess data rate through (5.330) and (5.331). Then another cycle of the (newly) weighted rate-sum maximization should occur. If the new weighted rate sum is such that $\boldsymbol{\theta} \cdot \Delta\boldsymbol{b} < 0$, the new $\boldsymbol{\theta}$ is outside the ellipse centered at the new $\boldsymbol{b}$ and the rate vector is not admissible. Adjusting the $\boldsymbol{\theta}$ to move this point back in essentially would undo the previous ellipsoid update that was steepest descent, which means the problem can't be solved (the data rate is too high for the channel and given energy vector). Each execution of the weighted rate sum produces a new $\boldsymbol{w} \succeq 0$ if the point is admissible. Ultimately this $\boldsymbol{w}$ and the associated $\boldsymbol{\theta}$ will be such that (5.304) and (5.305) essentially cancel and the duality gap is zero, or abort if $b_{min} \notin \mathcal{C}(\boldsymbol{b})$. This provides a way to test $\boldsymbol{b}' + \gamma_b \odot \mathbf{1} \in \mathscr{C}(\boldsymbol{b})$ rates by increasing γ_b until the procedure produces an admissible point on the boundary, that is, a vector $\boldsymbol{b}'$ for which no further increase of γ_b is possible.

5.4.5.1 Matlab admMAC programs

Thus the culmination of this section's programs are the Matlab admMAC programs that execute Figure 5.39's joint optimization. By itself, the ellipsoid optimization cannot resolve a situation where two (or more) successive values are equal, $\theta_i = \theta_{i+1}$. This situation corresponds to a rate sum for the two users that is best, and that any combination of the corresponding users' rates that sums to the proper value is acceptable (and admissible). Nonetheless, that $\boldsymbol{b}$ value may not equal $\boldsymbol{b}_{min}$ for users $u = i$ and $u = i+1$. And indeed that corresponding rate sum will lead to a $\Delta\tilde{\boldsymbol{b}}$ in the null space of the ellipsoid (which equivalently is singular or has two dimensions that are identical like a hypersphere). This first execution of the weighted rate-sum maximization for a current given $\boldsymbol{\theta}$ is nonetheless a potential vertex. All its user-subsets for rate sums are faces of an initial capacity-region estimate. The admMAC_cvx program will find the proper vertex sharing when it is needed (corresponding, as with minPMAC to FEAS_FLAG

=2. This program is courtesy of Dr. Mehdi Mohseni and Dr. Yun Liao, with significant edits by this author.

Matlab admMAC.m : A matlab program that finds a design, when feasible, for a given $\boldsymbol{b}_{min}$ and $\boldsymbol{\mathcal{E}}_{max}$ is admMAC_cvx.m

```
>> help admMAC
  function [FEAS_FLAG, bu_a, info] = admMAC_cvx(H, Lxu, bu, Eu, cb)
  admMAC_rate_region determines whether the target rate vector bu is
  feasible for (noise-whitened) channel H and energy/symbol Eu via rate region
    Input arguments:
        - H: Ly-by-Lx-by-N channel matrix. H(:,:,n) denotes the channel for
            the n-th tone.
        - Lxu: number of transmit antennas of each user. It can be either a
            scalar or a length-U vector. If it is a scalar, every user has
            Lxu transmit antennas; otherwise user u has Lxu(u) transmit
            antennas.
        - bu: target rate of each user, length-U vector.
        - Eu: Energy/symbol on each user, length-U vector.
    Outputs:
        - FEAS_FLAG: indicator of achievability. FEAS_FLAG=0 if the target
            is not achievable; FEAS_FLAG=1 if the target is achievable by a
            single ordering; FEAS_FLAG=2 if the target is achievable by
            time-sharing
        - bu_a: U-by-1 vector showing achieved sum rate of each user.
        - info: various length output depending on FEAS_FLAG
            --if FEAS_FLAG=0: empty
            --if FEAS_FLAG=1: 1-by-5 cell array containing
                {Rxxs, Eun, bun, theta, w} corresponds to the single vertex
            --if FEAS_FLAG=2: v-by-7 cell array, with each row representing
                a time-shared vertex {Rxxs, Eun, bun, theta, w, frac, cluster}
       info's row entries in detail (one row for each vertex shared
        - Rxxs: U-by-N cell array containing Rxx(u,n)'s if Lxu is a
            length-U vector; or Lxu-by-Lxu-by-U-by-N tensor if Lxu is a
            scalar. If the rate target is infeasible, output 0.
        - Eun:  U-by-N matrix showing users' transmit energy on each tone.
            If infeasible, output 0.
        - bun: U-by-N matrix showing users' rate on each tone. If
            infeasible, output 0.
        - theta: U-by-1 Lagrangian multiplier w.r.t. target rates
        - w: U-by-1 Lagrangian multiplier w.r.t. energy constraints
        - order: U-by-1 user order
        - frac: fraction of dimensions for each vertex in time share (FF =2
            ONLY), per cluster
        -- cluster 1 has largest common-theta, cluster 2 has second largest
            common-theta group, and so on
       Subroutines called
        - maxRMAC_cvx
        - maxRMACMIMO
 ************************************************************************
```

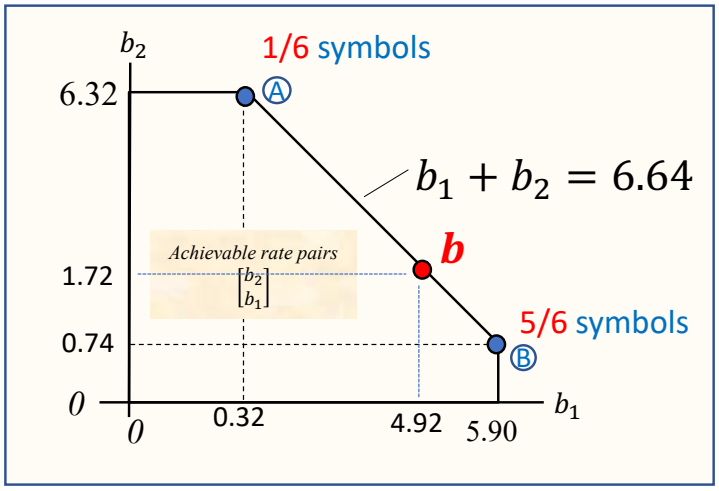


Figure 5.40: Capacity region for real-baseband $H = [80\ \ 60]$ channel in Example 5.4.12.

EXAMPLE 5.4.12 *[Vertex-Share Example]* This example revisits the earlier often-used $H = [80\ \ 60]$ channel to achieve a point that clearly requires time sharing from previous Chapter 2 Section 7 invocations, and for which the capacity region reappears in Figure 5.40:

```
H = [80 60];
>> bu_min = [1.72;4.92];
>> Eu = [1;1];
Lxu=[1 1];
>>  [FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H, Lxu, bu_min, Eu, cb)
flag =     2
bu_achieved =    1.7210    4.9229
info =
  2x8 table
          bu_v              Rxxs           Eun             bun            theta           order          frac      clu
    ___________________    __________    ___________    ___________    ___________    ___________    _______    ---
      6.322     0.32189    {2x1 cell}    {1x2 double}    {1x2 double}    {1x2 double}    {1x2 double}    0.17621
    0.73684      5.9071    {2x1 cell}    {1x2 double}    {1x2 double}    {1x2 double}    {1x2 double}    0.82379
>>Rxxs    %  2x1 cell array  same both vertices
    {[     1]}
    {[1.0000]}>> info.Eun{:,:}
> Eun =   % same both vertices
    1.0000
    1.0000
>> theta =
    0.9272
    0.917
>> w =
    0.5969
    0.3302
>> info.frac =
    0.1762
    0.8238
>> info.bu_v'*info.frac =
    1.7210
    4.9229  (vertex-sharing checks)
    >> info.order{:} =
     2     1
     1     2
```

In this case, $\boldsymbol{b}_{min} \in \mathscr{C}(\boldsymbol{b})$ – that is on the capacity-region boundary. So the vertex $\boldsymbol{b}^*_{\text{Ⓐ}} = [6.322\ .322]$ occurs roughly 17.6 % of the uses and the other vertex $\boldsymbol{b}^*_{\text{Ⓑ}} = [.737\ 5.907]$ occurs the remaining 82.4% of the uses. The different order of course implies a different MAC receiver decoding order for each shared vertex. So roughly, the receiver uses 1/6 symbols

with one order and the remaining 5/6 of symbols with the other order. Different orders correspond to subuser components and tacitly $U' = 4 > U = 2$. The point can be tested for extra margin:

```
[flag, bu_achieved, info] = admMAC(H, 1, bu_min+[1 ; 1], Eu,2)
flag =     0
>> [flag, bu_achieved, info] = admMAC(H, 1, bu_min+[0.1 ; 0.1], Eu,2)
flag = 0
```

which confirms the point is on the boundary, $\boldsymbol{b} \in \mathcal{C}(\boldsymbol{b})$.

For the same $\widetilde{H}$, but viewed as complex baseband (cb =1), admMAC provides a higher-performing design because two more dimensions become available; admMAC now attempts to achieve only the input data rate $\boldsymbol{b} = [1.72 \;\; 4.92]^t$ that is well within the enlarged capacity region in Figure 5.41. The design for the data rate [$\boldsymbol{b}$ now performs better because it is relatively speaking half the rate attempted earlier when on the boundary.

```
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H, Lxu, bu_min, 2*Eu, 1);
flag =     2
>> bu_a =
    3.4847
    9.9679
>> Eun =
    2.0000
    1.9989
flag =     2
bu_achieved =    3.6504   10.4418
info =  2x8 table
          bu_v              Rxxs           Eun             bun            theta           order          frac      clus
    _________________    __________    ___________    ___________    ___________    ___________    ________    ____
    1.4732     11.814     {2x1 cell}    {1x2 double}    {1x2 double}    {1x2 double}    {1x2 double}    0.83472       1
    13.644    .64352     {2x1 cell}    {1x2 double}    {1x2 double}    {1x2 double}    {1x2 double}    0.16528       1
>> info.order{:} =
     1     2
     2     1
```

The vertex sharing produced by admMAC sums to less than 1, indicating that $\boldsymbol{b}$ is in the capacity-region interior. Transmission of zero energy for portions of the time wastes dimensions and is not a good design. The design proceeds then to find a positive-margin image of $\boldsymbol{b}' \in \mathscr{C}(\boldsymbol{b})$. For this example, the capacity region is readily found as in Figure 5.41, indeed using the admmac command also. For the admMAC command, the Eu input is energy per SYMBOL, so it should be doubled if the complex system effectively operates at 1/2 the symbol rate for a common bandwidth. The present admMAC command seems better suited to finding capacity-region boundary points than for finding interior points (for which a few minPMAC tries may be more wise).

```
-----------------------------------------------
>> [flag, bu_achieved, info] = admMAC(H, Lxu, 2*bu_min, 2*Eu, 1)
flag =    2
bu_achieved =
    3.4400
    9.8400
info =  2x8 table
          bu_v              Rxxs           Eun             bun            theta           order          frac      clus
    _________________    __________    ___________    ___________    ___________    ___________    ________    ____
    12.644    0.64377    {2x1 cell}    {1x2 double}    {1x2 double}    {1x2 double}    {1x2 double}    0.19487
    1.4738     12.814    {2x1 cell}    {1x2 double}    {1x2 double}    {1x2 double}    {1x2 double}    0.80513
>> slope=(info.bu_v(1,1)-info.bu_v(1,2))(info.bu_v(2,1)-info.bu_v(2,2)) =   -0.7954
>> int=info.bu_v(1,1)-slope*info.bu_v(1,2) = 10.8706
>> rsum=sum(bu_a)=   13.4526
>> gammab=(rsum-bu_min(2)-bu_min(1))/2 =     3.4063
```

The last few commands compute the line connecting the two vertices and the intersection with that line used for the margin calculation, which then corresponds to 10.2 dB roughly for this data rate well within the capacity region interior. The margin can also be inferred from a few runs of the admMAC program with increasing bit gap:

```
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H, Lxu, bu_min +4*[1; 1], 2*Eu, 1)
flag =     0
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H, Lxu, bu_min +3.4*[1; 1], 2*Eu, 1)
flag =     0
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H, Lxu, bu_min +3.3*[1; 1], 2*Eu, 1)
flag =     2
bu_a' =    5.2878    8.6585
sum(bu_a) =   13.9464
```

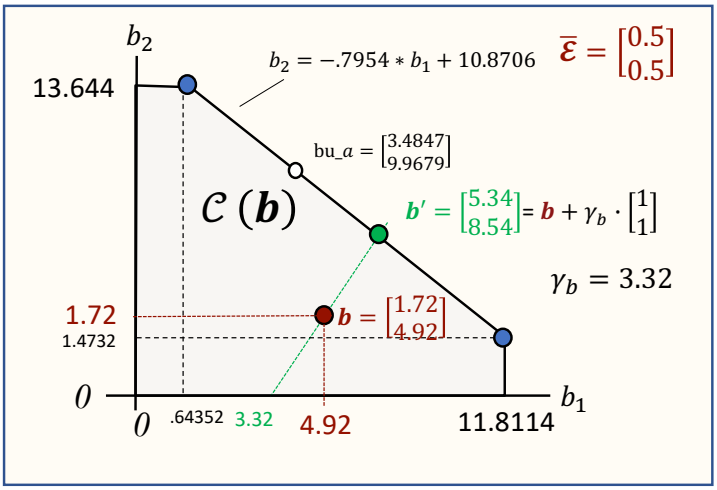


Figure 5.41: Capacity region for complex-baseband $H = [80\ \ 60]$ channel in Example 5.4.12.

The designer could experiment with various strategies that might apply unequal margin to the different users simply by changing the γ_b to be variable with user.

Or, now that it is clear the point is feasible, minimize the energy sum to achieve it:

```
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H, bu_min, [1 1], 1)
Eun =
    0.0109    0.0081
theta =
    0.0312    0.0385
bun =
    1.7200    4.9200
FEAS_FLAG =     1
bu_a =    1.7200              4.9200
```

The energy sum required for this case is very low.

Ellipsoid initialization: Initialization is easier in this case. 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 covers the unit cube and would then be:

$$A_0^{-1} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{bmatrix} \quad . \tag{5.340}$$

The $R_{\boldsymbol{xx}}$ 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 algorithm computes the tonal Lagrangian in exactly the same manner, but the overall sum should conform to the Lagrangian in (5.315).

Tracing the Capacity Region: The capacity region is then found by gradually increasing the elements of $\boldsymbol{b}$ in a nested U-loop that 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.

5.4.5.2 Further examples of the use of admMAC

This section returns to the Example 5.4.4 to find the admissibility and then also the proper weight vector for use in minPMAC.

For the complex baseband inter-symbol interference example of Section 2.7 a similar investigation produces

EXAMPLE 5.4.13

```
>> H=zeros(1,2,8);
>> H(1,1,:)=fft([1 .9],8);
>> H(1,2,:)=fft([1 -1],8);
>> H=(1/sqrt(.181))*H;
>> b=[1
1];
>> energy=[8
8];
```

This higher energy essentially corresponds to the 8 dimensions each with one energy unit. Additionally, these 8's are scaled by the 8/9 cyclic prefix factor loss on the input for this channel.

```
FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H,[1 1], 18*b, (8/9)*energy,1)
FEAS_FLAG =     2
bu_a' = 19.6385            19.6385
info =  2x7 table
          bu_v              frac
______________   ______
 21.698    17.579    0.5814
16.778    22.499    0.4186
>> buntop=info.bun{1,:,:}; bunbot=info.bun{2,:,:};
>> reshape(buntop,2,8) =
 5.2089    4.9796    3.2601    0.0047    0.0000    0.0047    3.2601    4.9796
         0    0.0001    1.0629    5.0737    5.3058    5.0737    1.0629    0.0001
>> reshape(bunbot,2,8) =
5.2089    4.9768    0.8077    0.0001    0.0000    0.0001    0.8077    4.9768
         0    0.0028    3.5152    5.0783    5.3059    5.0783    3.5152    0.0028
>> Eun =
    1.8043    1.7937    0.8580    0.0011    0.0006    0.0011    0.8580    1.7937
    0.0005    0.0006    0.9443    1.7381    1.7447    1.7381    0.9443    0.0006
>> sum(Eun,2)' =    7.1105    7.1111
info =
  2x8 table
          bu_v              Rxxs             Eun                bun              theta            order          frac      c
    _________________    __________    _____________    _____________    ____________    ____________    _______    -
    21.698    17.579    {1x8 cell}    {1x2x8 double}    {1x2x8 double}    {1x2 double}    {1x2 double}    0.5814
    16.778    22.499    {1x8 cell}    {1x2x8 double}    {1x2x8 double}    {1x2 double}    {1x2 double}    0.4186


>> info.order{:,:} =
     2     1
     1     2
   >> info.w{:} =
    0.5160   needs check on new admMAC
    0.5702
 =
    0.2646  needs check on new admMAC
    0.7212
```

So this is a valid solution because the total energy is less than (equal to) 64/9 for each user and the bit rates are those desired.

Lastly, the 64-tone 3-user channel from earlier is revisted here:

EXAMPLE 5.4.14 *3-user 64-tone Example revisited with admMAC* An alternative approach to the 3-user design of Example 5.4.1 might use admMAC

```
>> >> [FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H64, [1 1 1], [410 390 210]', 64^2/67*[1 1 1], 1)
flag =     2
bu_achieved =
```

```
  410.3922  390.3730  210.2009
info =  3x8 table
              bu_v                  Rxxs              Eun               bun             theta           order
    ____________________________    __________    _____________    _____________    ___________    ___________
    481.11    386.38    143.47    {1x64 cell}    {1x3x64 double}    {1x3x64 double}    {1x3 double}    {1x3 double}
    385.85    207.15    417.96    {1x64 cell}    {1x3x64 double}    {1x3x64 double}    {1x3 double}    {1x3 double}
    207.15    417.96    385.85    {1x64 cell}    {1x3x64 double}    {1x3x64 double}    {1x3 double}    {1x3 double}
    >> info.bu_v'*info.frac =  410.3922  390.3730  210.2009
>> sum(cell2mat(info.Eun),3) =
   61.1343   61.1343   61.1343
   61.1343   61.1343   61.1343
   61.1343   61.1343   61.1343


 >> info.order{:} =
     3     2     1
     2     1     3
     1     3     2
```

An alternative design that circumvents the order ambiguity caused by the all equal thetas on one of the outputs is to use minPMAC with different user weightings and the same target data rates:

```
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H,[410 390 210]', [1 1 1], 1);
theta =    2.8865    2.8865    2.8739
FEAS_FLAG = 2
bu_a =  410.0000  390.0000  210.0002
theta =    2.9830    2.9830    2.9830
info =  2x6 table
              bu_v                      Eun                          bun                        theta
    410.24    389.76    210    {1x3x64 double}    {1x3x64 double}    {1x3 double}     0.98818        1
    390.02    409.98    210    {1x3x64 double}    {1x3x64 double}    {1x3 double}    0.011823        1
>> Eun=info.Eun{1,:,:};  >> sum(Eun(1,:,:),3) =   59.5706   56.2791   66.2707
The last user misses energy constraint, try weightings
 >> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H,[410 390 210]', [1.03 1 1.08], 1) bu_a' =  410  390  210
info =  2x7 table
             bu_v                          Eun                bun             theta         order         frac      cluste
    410    397.72    202.28    {1x3x64}    {1x3x64}    {1x3}    {1x3}      0.9677         1
    410    158.72    441.28    {1x3x64}    {1x3x64}    {1x3}    {1x3}    0.032299       1
>> Eun=info.Eun{1,:,:};
>> sum(Eun(1,:,:),3) = 60.9146   60.3152   61.0854 - All meet constraint of 61.1
```

Then, using the ouptuts from the programs

```
>>  for n=1:N
Aopt(:,:,n)=diag(Eun(1,:,n));
end
>> Bu=zeros(64,3);
>> GU=zeros(3,3,64);
>> WU=zeros(3,2,64);
>> S0=zeros(3,3,64);
>> Wu=zeros(3,3,64);
>> MSWMFU=zeros(3,2,64);
for n=1:N
[ Bu(n,:) GU(:,:,n), WU(:,:,n), S0(:,:,n), MSWMFU(:,:,n)] = ...
mu_mac(H(:,:,n), Aopt(:,:,n), [1 1 1] , cb);
end
>> GU(:,:,23) =
   1.0000 + 0.0000i  -2.0650 + 0.5325i  -0.3842 + 0.3887i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.6506 + 0.3377i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
```

This design's user energies no longer have equal weight in the minimized energy sum. This design actually has slightly better data rates with more energy on user 1 but still meets (roughly) the energy-vector constraint. The WU, GU, etc for each tone can be computed. Then the fractional entries from info relate how often. Since the 3rd vertex is rarely used, it can be ignored. Thus 2 vertices have uses roughly 98 of 100 on the top vertex, and 2 on the other vertex.

Comment on ODM: Cellular and Wi-Fi wireless systems often aggregate many tones into resource blocks (cellular) or "channels" (Wi-Fi) for which the energy/tone is constant for all tones within the tone set. Such systems can treat each set as a single tone in the use of minPMAC and admMAC processes. A restriction to OMA (orthogonal multiple access) means no user sharing, or no crosstalk allowed. Clearly such systems cannot improve performance over the "NOMA" optimums, and indeed have more complex searches necessary to calculate best ODM dimensional use (which becomes evident in Section 5.5. It may nonetheless with a sufficient number of tone sets (which may occur through MIMO's ability to allocate different energy to different spatial dimensions) to see an ODM like nature to the minPMAC and admMAC solutions, which provides a simpler estimate. Indeed, such ODM restriction, in effect, creates an IC and Chapter 2's optimum spectrum balancing and approximations thereto become a means to find best energy allocation, albeit at performance loss relative to this section's more simply attained actual optimum if the situation is indeed a MAC or a BC.

5.4.5.3 Distributed Implementations

Figure 5.42 shows a possible MAC implementation where a controller executes the resource (bits and energy) allocation for all U users and passes the resulting information/user and energy/user distributions to each user. A noise change (or an energy/rate change of any of the users), or a channel H_u change, could cause need for the re-execution of Mohseni's Algorithm. The area of efficient incremental changes remains is a good one for research. Alternatively, Section 4.4's Separation Theorem and Nested Loading may find use with appropriate gaps or tolerances.

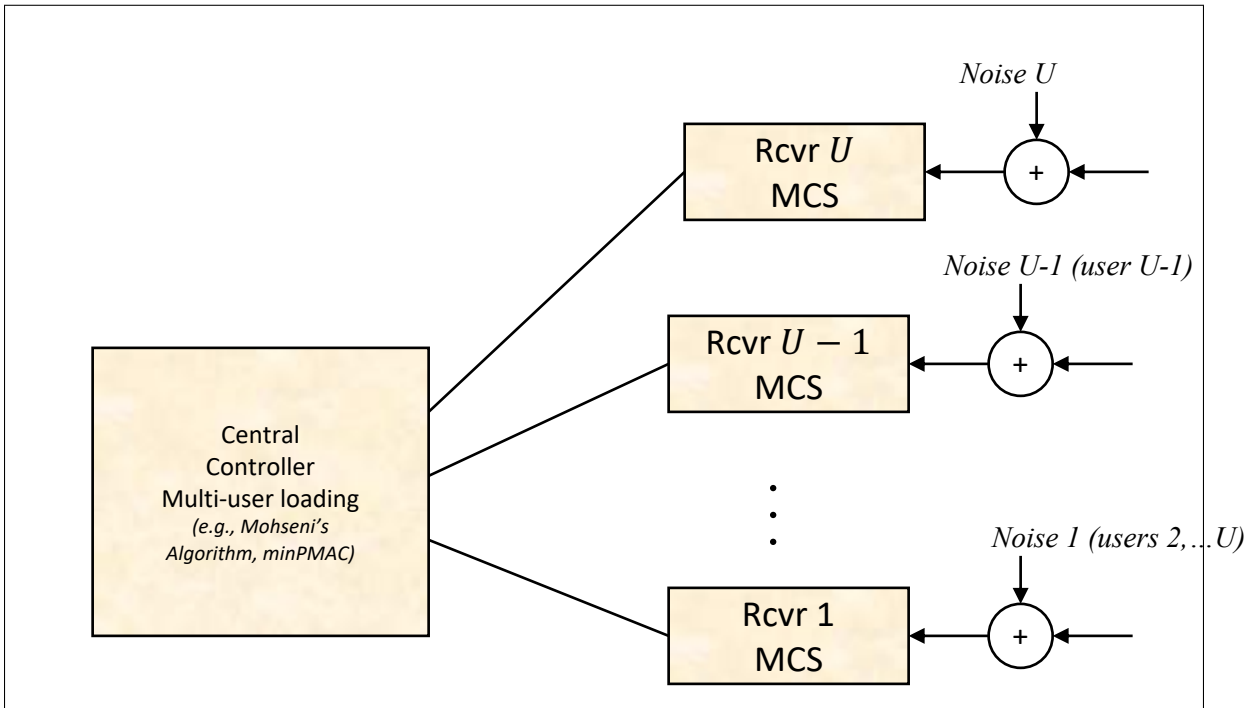


Figure 5.42: Central Control for swapping .

5.5 The Gaussian BC via Duality and GDFE

Figure 5.43's vector BC GDFE channel has diagonal receiver processing, and an $R_{\boldsymbol{xx}}$-dependent worst-case noise that determines primary-components' receiver structure, as in Chapter 2, Section 8. Alternately, with known $R_{\boldsymbol{xx}}(u)$, a MMSE process specifies the precoder matrix G, modulator matrices A, and each receiver's $L_x \times L_{y,u}$ pre-detector matrix processing[80]. This section shows that to determine the $\{R_{\boldsymbol{xx}}(u)\}$ for a particularly $\boldsymbol{b}$, it is possible to re-use Section 5.4's MAC design through vector duality. For any given admissible rate vector $\boldsymbol{b}$, vector duality finds the BC's $\{R_{\boldsymbol{xx}}(u)\}$ from an equivalent dual-MAC set. This design avoids worst-case noise. The MAC's $\boldsymbol{\mathcal{E}}$ observes only an energy-sum constraint: Thus, a designer can run minPMAC[81] can run for any given $\boldsymbol{b}_0$ and $\boldsymbol{w} = \mathbf{1}$ to check if the resultant $\boldsymbol{w}^* \cdot \boldsymbol{\mathcal{E}}_{\boldsymbol{x}} \leq \mathcal{E}_x$. If so, $\boldsymbol{b}_0 \in \mathcal{C}(\boldsymbol{b})$ is admissible, and if not, $\boldsymbol{b}_0$ is an inadmissible rate vector for the energy constraint $\mathcal{E}_x$, $\boldsymbol{b}_0 \notin \mathcal{C}(\boldsymbol{b})$. Any boundary point $\boldsymbol{b}_0 \in \mathcal{S}_{e-sum,mac}(\boldsymbol{b})$ corresponds to a best BC design and produces $\boldsymbol{\mathcal{E}}_{\boldsymbol{x}} \ni \boldsymbol{\mathcal{E}}_{\boldsymbol{x}} \cdot \mathbf{1} = \mathcal{E}_x$. The BC-design thus uses minPMAC (and/or admMAC, maxRMAC, etc) to design $R_{\boldsymbol{xx}}$ and/or $\boldsymbol{b}$ from a dual vector MAC and applies it to a vector BC for such admissible data rate on the dual energy-sum MAC.

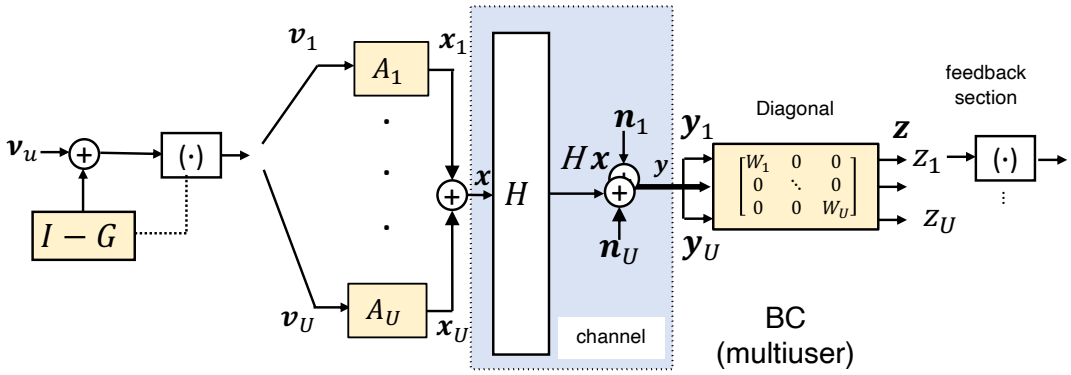


Figure 5.43: Broadcast Channel Review.

Subsection 5.5.1 re-investigates single-user vector-channel duality. Subsection 5.5.2 then expands duality to the MAC and corresponding dual vector BC. Subsection 5.5.4 then uses Chapter 2's MMSE design series (now recognized well as GDFEs) that construct the BC design's precoder (without need of worst-case noise). Subsection 5.5.3 revists Vector DMT for the BC. While Subsection 5.5.5 revisits generation of $\mathcal{C}_{BC}(\boldsymbol{b})$.

5.5.1 Single-User Dual Channels

A single-user dual channel transposes the channel and reverses dimensional order. Dual channels have the same sum rate and energy sum. Subsection 5.5.2 specializes dual channels that match a multiuser MAC to a multiuser BC.

Vector duality generalizes Section 2.8's scalar concept. Subsection 5.5.1.1 introduces duality concepts for any matrix channel. Subsection 5.5.2.1 then refines duality to determine a special set of autocorrelation matrices that apply to the dual vector BC and MAC. The more general GDFE approach taken here easily shows that the mutual information of dual channels must be the same (so the rate sums are the same) and that the energy sum is also the same.

[80] Each BC receiver estimates possibly as many as L_x possible sub-user components for user u.
[81] The admMAC addresses the MAC's energy-vector constraint, which vector is not part of BC design.

5.5.1.1 Single-User Dual-Matrix Channels - Any H, $R_{\boldsymbol{nn}}$, and $R_{\boldsymbol{xx}}$

Duality begins with a initial matrix AWGN

$$\boldsymbol{y} = H \cdot \boldsymbol{x} + \boldsymbol{n} \quad , \tag{5.341}$$

where H is an $\mathscr{L}_y \times \mathscr{L}_x$ matrix and therefore so is $\widetilde{H} \triangleq R_{\boldsymbol{nn}}^{-1/2} \cdot H$ for the equivalent white-noise channel with $\widetilde{H}$. The input has autocorrelation matrix $R_{\boldsymbol{xx}}$. Duality's most general mathematical representation can use the order-reversal matrices:

$$\mathcal{J}_x \triangleq \begin{bmatrix} 0 & 0 & I_{L_{x,1}} \\ 0 & \iddots & 0 \\ I_{L_{x,U}} & 0 & 0 \end{bmatrix} \quad \mathcal{J}_y \triangleq \begin{bmatrix} 0 & 0 & I_{L_{y,U}} \\ 0 & \iddots & 0 \\ I_{L_{y,1}} & 0 & 0 \end{bmatrix} \quad , \tag{5.342}$$

although use with matlab as in this tex's upcoming programs, the MAC and BC special cases will really need only $\mathcal{J}_x$.

> **Definition 5.5.1 [The Dual Channel]** For (equivalent white-noise) channel matrix $\widetilde{H}$, the **dual channel** is
>
> $$\widetilde{H}_{dual} \triangleq \mathcal{J}_x \cdot \widetilde{H}^* \cdot \mathcal{J}_y \quad . \tag{5.343}$$
>
> This text's $\widetilde{H}$ corresponds to a MAC and $\widetilde{H}_{dual}$ to a BC; however duality mapping can also be BC to MAC. The meaning of L_x and L_y, or more generally $\mathscr{L}_x$ and $\mathscr{L}_y$, for the dual to correspond to number of **outputs** and **inputs** respectively. The BC reversed ordering aligns with matlab's natural ordering, so it is convenient at times to ignore the $\mathcal{J}_y$ because that reversal occurs naturally without explicit consideration in software.

The order reversal may appear superfluous above, but it will be notationally convenient for MAC to/from BC duality. The channel has "economy mode" SVD[82]

$$\widetilde{H} = F \cdot \Lambda \cdot M^* \tag{5.344}$$

and so the dual channel has corresponding "economy-mode" SVD

$$\widetilde{H}_{dual} = \mathcal{J}_x \cdot \widetilde{H}^* = \mathcal{J}_x \cdot M \cdot \Lambda \cdot F^* \cdot \mathcal{J}_y \quad . \tag{5.345}$$

The matrix $\mathcal{J}_x \cdot M$ retains all orthonormal columns, so (5.345) is a valid "economy-mode" SVD for the dual channel. The dual channel input is $\bar{\boldsymbol{x}}$, as in Figure 5.45. When $\widetilde{H}$ is not square, the input $\boldsymbol{x}$ or $\bar{\boldsymbol{x}}$ with the smaller dimensionality may decompose into a sum of components equal in number up to the larger dimensionality. The dual-channel input is

$$\bar{\boldsymbol{x}} \triangleq \mathcal{J}_y \cdot F \cdot M^* \cdot \mathcal{J}_x \cdot \boldsymbol{x} \quad . \tag{5.346}$$

with autocorrelation matrices related as

$$R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}} = \mathcal{J}_y \cdot M^* \cdot \mathcal{J}_x \cdot R_{\boldsymbol{xx}} \cdot \mathcal{J}_x \cdot M \cdot F^* \cdot \mathcal{J}_y \quad . \tag{5.347}$$

If $\boldsymbol{x}$ is $U \times 1$ with $U > 1$, then $\bar{\boldsymbol{x}}$ consists of U components $\bar{\boldsymbol{x}} = \sum_{u=1}^{U} \bar{\boldsymbol{x}}_u$. The dual of the dual is the original channel.

[82] Economy-mode SVD retains column vectors of F and M that correspond only to non-zero singular values, so $H = F \cdot \Lambda \cdot M^*$ and Λ is a diagonal matrix of rank $\mathcal{Q}_H$. Matlab computes this with command svd(H,'econ') for matrix H. This means that F and F^* are $L_y \times \mathcal{Q}_{\widetilde{H}}$ and $\mathcal{Q}_{\widetilde{H}} \times L_y$ pseudoinverses of one another so that $F^* \cdot F = I_{\mathcal{Q}_{\widetilde{H}}}$ while $F \cdot F^* = \begin{bmatrix} I_{\mathcal{Q}_{\widetilde{H}}} & 0 \\ 0 & 0 \end{bmatrix}$, and similarly with M and M^*.

Theorem 5.5.1 **[Dual Single-User Channels]** *The channel* $\widetilde{H}$ *and its dual* $\widetilde{H}_{dual}$ *have the same:*

mutual information

$$\mathcal{I}(\boldsymbol{x};\boldsymbol{y}) = \mathcal{I}(\bar{\boldsymbol{x}};\bar{\boldsymbol{y}}) \text{ and} \tag{5.348}$$

input energy

$$trace\{R_{\boldsymbol{xx}}\} = trace\{R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}\} \quad . \tag{5.349}$$

Proof: The energy preservation and mutual-information invariance follows because the input transformation in (5.346) is both unitary and invertible on the input pass-space components that flow through the channel. The mutual information for the original channel with order reversal depends on the determinant

$$\left|\mathcal{J}_y \cdot \widetilde{H} \cdot \mathcal{J}_x \cdot R_{\boldsymbol{xx}} \cdot J_x \cdot \widetilde{H}^* \cdot \mathcal{J}_y + I\right| = |\Lambda \cdot M^* \cdot J_x \cdot R_{\boldsymbol{xx}} \cdot J_x \cdot M \cdot \Lambda + I| \tag{5.350}$$

since $|F| = |\mathcal{J}_y| = 1$. Similarly the dual channel's mutual information depends on the determinant

$$\left|\widetilde{H}_{dual} \cdot R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}} \cdot \widetilde{H}^*_{dual} + I\right| = |\Lambda \cdot F^* \cdot R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}} \cdot F \cdot \Lambda + I| \quad . \tag{5.351}$$

These two determinants are equal when (5.345), or equivalently (5.347) is true. Thus the two mutual information quantities (and indeed any GDFE SNR's), equivalently $S_{0,1}$ are the same. **QED.**

When $rank(R_{\boldsymbol{xx}}) = \varrho_x > \varrho_{\widetilde{H}}$, the input $\boldsymbol{x}$ has autocorrelation matrix $R_{\boldsymbol{xx}}$ that necessarily contains nonzero energy in the channel null space $\mathcal{N}_{\widetilde{H}}$. There will be thus rate loss; since both duals have the same $\mathcal{I}$, then both exhibit this same loss. The input energies of both the dual channel and the original channel satisfy

$$\mathcal{E}_{\boldsymbol{x}} = \mathcal{E}_{\bar{\boldsymbol{x}}} = \mathcal{E}_{\boldsymbol{x}}(\mathcal{P}_{\widetilde{H}}) + \mathcal{E}_{\boldsymbol{x}}(\mathcal{N}_{\widetilde{H}}) = \mathcal{E}_{\boldsymbol{x}}(\mathcal{P}_{\widetilde{H}^*_{dual}}) + \mathcal{E}_{\boldsymbol{x}}(\mathcal{N}_{\widetilde{H}^*_{dual}}) \quad . \tag{5.352}$$

If $\varrho_x \leq \varrho_{\widetilde{H}}$, the channel has sufficient dimensionality to avoid energy loss to the null space, although this requires proper input construction (which may not be possible, for instance with a MAC's separated input locations) so it is still possible for a good multiuserdesign's energy to enter the channel null space.

Notational Observations for $\widetilde{H}$ as a MAC and its dual as a BC: First this special case allows $\mathcal{J}_y \to I$, so remove it from the earlier equations, which all still hold. With $\widetilde{H}$ assumed as a MAC:

$$\underbrace{\widetilde{H}}_{L_y \times \mathscr{L}_x} = \widetilde{H}_{MAC} = \left[\underbrace{\widetilde{H}_U}_{L_y \times L_{x,U}} \quad ... \quad \underbrace{\widetilde{H}_1}_{L_y \times L_{x,1}}\right] \tag{5.353}$$

and has a dual $\mathscr{L}_x \times L_y$ matrix $\widetilde{H}_{BC} \triangleq \mathcal{J}_x \cdot \widetilde{H}^*$ that corresponds to dual BC channel

$$\widetilde{H}_{BC} = \mathcal{J}_x \cdot \widetilde{H}^*_{MAC} = \mathcal{J}_x \cdot \begin{bmatrix} \widetilde{H}^*_U \\ \vdots \\ \widetilde{H}^*_1 \end{bmatrix} = \begin{bmatrix} \widetilde{H}^*_1 \\ \vdots \\ \widetilde{H}^*_U \end{bmatrix} \begin{matrix} \} L_{x,1} \times L_y \\ \vdots \\ \} L_{x,U} \times L_y \end{matrix} \quad . \tag{5.354}$$

Note the $\mathcal{J}_y$ essentially moves to a reordered BC input with user $u = 1$ at the top, which is the natural order defined here for the BC (so the $\mathcal{J}_y$ essentially hides in an order-reversed BC-input dimensionality. Essentially since the BC input is $\boldsymbol{x}$, which sums all users components, the ultimate input order occurs

with $\boldsymbol{v}$ and $\boldsymbol{\nu}$ and would correspond there to user 1 at the top of these vectors that do have dimensionality specifically associating with user order. $\widetilde{H}_{BC}$'s input is a sum of U components

$$\bar{\boldsymbol{x}} = \sum_{u=1}^{U} \bar{\boldsymbol{x}}_u \quad , \tag{5.355}$$

while by contrast the dual MAC $\widetilde{H}$'s input is a stacked column vector of the input components

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_U \\ \vdots \\ \boldsymbol{x}_1 \end{bmatrix} \quad . \tag{5.356}$$

For the dual BC channel, the dimensional notation will match that of its MAC dual, as in Figure 5.44. The discrete modulator A is a fat matrix that adds within it all the BC user contributions. If that channel were not a dual, then its diagram would have $y \leftrightarrow x$ in the dimensional expressions.

Preparation for multiuser - and individual $\widetilde{H}_u$ element: Figure 5.44 illustrates a GDFE for user u's subset of the dual channel's row dimensions, $\widetilde{H}_u$.

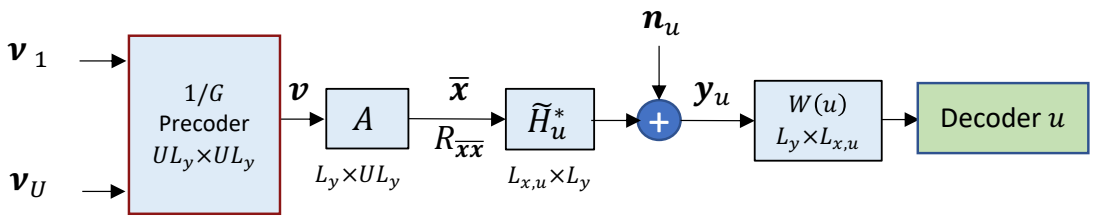


Figure 5.44: Dual Channel's (BC's) precoded GDFE for subset of dimensions indexed by u.

Figure 5.44's receiver linear processing is variables separable when estimating $\boldsymbol{v}$. Thus, it is the MMSE solution for those channel outputs that it processes[83], denoted $\boldsymbol{y}_u$. The BC input $\bar{\boldsymbol{x}}$ sums components from all users, not just user index u. Figure 5.44's receiver estimates all input dimensions. As before, a GDFE with lossless precoder essentially moves the feedback-section to the transmitter and retains the MMSE performance.

5.5.2 User-Level MAC-BC Duality

In addition to the overall duality (viewing $\widetilde{H}$ and $\widetilde{H}_{BC}$ as single-user channels), each vector BC user has dual $\widetilde{H}_u^*$ channel that corresponds to same dual-MAC user $\widetilde{H}_u$ channel that can have the same mutual information, as in Figure 5.44's specific receiver processing of only that user's BC output dimensions. The BC's order-reversal convention causes the best BC user position to correspond to the MAC user that is in the worst position; this is desired. This means $\boldsymbol{\pi}_{BC} = \mathcal{J}_x \cdot \boldsymbol{\pi}_{MAC}$. Single-user dual inputs may not satisfy the MAC's independent-input requirement. A BC maximum rate sum can never exceed it's dual energy-sum MAC's maximum rate sum. Further, their input-sum energies are equal.

$$\sum_{u=1}^{U} \mathcal{E}_u^B = \sum_{u=1}^{U} \mathcal{E}_u^M \quad . \tag{5.357}$$

Equivalently, macmax.m and bcmax.m produce the same data rate when their channel inputs are dual inputs for the corresponding dual MAC and BC channels. However, design facilitation prefers that the dual users also have equal data rates.

[83] A better solution occurs when the receiver processes all dimensions; but only $\boldsymbol{y}_u$ is available here. Figure 5.44 is a MMSE estimator for $\boldsymbol{y}_u$ from $\boldsymbol{v}$.

Input Deflection: **Input deflection** applies to Figure 5.45's individual-user channel $\widetilde{H}_u$ in a MAC/BC dual set. It is a consequence of noise-whitening's effect on a dual-channel input's appearance. While the channel noise $\boldsymbol{n}$ is white ($R_{\boldsymbol{nn}} = I$), the individual-user noises include crosstalk from other users, namely as[84]

$$\mathcal{R}_{\boldsymbol{nn}}(u) \triangleq I + \sum_{i=u+1}^{U} \widetilde{H}_i \cdot R_{\boldsymbol{xx}}(i) \cdot \widetilde{H}_i^* \tag{5.358}$$

$$\mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}(u) \triangleq I + \widetilde{H}_u^* \cdot \left[\sum_{i=1}^{u-1} R_{\boldsymbol{xx}}(i)\right] \cdot \widetilde{H}_u \ . \tag{5.359}$$

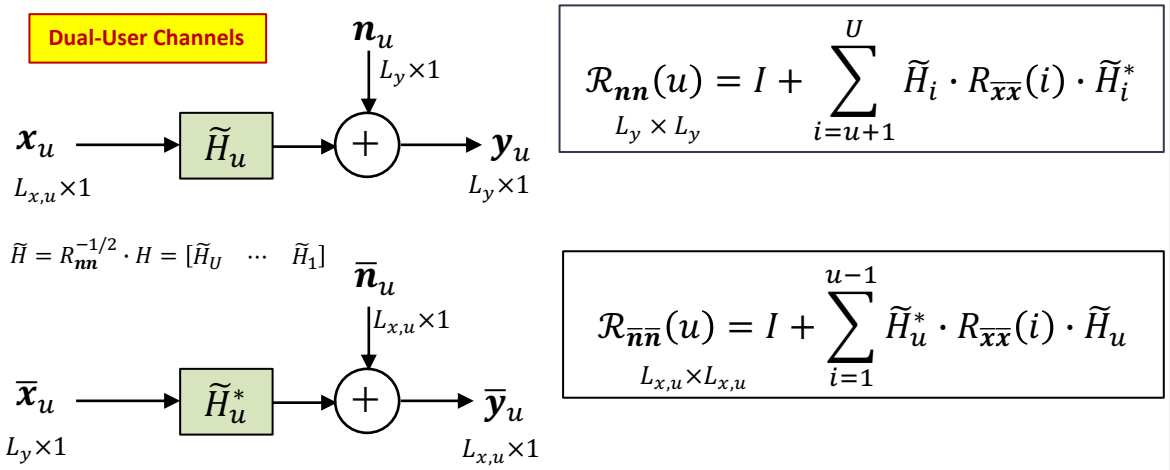


Figure 5.45: The dual-user theoretical channels, which essentially redefine noise and crosstalk.

Thus, the individual-users' mutual information expressions for any given $R_{\boldsymbol{xx}}(u)$ and $R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}(u)$ can equate through

$$2^{\mathcal{I}_u} = \frac{\left|\widetilde{H}_u \cdot R_{\boldsymbol{xx}}(u) \cdot \widetilde{H}_u^* + \mathcal{R}_{\boldsymbol{nn}}(u)\right|}{|\mathcal{R}_{\boldsymbol{nn}}(u)|} \tag{5.360}$$

$$= \frac{\left|\widetilde{H}_u^* \cdot R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}(u) \cdot \widetilde{H}_u + \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}(u)\right|}{|\mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}(u)|} \tag{5.361}$$

$$= \left|\grave{H}_u \cdot R_{\grave{\boldsymbol{x}}\grave{\boldsymbol{x}}}(u) \cdot \grave{H}_u^* + I\right| \tag{5.362}$$

$$= \left|\grave{H}_u^* \cdot R_{\grave{\bar{\boldsymbol{x}}}\grave{\bar{\boldsymbol{x}}}}(u) \cdot \grave{H}_u + I\right| \ , \tag{5.363}$$

where $\grave{H}_u$, $\grave{\boldsymbol{x}}_u$, and $\grave{\bar{\boldsymbol{x}}}_u$ have definition in the succeeding Lemma 5.5.1, and the last two expressions equivalently diagonalize the noise (including crosstalk) autocorrelation matrix with this **deflected-input channel**. The input-deflection lemma is general, but applies here in the context of the deflecting matrices being the square roots of a $\mathcal{R}_{\boldsymbol{nn}}(u)$ and $\mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}(u)$ for the vector MAC and BC dual set.

Lemma 5.5.1 [Input Deflection with Correlated Crosstalk Noise] *For two user-specific channels $\widetilde{H}$ and $\widetilde{H}^*$ with different noise autocorrelation matrices, as in Figure 5.45,*

[84] There is a font difference on $\mathcal{R}_{\boldsymbol{nn}}$ and $R_{\boldsymbol{nn}}$.

here more specifically as $\mathcal{R}_{\boldsymbol{nn}}$ *and* $\mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}$*, and the same mutual information, the related dual channels*

$$\grave{H}_u \triangleq \mathcal{R}_{\boldsymbol{nn}}^{-1/2}(u) \cdot \widetilde{H}_u \cdot \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{-*/2}(u) \quad , \tag{5.364}$$

and $\grave{H}^*$*, have the same mutual information with new-noise autocorrelation matrices* I *and deflected inputs*

$$\grave{\boldsymbol{x}}_u = \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{1/2}(u) \cdot \boldsymbol{x}_u \tag{5.365}$$

$$\grave{\bar{\boldsymbol{x}}}_u = \mathcal{R}_{\boldsymbol{nn}}^{*/2}(u) \cdot \bar{\boldsymbol{x}}_u \; . \tag{5.366}$$

Furthermore, from the input deflections,

$$R_{\grave{\boldsymbol{x}}\grave{\boldsymbol{x}}}(u) = \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{*/2}(u) \cdot R_{\boldsymbol{xx}}(u) \cdot \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{1/2}(u) \quad , \tag{5.367}$$

and equivalently

$$R_{\grave{\bar{\boldsymbol{x}}}\grave{\bar{\boldsymbol{x}}}}(u) = \mathcal{R}_{\boldsymbol{nn}}^{*/2}(u) \cdot R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}(u) \cdot \mathcal{R}_{\boldsymbol{nn}}^{1/2}(u) \quad . \tag{5.368}$$

Proof: Both noises are full rank because of the identy matrices in (5.358) and (5.359), and thus they are invertible 1-to-1 mappings and do not change mutual information, whether applied to input or output. Both matrices cause the noise-plus-crosstalk to be white when applied at output, so the mutual information value is then the same (denominator-determinant and numerator-determinant's added term are identities so the noise also has the same autocorrelation in both). **QED.**

Figure 5.46 illustrates the fully whitened dual-user channels.

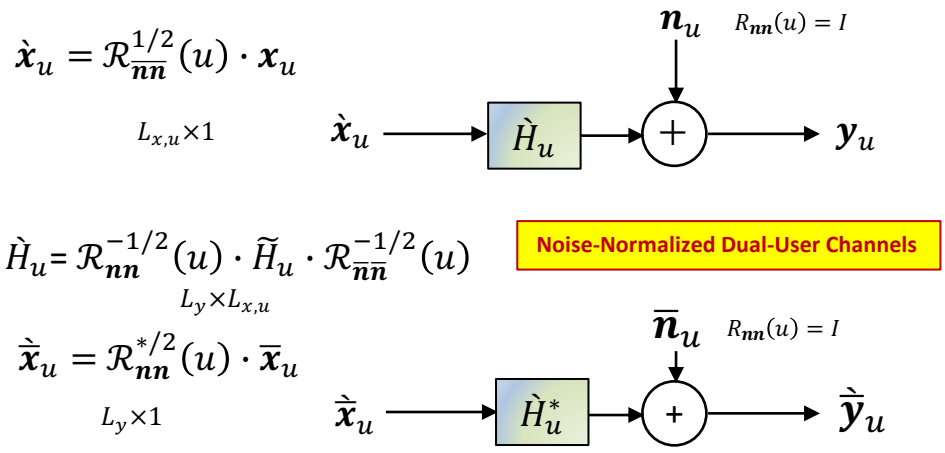


Figure 5.46: The dual of transpose channel with different noise autocorrelation matrix.

The input deflections can however change the transmitted energy, and indeed duality focuses on the energy sum for all users' matrix channels, not on equality of individual users' corresponding energies. User u's data rate is the same, b_u. The transformations in (5.365) and (5.366) do not introduce correlation between users if applied each separately for a $\mathcal{R}_{\boldsymbol{nn}}(u)$ and/or $\mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}(u)$, and so these mappings preserve user independence. The remaining duality step sets the user energy sums equal. However, first a scalar-duality reminder follows.

Scalar Duality Revisted By returning to the scalar duality in Subsection 2.8.4, the energy mappings from BC to/from MAC are again:

$$\mathcal{E}_1^{BC} = \mathcal{E}_1^{MAC} \cdot \frac{1}{1 + \mathcal{E}_2^{MAC} \cdot g_2 + ... + \mathcal{E}_U^{MAC} \cdot g_U} \tag{5.369}$$

$$\mathcal{E}_2^{BC} = \mathcal{E}_2^{MAC} \cdot \frac{1 + \mathcal{E}_1^{BC} \cdot g_2}{1 + \mathcal{E}_3^{MAC} \cdot g_3 + ... + \mathcal{E}_U^{MAC} \cdot g_U} \tag{5.370}$$

$$\vdots = \vdots \tag{5.371}$$

$$\mathcal{E}_U^{BC} = \mathcal{E}_U^{MAC} \cdot \left(1 + \left[\mathcal{E}_1^{BC} + ... + \mathcal{E}_{U-1}^{BC}\right] \cdot g_U\right) \tag{5.372}$$

$$\tag{5.373}$$

These equations are equivalent to writing the deflected inputs as

$$x_1^{BC} \cdot \sqrt{1 + \mathcal{E}_2^{MAC} \cdot g_2 + ... + \mathcal{E}_U^{MAC} \cdot g_U} = x_1^{MAC} \tag{5.374}$$

$$x_2^{BC} \cdot \sqrt{1 + \mathcal{E}_3^{MAC} \cdot g_3 + ... + \mathcal{E}_U^{MAC} \cdot g_U} = x_2^{MAC} \cdot \sqrt{1 + \mathcal{E}_1^{BC} \cdot g_2} \tag{5.375}$$

$$\vdots = \vdots \tag{5.376}$$

$$x_U^{BC} = x_U^{MAC} \cdot \sqrt{\left(1 + \left[\mathcal{E}_1^{BC} + ... + \mathcal{E}_{U-1}^{BC}\right] \cdot g_U\right)} \ . \tag{5.377}$$

These indeed are the input deflection mappings of Lemma 5.5.1, but simpler versions for the scalar case. which can be viewed on either side as noise whitening with corresponding input deflection for any u. The square-root terms include the noise plus the other – uncancelled/precoded – user effects in noise whitening the single scalar output or in deflecting any particular scalar input. The scalar energies sum to the same total on each of the scalar duals. The vector case requires more tedious bookkeeping to preserve the specified $R_{\boldsymbol{xx}}$ or $R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}$ and the corresponding energy sum. Prior to duality's vector generalization, an example is revisited to help illustrate scalar duality.

EXAMPLE 5.5.1 [Revisit of Chapter 2's scalar BC Example] Example 5.4.1's (noise-normalized/whitened) scalar channel is $H_{MAC} = [80\ 50]$, but this BC example will start with the dual

$$H_{BC} = \begin{bmatrix} 50 \\ 80 \end{bmatrix} \ , \tag{5.378}$$

for which the dual is $H_{BC} = \mathcal{J}_y \cdot H_{MAC}^*$ ($J_x = 1$). User 2 for both MAC and BC corresponds to the 80 (the primary user/component). The maximum MAC rate sum is 6.322 bits/sub-symbol with all energy $\mathcal{E}_2 = 1$ on user 2, so $\mathcal{E}_1 = 0$ for user 1 as the secondary user/component. If the initial BC energies are instead

$$\boldsymbol{\mathcal{E}}^{BC} = \begin{bmatrix} 1/4 \\ 3.4 \end{bmatrix} \rightarrow \begin{bmatrix} 1/7504 \\ 7503/7502 \end{bmatrix} = \boldsymbol{\mathcal{E}}^{MAC} \ , \tag{5.379}$$

then the dual-channel's MMSE MAC design follows:

```
>> Hmac=[50 80];
>> Rxx=diag([1/7504 7503/7504]);
>> A=sqrtm(Rxx);
>> [Bu, GU, WU, S0, MSWMFU] = mu_mac(Hmac, A, [1 1] , 2)
Bu =
    0.2074    6.1146
GU =
    1.0000  138.5918
         0         1.0000
WU =
    3.0016         0
   -0.0072    0.0002
S0 =   1.0e+03 *
    0.0013         0
         0    4.8010
```

```
MSWMFU =
    1.7325
    0.0125
>> sum(Bu) =    6.3220
>> MSWMFU*Hmac*A  =
    1.0000  138.5918
    0.0072      1.0000
```

The corresponding BC could also be designed directly in this simple case as

```
>> Hbc=[80 ; 50];
>> AU=[sqrt(.75) sqrt(.25)];
>> Lyu=[1 1];
>> cb=2;
>> [Bu, GU, S0, MSWMFunb , B] = mu_bc(Hbc, AU, Lyu , cb)
Bu =
    6.1146    0.2074
GU =  2x1 cell array
S0 =  2x1 cell array
    {[4.8010e+03]}
    {[    1.3332]}
MSWMFunb = 2x1 cell array
    {[0.0144]}
    {[0.0400]}
B =  2x1 cell array
    {[6.1146]}
    {[0.2074]}
>> sum(Bu) =    6.3220
>> MSWMFunb{1}*Hbc(1)*AU(1) =    1
>> MSWMFunb{2}*Hbc(2)*AU(2) =    1.0000
>> GU{:,:} =
    1.0000    0.5774
     0             1
```

The user data rates are equal, but follow the convention of user 1 at the bottom/right of MAC and at the top/left of BC. The maximum single-user rate for this channel has been found earlier from vector coding to be 6.56 bits/symbol.

The example can be repeated with both BC users having equal energy:

$$\boldsymbol{\mathcal{E}}^{BC} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \rightarrow \begin{bmatrix} 1/2502 \\ 2501/2502 \end{bmatrix} = \boldsymbol{\mathcal{E}}^{MAC} \quad , \tag{5.380}$$

then the dual-channel's MMSE MAC design follows:

```
>> Hmac=[50 80];
>> Rxx=diag([1/2 1/2]);
>> A=sqrtm(Rxx);
>> [Bu, GU, WU, S0, MSWMFU] = mu_mac(Hmac, A, [1 1] , 2)
Bu =
    5.1444    0.9155
GU =
    1.0000    1.6000
         0    1.0000
WU =
    0.0008         0
   -0.6250    0.3909
S0 =   1.0e+03 *
    1.2510         0
         0    0.0036
MSWMFU =
    0.0283
    0.0177
>> MSWMFU*Hmac*A  =
    1.0000    1.6000
    0.6250    1.0000
>> sum(Bu)  =    6.0600
```

with direct design

```
>> Hbc=[80 ; 50];
>> AU=[sqrt(1/2502) sqrt(2501/2502)];
>> Lyu=[1 1];
cb=2;
```

```
>> [Bu, GU, S0, MSWMFunb , B] = mu_bc(Hbc, AU, Lyu , cb)
Bu =
    0.9155    5.1444
GU =  2x1 cell array
S0 =  2x1 cell array
    {[    3.5580]}
    {[1.2510e+03]}
MSWMFunb =  2x1 cell array
    {[0.6252]}
    {[0.0200]}
B =  2x1 cell array
    {[0.9155]}
    {[5.1444]}
>> sum(Bu) =    6.0600
>> MSWMFunb{1}*Hbc(1)*AU(1) =    1
>> MSWMFunb{2}*Hbc(2)*AU(2) =    1.0000
>> GU{:,:} =
    1.0000   50.0100
     0            1
```

The sum rate is reduced because the secondary component on user 2 has more energy. The individual user data rates also are both less. The BC design is somewhat trivial on this channel. The designer could create designs for the dual MAC, finding in particular energy distributions for certain data rates in the capacity region using minPMAC or admMAC, and then use the mac2BC energy transformation to find the corresponding dual BC's energies. Then mu_bc.m could be used with those energies to generate the BC design. The unbiased feedback sections differ because there is MAC receiver matrix multiply accommodates the channel scaling of both users, while for the BC the crosstalk cancellation occurs before this scaling affects both users.

Section 2.8's later worst-case-noise design for this same BC instead only finds the rate sum, because that design's user 1 is a "free-loading" secondary user for which the feedback coefficient's product with the corresponding input is always the square-root energy fraction. If $\mathcal{E}_1 = 1$, then both the index-reversed energy-sum MAC and the BC obtain 6.322 by placing all the energy on the better-gain user channel. The

The designer uses duality to find a dual-Esum-MAC's $\mathcal{E}_x$ that corresponds to a specified rate vector $\boldsymbol{b}$ (and consequent minimized weighted energy sum) or for the specified energy vector and a maximized weighted rate sum. If the point is admissible on the dual MAC, it is also admissible on the BC. The rate sum for a fixed set of $R_{\boldsymbol{xx}}(u)$ can be less on the MAC than on the Esum-MAC. However, the MAC is artificial and used to design the BC.

Example 5.5.1's key point is that input deflection does not change the bit rate $\mathcal{I}(\boldsymbol{x};\boldsymbol{y})$ nor the individual user's rates, although the corresponding energy distributions are different for the BC and its dual. The BC's maximum rate-sum corresponds to the dual MAC's order-reversed maximum rate sum (for an energy-sum MAC).

The lack of coordination at the MAC transmitters restricts the input autocorrelation matrices so that reduction in data rate relative to single user is consequent, while the order reversal tacitly imposes worst-case noise rate-sum reduction. The BC loss parameter measures this loss.

Lemma 5.5.2 [BC loss relative to single user]

$$\gamma_{BC} \stackrel{\Delta}{=} \frac{2^{2\cdot\bar{\mathcal{C}}} - 1}{2^{2\cdot\bar{\mathcal{C}}_{BC}} - 1} \leq \gamma_{e-sum,dual-MAC} = \gamma_{single-user} = 1 \; . \tag{5.381}$$

Proof: *Duality assures this result. The energy-sum MAC will use best MAC for its maximum rate-sum calculation, which may not correspond to the BC's best order.* **QED**

Again, this helpful concept of input deflection and the dual channel applies not to the overall channel $\widetilde{H}$ (where the overall duality preserved sum rate and total sum energy), but instead to the individual user data rates and channel elements in dual vector BC and vector MAC channels.

Duality Mappings: Figure 5.47 illustrates a general vector dual BC and its original vector MAC at user level. The $\widetilde{H}_u$ are the MAC columns and the $\widetilde{H}_u^*$ are the BC (order-reversed) rows. Theorem 5.5.1's translations apply directly to the overall $\widetilde{H}$ channel (both overall noises become white) so that the individual users' rates and overall rate sum of both channels are equal. However, individual $R_{\boldsymbol{xx}}$ and $R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}$ are of interest.

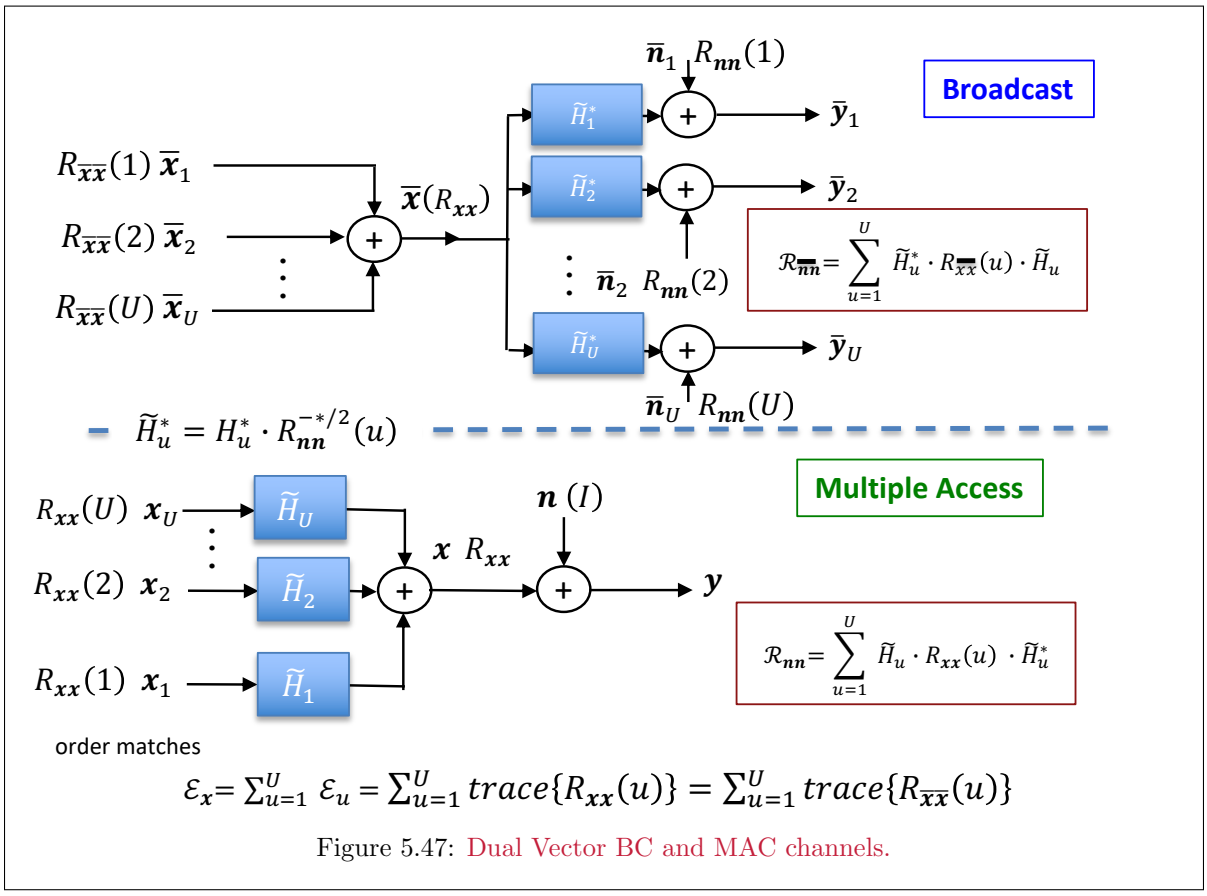


Figure 5.47: Dual Vector BC and MAC channels.

Designers can choose the individual user and input autocorrelation matrices $R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}(u)$ and $R_{\boldsymbol{xx}}(u)$ to be equal, as in (5.360) - (5.363). These two individual user channels correspond to the data rates in (5.360) and look almost like duals already, except they have different noises. As in Figure 5.46, the dual design accommodates different noises by input deflection. Each individual channel description deflects its input with equivalent noises that include crosstalk, which is not necessarily white. The deflections equate user u's individual MAC and BC data rates. That is, each equivalent-user deflects their input by the corresponding dual's square-root noise autocorrelation, where that noise autocorrelation contains the other users' noises, as in Equations (5.367) and (5.368).

Such input deflection does not change the individual user's mutual information since it is 1-to-1, so the user's rate on both dual and original channel is the same, $\forall\ u$. Returning to (5.362) and (5.363) with input deflection, noting trivially that $\mathcal{R}_{\boldsymbol{nn}}^{1/2} \cdot \mathcal{R}_{\boldsymbol{nn}}^{-1/2} = I$, the equal individual-user mutual information follows if

$$\left| \grave{H}_u \cdot \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{*/2}(u) \cdot R_{\boldsymbol{xx}}(u) \cdot \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{1/2}(u) \cdot \grave{H}_u^* + I \right| = \left| \grave{H}_u^* \cdot \mathcal{R}_{\boldsymbol{nn}}^{*/2}(u) \cdot R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}(u) \cdot \mathcal{R}_{\boldsymbol{nn}}^{1/2}(u) \cdot \grave{H}_u + I \right| \quad , \quad (5.382)$$

which uses the input-deflection mappings from (5.365) and (5.366) to undo the input deflection in the channel model $\grave{H}_u$. This deflection also absorbs the channel's other-side noise normalization, thereby setting the denominator determinant to unity.

The (again "economy-mode" SVD) of $\grave{H}_u$ is

$$\grave{H}_u = F_u \cdot \Lambda_u \cdot M_u^* \quad . \tag{5.383}$$

Equation (5.382) requires appropriate choice of $R_{\boldsymbol{xx}}$ and $R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}$. The first determinant's argument is

$$F_u \cdot \Lambda_u \cdot M_u^* \cdot \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{*/2} \cdot R_{\boldsymbol{xx}}(u) \cdot \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{1/2} \cdot M_u \cdot \Lambda_u \cdot F_u^* + I \quad . \tag{5.384}$$

The pre-multiplication and post-multiplication by F_u* and F_u respectively does not change the determinant, nor user u's mutual information. Then, if the two input autocorrelation matrices $R_{\grave{\boldsymbol{x}}\grave{\boldsymbol{x}}}$ and $R_{\grave{\tilde{\boldsymbol{x}}}\grave{\tilde{\boldsymbol{x}}}}$ satisfy

$$R_{\grave{\boldsymbol{x}}\grave{\boldsymbol{x}}}(u) = M_u \cdot F_u^* \cdot R_{\grave{\tilde{\boldsymbol{x}}}\grave{\tilde{\boldsymbol{x}}}}(u) \cdot F_u \cdot M_u^* \quad . \tag{5.385}$$

Insertion of (5.385) into (5.382), using also (5.358), equates the individual-user mutual information between vector MAC and vector BC. The autocorrelation matrices relate as (BC to MAC)

$$R_{\boldsymbol{xx}}(u) = \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{-*/2}(u) \cdot M_u \cdot F_u^* \cdot \mathcal{R}_{\boldsymbol{nn}}^{1/2}(u) \cdot R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}(u) \cdot \mathcal{R}_{\boldsymbol{nn}}^{*/2}(u) \cdot F_u \cdot M_u^* \cdot \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{-/2}(u) \quad , \tag{5.386}$$

which reverses to the MAC-to-BC relationship

$$R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}(u) = \mathcal{R}_{\boldsymbol{nn}}^{-/2}(u) \cdot F_u \cdot M_u^* \cdot \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{*/2}(u) \cdot R_{\boldsymbol{xx}}(u) \cdot \mathcal{R}_{\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}}^{1/2}(u) \cdot M_u \cdot F_u^* \cdot \mathcal{R}_{\boldsymbol{nn}}^{-*/2}(u) \quad . \tag{5.387}$$

The relationships hold for each and every u. Recursive use of Equation 5.386 must start with user U and successively pass to $U-1, ..., 1$ because each successive $\mathcal{R}_{\boldsymbol{nn}}(u)$ in (5.358) depends on higher indices of $R_{\grave{\boldsymbol{x}}\grave{\boldsymbol{x}}}(i > u)$. Equation (5.387) must start with user 1 and pass succesively to $2, ..., U$ because each successive $R_{\grave{\tilde{\boldsymbol{x}}}\grave{\tilde{\boldsymbol{x}}}}(u)$ in (5.359) depends on lower indices of $R_{\grave{\tilde{\boldsymbol{x}}}\grave{\tilde{\boldsymbol{x}}}}(i < u)$.

The dual relationships correspond to a recursion that Figures 5.48's and 5.49's flow charts respectively illustrate. For MAC to BC, $R_{\grave{\tilde{\boldsymbol{x}}}\grave{\tilde{\boldsymbol{x}}}}(u)$ forms recursively after each successive duality step for the next user. The overall channels (viewed from single-user perspective) are still duals and have the same rate sum. The energies of the individual users are not the same because of the input deflection that makes individual vector BC and vector MAC users' data rates, b_u, equal, or equivalently these individual users' energies were never the same in the original overall channel, just their sums were equal. While individual users and rate sums are equal, duals need not otherwise be optimum in any sense, unless the specific rate $\boldsymbol{b}_0$ is on the capacity region's border $\mathcal{S}(\boldsymbol{b})$

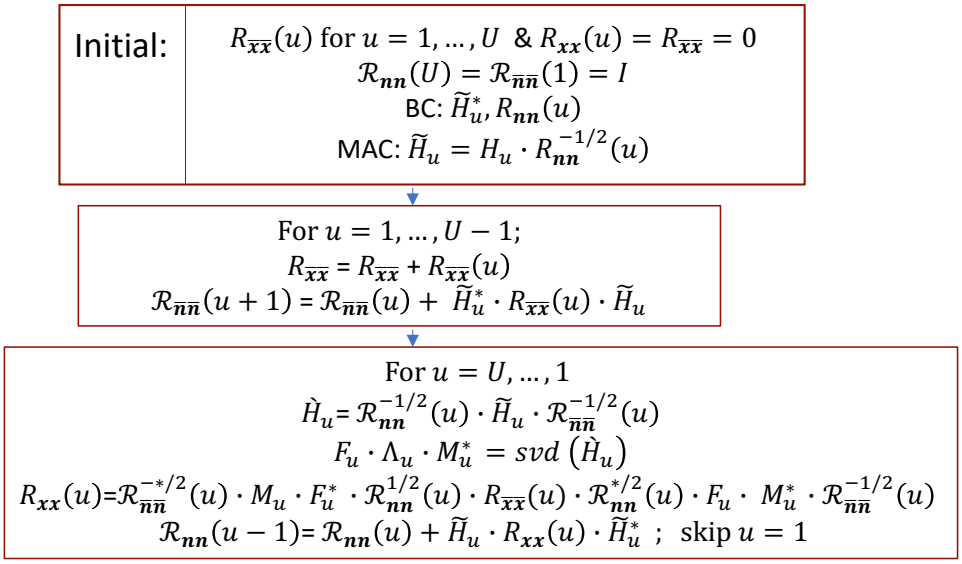


Figure 5.48: Duality flow chart for BC to MAC.

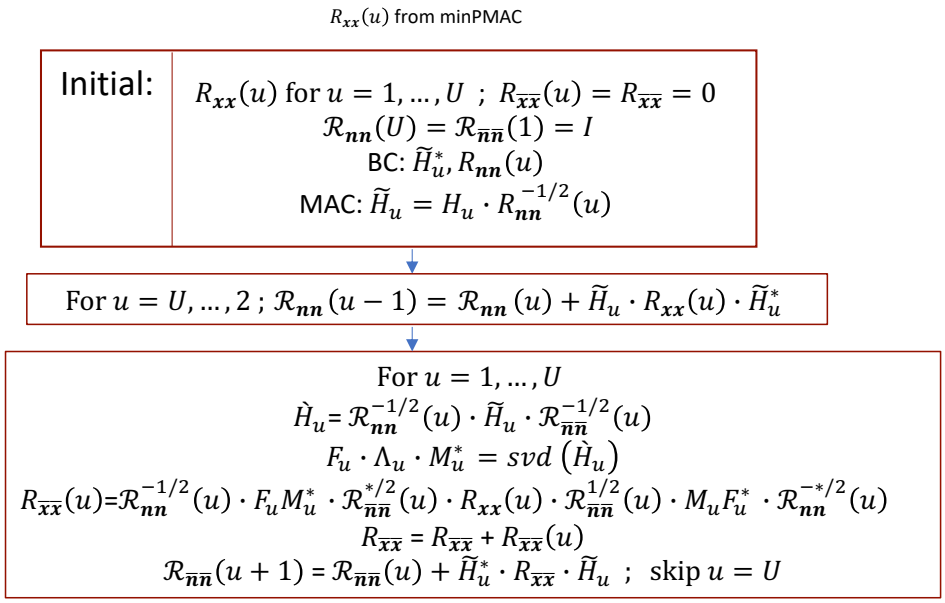


Figure 5.49: Duality flow chart for MAC to BC.

mac2bc Program: The mac2bc program computes dual autocorrelation matrices. The 2 inputs are:

1. The input Rxxm $\{R_{\boldsymbol{x}\boldsymbol{x}}(u)\}$ is an $L_x \times L_x \times U$ array of autocorrelation-matrices. The $L_x = L_{x,u}$ is constant for this program. The designer can attempt variable $L_{x,u}$ by enlarging this input (and $\widetilde{H}_u$ also below) to have the maximum number $L_x = \max_u L_{x,u}$ via adding zero columns (and rows for Rxxm) to users with less antennas.

2. The input Hmac ($\{\widetilde{H}_u\}$), is the set of $L_y \times L_x$ MAC user submatrices specified in an $L_y \times L_x \times U$ array.

The output Rxxb is the $L_y \times L_y \times U$ array of BC autocorrelation component matrices $\{R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}(u)\}$. These may be added over the users to produce the aggregate vector BC transmit autocorrelation $R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}$.

```
>> help mac2bc
function Rxxb = mac2bc(Rxxm, Hmac, N)
  This function converts U users' MAC autcorrelation matrices to dual BC
  autocorrelation matrices. These output matrices achieve the same set of
  rates in the dual BC by a lossless precoder. The user should order the
  inputs Rxxm and Hmac (see below) so that they have the desired order
  already.
  Inputs
  Rxxm - an Lx x Lx x U array of users autocorrelation matrices
         corresponding to [U:-1:1]=order
  Hmac - the Ly x Lx x U MAC channel for with users listed U...1, from left
  Output
  Rxxb  - Ly  Ly x U dual BC's output autocorrelation matrix, which has the
          left user in order reversed (order *J) at top/left in Rxxb.
```

EXAMPLE 5.5.2 (Return to simple BC channel) Chapter 2's scalar BC had $\widetilde{H}_{bc} = [80 \;\; 50]^*$, so with order of $u = 1$ at the top/left, then $\widetilde{H}_{bc,1} = 80$ and $\widetilde{H}_{bc_2} = 50$. The MAC has user $u = 2$ at the top left. Thus, $\widetilde{H}_{MAC} = [50 \;\; 80]$ so user 2 has gain 50 on both the BC and its dual MAC in this labelling convention. Thus BC has one unit of input energy, $\mathcal{E}_{\boldsymbol{x}} = 1$. Returning to Example 5.5.1's first-instance

energy example, the data rates were

$$b_{2,MAC} = \frac{1}{2}\cdot\log_2\left(1+50^2\cdot\frac{1}{7504}\right) = .2074 \tag{5.388}$$

$$b_{1,MAC} = \frac{1}{2}\cdot\log_2\left(1+80^2\cdot\frac{1/2}{1+50^2\cdot 1/2}\right) = 6.116 \tag{5.389}$$

$$b_{sum,MAC} = b_{2,MAC}+b_{1,MAC} = 6.322 = 0.5*\log_2|\boldsymbol{R_{yy}}| \tag{5.390}$$

Using the mac2bc.m program

```
Rxxm=zeros(1,1,2);
Rxxm(1,1,1)=1/7504;
Rxxm(1,1,2)=7503/7504;
Hmac=zeros(1,1,2);
Hmac(1,1,1)=50;
Hmac(1,1,2)=80;
Rxxb=mac2bc(Rxxm,Hmac)
Rxxb(:,:,1) =    0.7500 (3/4 at top of BC)
Rxxb(:,:,2) =    0.2500 (1/4 at bottom of BC)
```

Correspondingly the two BC rates are then

$$b_{1,BC} = \frac{1}{2}\cdot\log_2\left(1+80^2\cdot(3/4)\right) = 6.1146 \tag{5.391}$$

$$b_{2,BC} = \frac{1}{2}\cdot\log_2\left(1+50^2\cdot(1/4)/(1+50^2\cdot(3/4))\right) = .2074 \tag{5.392}$$

$$b_{sum,BC} = b_{2,BC}+b_{1,BC} = 6.322 \tag{5.393}$$

The nominal direct mutual information for the BC must use worst-case noise to whiten the determinant, and the designer can check this is the rate sum in that case:

```
>> [Rwcn , bsum]=wcnoise(1, [80 ; 50], 1)
Rwcn =
    1.0000    0.6250
    0.6250    1.0000
bsum =    6.3220
```

More simply on the second energy-set in Example 5.5.1, the process of dual users' energy calculation is

```
Rxxm(1,1,1)=1/2;
Rxxm(1,1,2)=1/2;
Rxxb=mac2bc(Rxxm,Hmac)
Rxxb(:,:,1) =   3.9968e-04 (1/2502)
Rxxb(:,:,2) =    0.9996  (2501/2502)
```

bc2mac Program: The bc2mac program reversely computes the dual MAC's autocorrelation matrix set $\{R_{\boldsymbol{xx}}(u)\}$ from a broadcast channel $\tilde{H}_{dual}$ and the BC autocorrelation matrix set $\{R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}(u)\}$.

```
>> help bc2mac
 function Rxxm = bc2mac(Rxxb, Hmac)
  This function converts the BC autocorrelation matrices set in Rxxb to the
  corresponding set of autocorrelation matrices to Rxxm of a dual MAC.
  The command Hbc=conj(permute(Hmac(:,:,end:-1:1),[2 1 3])), where
  generates the Hbc if needed for any reason.
  U = number of users, which is obtained from the 3rd dimension of the input
  tensor Hmac.  Lx and Ly correspond to this (dual) MAC that has Ly receive
  antennas and Lx transmit antennas per user.  Variable Lxu (so Lyu on Rxxb)
  needs to find maximum over u, and extend all by zero columns in Hmac and
  zeroed rows and columns in Rxxb.
  The output Rxxm user order is reversed restored) by this program.
  Inputs
  Rxxb is an Ly by Ly by U array containing the BC input Rxx(u) matrices.
    where Rxxb(:,:,u) is the autocorrelation matrix for BC user u.
    The overall autocorrelation matrix Rxxbsum is sum(Rxxb,3), but not
    output.
  Hmac is an Ly x Lx x U MAC channel matrix that for which the dual BC is
  computed internally. Hmacu(:,:,u) is user u's noise-whitened
    equivalent channel matrix.
  Output
  Rxxm contains the MAC autocorrelation matrices, which are Ly x Ly x U.
     Rxxm(:,:,u) is user u's aucorrelation matrix, reversed in order from
     the meaning of u in the input Rxxb.
```

[Reverse the mac2bc in Example 5.5.2] Continuing with the above Rxxb output

```
>> bc2mac(Rxxb,Hmac)
ans(:,:,1) =    0.5000
ans(:,:,2) =    0.5000    (checks)
```

or return to the first intance

```
Rxxm=zeros(1,1,2);
Rxxm(1,1,1)=1/7504;
Rxxm(1,1,2)=7503/7504;
Hmac=zeros(1,1,2);
Hmac(1,1,1)=50;
Hmac(1,1,2)=80;
Rxxb=mac2bc(Rxxm,Hmac);
bc2mac(Rxxb, Hmac)
ans(:,:,1) =   1.3326e-04 % 1/7504
ans(:,:,2) =    0.9999 % 7503/7504
Checks
```

Thus the bc2mac program reverses both the examples for the first order above.

As an example o a channel with 2 antennas/user

```
Rxxm=zeros(2,2,2);
Rxxm(:,:,1)=eye(2);
Rxxm(:,:,2)=[2 1
1 2]
Hmac=zeros(2,2,2);
Hmac(:,:,1)=[80 70
50 60];
Hmac(:,:,2)=[80 -50
40 -25]
Rxxb=mac2bc(Rxxm,Hmac)
Rxxb(:,:,1) =
    0.0055   -0.0080
   -0.0080    0.0121
Rxxb(:,:,2) =
    2.0391    0.2984
    0.2984    3.9433
 >> bc2mac(Rxxb, Hmac)
ans(:,:,1) =
    1.0000    0.0000
    0.0000    1.0000
ans(:,:,2) =
    2.0000    1.0000
    1.0000    2.0000  (checks)
```

Now progressing to the more sophisticated example with 3 users:

EXAMPLE 5.5.3 *[3-user 64-tone channel]* Continuing with Example 5.4.6, which has 3 users as lowpass/bandpass/highpass channel with $\bar{N} = 64$, the corresponding dual for the equal-energy-input MAC follows as:

```
N=64;
nu=3;
h=cat(3,[1 0 .8 ; 0 1 1],[.9 -.3 0 ; .5 -1 -1],[0 .2 0 ; .4 -.63 0],[0 0 0 ; 0 .648 0])*10;
H = fft(h, N, 3);
Hbc=zeros(3,2,N);
Rxxm=zeros(1,1,3);
Rxxm(1,1,:)= N/(N+nu)*[1 1 1];
Rxxb=zeros(2,2,3,N);
bbc=zeros(3,N);
Hbc=zeros(3,2,N);
for n=1:N
Rxxb(:,:,:,n)=mac2bc(Rxxm, reshape(H(:,:,n),2,1,3));
Hbc(:,:,n)=H(:,end:-1:1,n)';
bbc(1,n)=real(log2(1+Hbc(1,:,n)*Rxxb(:,:,1,n)*Hbc(1,:,n)'));
bbc(2,n)=real(log2((1+Hbc(2,:,n)*(Rxxb(:,:,2,n)+Rxxb(:,:,1,n))*Hbc(2,:,n)')/(1+Hbc(2,:,n)*Rxxb(:,:,1,n)*Hbc(2,:,n)')))
bbc(3,n)=real(log2((1+Hbc(3,:,n)*(Rxxb(:,:,3,n)+Rxxb(:,:,2,n)+Rxxb(:,:,1,n))*Hbc(3,:,n)')/(1+Hbc(3,:,n)*(Rxxb(:,:,2,n)
end
bvec=sum(bbc') =  132.7477  412.8794  445.1264
 bsum=sum(bvec) =  990.7535
```

The BC places user 3 at the right/bottom, corresponding to the MAC's placement of user 3 at the top/left, thus explaining why the order above appears reversed. User 3 on the MAC is in the best position (with other users' crosstalk removed), while it is in the worst position (having other users' crosstalk as noise) in the BC.

5.5.2.1 Determination of $R_{\boldsymbol{xx}}$ for given rate tuple

The dual vector MAC for a vector BC allows determination of a rate vector $\boldsymbol{b}$'s feasibility. This rate tuple can be input, along with $\widetilde{H}$, and the weight vector $\boldsymbol{w} = [111...1]$ to minPMAC. The resultant output energy should be summed and compared with the total energy constraint. If the total energy constraint is not exceeded, then the point is feasible (in the capacity region) and a GDFE can be designed for the set of user input energies on the dual vector MAC.

5.5.3 Vector DMT and the vector BC

Section 5.5.1's Figure 5.27 illustrates Vector BC broadcast transmitter DMT symbols' synchronization. Vector BC transmit-symbol alignment is easier than for the vector MAC. Each transmitter has the same size DMT symbol and aligns symbol boundaries. However, often systems are bi-directional, so both cyclic prefixes and cyclic suffixes can be used to align both directions of transmission for a downstream (or down-link) or an upstream (up-link) system so that all symbols at the common hubs of multiple-access receivers and broadcast transmitters respectively align. Section 4.7.6 also discusses the Zipper method by Isaksson that can be used when a downstream vector BC or an upstream vector MAC share the same users in a bi-directional multiuser channel. Such alignment also ensures that interference from "down back into up" can be easily cancelled at the hub with a single complex coefficient per tone. Such cancellers are often called "NEXT" cancelers, and are not the same as cancellers that exploit spatial correlation.[85]

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 the $\grave{H}_u$, which means

$$\nu T' \geq \max_{u,i} \left\{ \text{length} \left(\tilde{\tilde{h}}_{u,i}(t) + (\Delta_u) \right) \right\} \ ,$$

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.

[85] These up-into-down NEXT cancellation must use spatial correlation with multiple antennas. NEXT cancellers remove any effect from the broadcast signal downlink into the received uplink multiple-access signals.

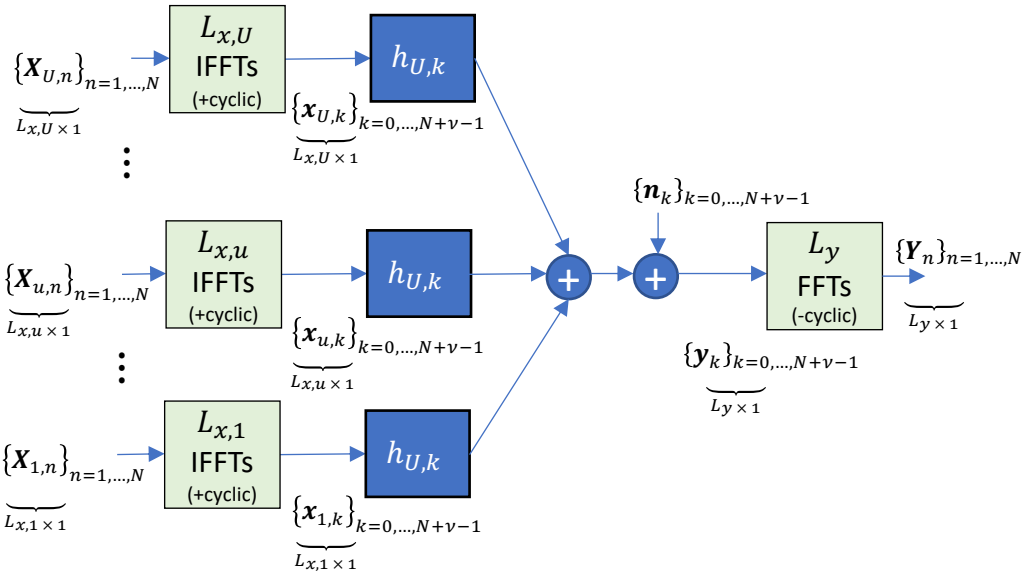


Figure 5.50: **Vector DMT vector BC system.**

Each tone of the $\mathscr{L}_y$ receivers' FFT outputs can then be modeled as

$$\underbrace{\boldsymbol{Y}_n}_{\mathscr{L}_y\times 1} = \underbrace{H_n}_{\mathscr{L}_y\times L_x}\cdot\underbrace{\boldsymbol{X}_n}_{L_x\times 1} + \underbrace{\boldsymbol{N}_n}_{\mathscr{L}_y\times 1} \quad , \tag{5.394}$$

where

$$H_n = \begin{bmatrix} H_{1,n} \\ \vdots \\ H_{U,n} \end{bmatrix} \tag{5.395}$$

$$\boldsymbol{X}_n = \begin{bmatrix} x_{1,n} \\ \vdots \\ x_{L_x,n} \end{bmatrix} = \sum_{u=1}^{U} \boldsymbol{X}_{u,n} \tag{5.396}$$

$$\boldsymbol{Y}_n = \begin{bmatrix} \boldsymbol{Y}_{1,n} \\ \vdots \\ \boldsymbol{Y}_{U,n} \end{bmatrix} \tag{5.397}$$

$$\boldsymbol{Y}_{u,n} = \begin{bmatrix} Y_{u,1,n} \\ \vdots \\ Y_{u,\mathscr{L}_y,n} \end{bmatrix} \tag{5.398}$$

$$= \grave{H}_{u,n}\cdot\boldsymbol{V}_n + \boldsymbol{N}_{u,n} \tag{5.399}$$

where

$$\grave{H}_{u,n} = R_{\boldsymbol{NN}}^{-1/2}(n)\cdot H_{u,n}\cdot[A_{1,n}\ A_{2,n}\ \ldots\ A_{U,n}] \tag{5.400}$$

and $\boldsymbol{V}_n$ is the input vector of unit(identity)-energy user inputs on tone n, with user 1 at the top to maintain vector BC duality-motivated order-reversal. The quantities $A_{u,n}$ simply form the individual-user autocorrelation matrix components (or energy scalings in the scalar case). The $(l_y, l_x)^{th}$ entry of $H_{u,n}$ is the DFT of the response from line/antenna l_x to line/antenna l_y of user u's output. The energy constraint becomes

$$\sum_n \text{trace}\,\{R_{\boldsymbol{XX}}(n)\} \le \mathcal{E}_x \quad . \tag{5.401}$$

The input autocorrelation on tone n is

$$R_{\boldsymbol{XX}}(n) = \mathbb{E}\left[\boldsymbol{X}_n \cdot \boldsymbol{X}_n^*\right] = \sum_{u=1}^{U} R_{\boldsymbol{XX}}(u,n) \quad . \tag{5.402}$$

Again, the tone-indexed model for DMT leads to tremendous computational reduction with respect to the full precoding (or GDFE) structure. Effectively, $\overline{N}$ smaller channels of size $\mathscr{L}_y \times \mathscr{L}_x$ replace a giant channel of size $\mathscr{L}_y \cdot \overline{N} \times \overline{N} \cdot \mathscr{L}_x$. The GDFE/precoder computational advantage when $L_x = U$ and $L_y = 1$ is a complexity of $U \cdot \overline{N} \cdot log_2(\overline{N}) + \overline{N} \cdot U^2$ versus the much larger $(\overline{N} \cdot U)^2$, or if $\overline{N} = 128$ and $U = 4$, the savings is a factor of about 50 times less computation (262,144 vs 5,632). Figure 5.50 is the result of the modeling.

The input for each tone then decomposes as

$$R_{\boldsymbol{XX}}(n) = \sum_{u=1}^{U} R_{\boldsymbol{XX}}(u,n) = \sum_{u=1}^{U} A_{u,n} \cdot R_{\boldsymbol{VV}}(u,n) \cdot A_{u,n}^* \quad . \tag{5.403}$$

Receiver u's noise (on any tone) can be found then as

$$\mathcal{R}_{\boldsymbol{nn}}(u,n) = R_{\boldsymbol{NN}}(u,n) + \sum_{i=u+1}^{U} \widetilde{H}_{u,n} \cdot R_{\boldsymbol{XX}}(i,n) \cdot \widetilde{H}_{u,n}^* \quad . \tag{5.404}$$

The noise-equivalent channel is then

$$\grave{H}_{u,n} = \underbrace{\mathcal{R}_{\boldsymbol{nn}}^{-1/2}(u,n)}_{\mathscr{L}_y \times \mathscr{L}_y} \cdot \underbrace{\widetilde{H}_{u,n}}_{\mathscr{L}_y \times L_x} \cdot \underbrace{A_{u,n}}_{L_x \times \rho_{u,n}} \quad . \tag{5.405}$$

More directly, user u's GDFE receiver estimates user u's signal by removing users' $i = u+1, ..., U$ crosstalk. Then a series of such designs on each tone is done for $u = 1, ..., U$ and the relevent rows of each feedback section $G_{unb,u,n}$ form the precoder for each tone with known $R_{\boldsymbol{XX}}(u,n)$.

5.5.4 BC GDFE Design

Duality enables BC design because duality specifies all the user input autocorrelation matrices and not just their sum. The following 2-user example illustrates a full BC design using duality for a complex=baseband vector DMT system with $\overline{N} = 8$ tones.

EXAMPLE 5.5.4 *[2-user channel with memory]* Figure 5.51 illustrates a complex baseband channel. Initially, this channel will be single-user MIMO, before later becoming the dual of a BC channel.

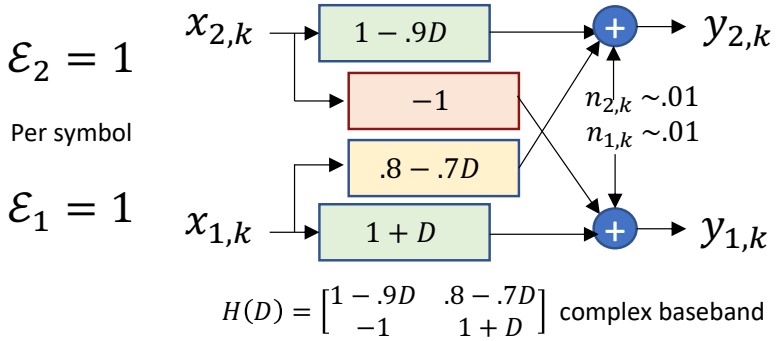


Figure 5.51: 2-user channel with memory for Example 5.51.

Figure 5.51 shows this channel's MIMO D-transform elements. This example uses a vector DMT size of $\overline{N} = 8$ to simplify results' display, The reader may want to increase $\overline{N}$ to investigate sufficiently large value to approximate infinite multi-tone performance. Initially the gap will be $\Gamma = 0$ dB for both users' codes. Energies and data rates should be interpreted then as used over 9 dimensions since $\nu = 1$ for this channel. The matlab commands to find the frequency-domain equivalent are (with noise whitening to unit variance per complex dimension):

```
h = cat(3, [1   .8 ;  -1    1], [-.9   -.7 ;  0   1]  )*10;
H = fft(h, 8, 3)
H(:,:,1) =
   1.0000 + 0.0000i   1.0000 + 0.0000i
 -10.0000 + 0.0000i  20.0000 + 0.0000i
H(:,:,2) =
   3.6360 + 6.3640i   3.0503 + 4.9497i
 -10.0000 + 0.0000i  17.0711 - 7.0711i
H(:,:,3) =
  10.0000 + 9.0000i   8.0000 + 7.0000i
 -10.0000 + 0.0000i  10.0000 -10.0000i
H(:,:,4) =
  16.3640 + 6.3640i  12.9497 + 4.9497i
 -10.0000 + 0.0000i   2.9289 - 7.0711i
H(:,:,5) =
  19.0000 + 0.0000i  15.0000 + 0.0000i
 -10.0000 + 0.0000i   0.0000 + 0.0000i
H(:,:,6) =
  16.3640 - 6.3640i  12.9497 - 4.9497i
 -10.0000 + 0.0000i   2.9289 + 7.0711i
H(:,:,7) =
  10.0000 - 9.0000i   8.0000 - 7.0000i
 -10.0000 + 0.0000i  10.0000 +10.0000i
H(:,:,8) =
   3.6360 - 6.3640i   3.0503 - 4.9497i
 -10.0000 + 0.0000i  17.0711 + 7.0711i
```

Every tone is a complex AWGN with the gains above.

This channel exhibits conjugate symmetry about the Nyquist (middle) frequency at index 5, but it is complex baseband. The conjugate symmetry in this case reflects that the time-domain channel has purely real coefficients. Single user design allows both transmit and receive coordination and is an upper bound on results (using vector coding with 16 real dimensions). The total energy over 16 real dimensions is then 16 units times (8/9) to account for the cyclic-prefix loss of one dimension's energy:

```
>> Hblock=blkdiag(H(:,:,1),H(:,:,2),H(:,:,3),H(:,:,4),H(:,:,5),H(:,:,6),H(:,:,7),H(:,:,8));
g=svd(Hblock);
gains=(g.*g)'
gains =
  651.4623  567.9619  567.9619  500.2007  447.4561  447.4561  405.2081  405.2081
  188.7919  188.7919   91.0919   91.0919   81.4901   81.4901   34.5377    1.7993
  >> En = waterfill(128/9, gains', 1);
>> En' =
    0.9285    0.9283    0.9283    0.9280    0.9278    0.9278    0.9276    0.9276
    0.9247    0.9247    0.9191    0.9191    0.9178    0.9178    0.9011    0.3743
>> bvec=log2(ones(16,1)+En.*gains')' =
    9.2429    9.0450    9.0450    8.8617    8.7010    8.7010    8.5579    8.5579
    7.4560    7.4560    6.4046    6.4046    6.2439    6.2439    5.0055    0.7428
>>  sumrate  = sum(bvec) =  116.6695
>> sumrate/9 =   12.9633
```

So the maximum number of bits/complex-dimension is roughly 13. This will upper bound the sum of the two users' rates when viewed as either a MAC or a BC.

Using equal energy on all tones should have little loss because the single-user optimum is close to such use (except one dimension). Thus, a reasonable MAC energy design might do so:

```
>> GU=zeros(2,2,8);
WU=zeros(2,2,8);
SO=zeros(2,2,8);
Bu=zeros(2,8);
```

```
MSWMFU=zeros(2,2,8);
AU=zeros(2,2,8);
for n=1:8  AU(:,:,n)=(sqrt(8)/3)*eye(2);  end
>> for n=1:8
[Bu(:,n), GU(:,:,n) , WU(:,:,n),SO(:,:,n), MSWMFU(:,:,n)] = mu_mac(H(:,:,n), AU(:,:,n), [1 1], 1);
end
bvec=sum(Bu')
Bsum =sum(bvec)    =   62.3515   54.1393
Bsum =  116.4908
Bsum/9 = 12.9434
>> Bu =
    6.5043    7.1048    7.9703    8.5075    8.6822    8.5075    7.9703    7.1048
    3.6736    7.7329    7.9256    6.8322    5.4843    6.8322    7.9256    7.7329
>> sum(Bu) =
   10.1778   14.8377   15.8959   15.3396   14.1665   15.3396   15.8959   14.8377
```

so the MAC has a rate sum that is slightly less than the single-user best for the equal-energy inputs. There is less information at DC as one of the users is high-pass.

The receiver has two matrix operations per tone, with the first being the combined MSWMFU and the second being the feedback section:

```
>> MSWMFU
MSWMFU(:,:,1) =
   0.0105 + 0.0000i  -0.1050 + 0.0000i
   0.2364 + 0.0000i   0.0412 + 0.0000i
MSWMFU(:,:,2) =
   0.0251 - 0.0439i  -0.0690 - 0.0000i
   0.0397 - 0.0382i   0.0392 + 0.0116i
MSWMFU(:,:,3) =
   0.0377 - 0.0340i  -0.0377 + 0.0000i
   0.0374 - 0.0084i   0.0449 + 0.0254i
MSWMFU(:,:,4) =
   0.0425 - 0.0165i  -0.0260 + 0.0000i
   0.0456 + 0.0096i   0.0681 + 0.0449i
MSWMFU(:,:,5) =
   0.0437 + 0.0000i  -0.0230 + 0.0000i
   0.0707 + 0.0000i   0.1329 + 0.0000i
MSWMFU(:,:,6) =
   0.0425 + 0.0165i  -0.0260 + 0.0000i
   0.0456 - 0.0096i   0.0681 - 0.0449i
MSWMFU(:,:,7) =
   0.0377 + 0.0340i  -0.0377 + 0.0000i
   0.0374 + 0.0084i   0.0449 - 0.0254i
MSWMFU(:,:,8) =
   0.0251 + 0.0439i  -0.0690 + 0.0000i
   0.0397 + 0.0382i   0.0392 - 0.0116i
>> GU
GU(:,:,1) =
   1.0000 + 0.0000i  -1.9703 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,2) =
   1.0000 + 0.0000i  -0.8335 + 0.4508i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,3) =
   1.0000 + 0.0000i   0.1530 + 0.3488i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,4) =
   1.0000 + 0.0000i   0.5244 + 0.1697i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,5) =
   1.0000 + 0.0000i   0.6182 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,6) =
   1.0000 + 0.0000i   0.5244 - 0.1697i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,7) =
   1.0000 + 0.0000i   0.1530 - 0.3488i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,8) =
   1.0000 + 0.0000i  -0.8335 - 0.4508i
   0.0000 + 0.0000i   1.0000 + 0.0000i
```

The MAC design assumes user 2 at the top of the order: this example repeats with the users' order reversed:

```
>> J=hankel([0 1]);
```

```
for n=1:8
Hdual(:,:,n)=J*H(:,:,n)*J;
end
for n=1:8
[Bu(:,n), GU(:,:,n) , WU(:,:,n),SO(:,:,n), MSWMF(:,:,n)] = mu_mac(Hdual(:,:,n), AU(:,:,n), [1 1], 1);
end
Bu =
    8.4816    8.3860    8.1253    7.8068    7.6511    7.8068    8.1253    8.3860
    1.6963    6.4517    7.7706    7.5328    6.5155    7.5328    7.7706    6.4517
sum(Bu) =
   10.1778   14.8377   15.8959   15.3396   14.1665   15.3396   15.8959   14.8377
bvec=sum(Bu')=   64.7687   51.7220
bsum=sum(bvec) =  116.4908
```

has different individual user rates, but the same rate sum.

SWF finds the highest MAC sum rate with an energy-vector contraint while macmax finds the maximum sum rate with a single sum-energy constraint:

```
>> Rnn=zeros(2,2,8);
for n=1:8
Rnn(:,:,n)=eye(2);
end
[Rxx, bsum , bsum_lin] = SWF((8/9)*[1 1], H, [1 1], Rnn, 1)
Rxx(:,:,1) =
    0.5500         0
         0    0.8401
Rxx(:,:,2) =
    0.9339         0
         0    0.8991
Rxx(:,:,3) =
    0.9401         0
         0    0.8996
Rxx(:,:,4) =
    0.9394         0
         0    0.8954
Rxx(:,:,5) =
    0.9344         0
         0    0.8829
Rxx(:,:,6) =
    0.9394         0
         0    0.8954
Rxx(:,:,7) =
    0.9401         0
         0    0.8996
Rxx(:,:,8) =
    0.9339         0
         0    0.8991
bsum =  116.5835
bsum_lin =  106.1991
>> gamma_mac = 10*log10((2^(116.5835/9) -1)/(2^(116.4908/9)-1))  =     0.0310 dB
>> linearloss = 10*log10( (2^(116.4908/9)-1) / (2^(106.1991)-1) )  =    3.4430 dB
```

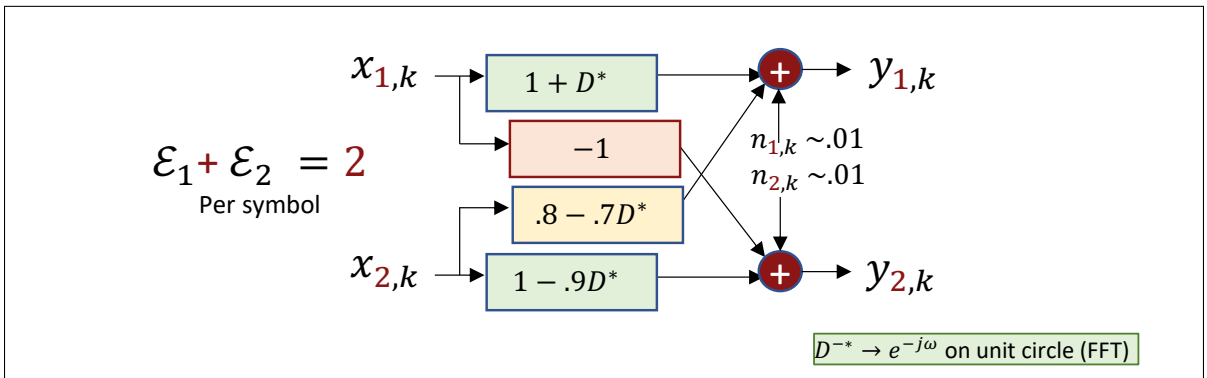


Figure 5.52: 2-user channel as dual BC for Example 5.51.

So, the MAC loss with respect to single user is very small, but the feedback sections are important and lead to 3.4 dB better performance that linear alone.

With this design for the MAC, its dual BC appears in Figure 5.52. The dual channel exists for each tone:

```
Hbc=zeros(2,2,8);
for n=1:8   % order reversal on one side, bc2mac.m does other side
Hbc(:,:,n)=H(:,end:-1:1,n)';
end
Hbc
Hbc(:,:,1) =
   1.0000 + 0.0000i  20.0000 + 0.0000i
   1.0000 + 0.0000i -10.0000 + 0.0000i
Hbc(:,:,2) =
   3.0503 - 4.9497i  17.0711 + 7.0711i
   3.6360 - 6.3640i -10.0000 + 0.0000i
Hbc(:,:,3) =
   8.0000 - 7.0000i  10.0000 +10.0000i
  10.0000 - 9.0000i -10.0000 + 0.0000i
Hbc(:,:,4) =
  12.9497 - 4.9497i   2.9289 + 7.0711i
  16.3640 - 6.3640i -10.0000 + 0.0000i
Hbc(:,:,5) =
  15.0000 + 0.0000i   0.0000 + 0.0000i
  19.0000 + 0.0000i -10.0000 + 0.0000i
Hbc(:,:,6) =
  12.9497 + 4.9497i   2.9289 - 7.0711i
  16.3640 + 6.3640i -10.0000 + 0.0000i
Hbc(:,:,7) =
   8.0000 + 7.0000i  10.0000 -10.0000i
  10.0000 + 9.0000i -10.0000 + 0.0000i
Hbc(:,:,8) =
   3.0503 + 4.9497i  17.0711 - 7.0711i
   3.6360 + 6.3640i -10.0000 + 0.0000i
>> for n=1:8
rho(n)=rank(H(:,:,n));
end
rho =     2     2     2     2     2     2     2     2
```

This channel has full rank on each tone, which means that the worst-case noise program could be used to determine sum rate on each tone without concern for singularity. However, the current BC design requires the MAC energies as input to mac2bc:

```
for n=1:8
Rxxb(:,:,:,n)=mac2bc(Rxxm, reshape(H(:,:,n),2,1,2));
Hbc(:,:,n)=H(:,end:-1:1,n)';
bbc(1,n)=real(log2(1+Hbc(1,:,n)*Rxxb(:,:,1,n)*Hbc(1,:,n)'));
bbc(2,n)=real(log2((1+Hbc(2,:,n)*(Rxxb(:,:,2,n)+Rxxb(:,:,1,n))*Hbc(2,:,n)')/(1+Hbc(2,:,n)*Rxxb(:,:,1,n)*Hbc(2,:,n)')))
end
bbc =
    3.6736    7.7329    7.9256    6.8322    5.4843    6.8322    7.9256    7.7329
    6.5043    7.1048    7.9703    8.5075    8.6822    8.5075    7.9703    7.1048
bvec =
   54.1393   62.3515
bsum =  116.4908
bvec=sum(bbc')
bsum=sum(bvec)
```

The data rates check with the dual MAC, with the order reversed in duality as the BC has user 1 at the top. The BC individual user autocorrelation matrices are:

```
>> Rxxb
Rxxb(:,:,1,1) =
   0.5841 + 0.0000i   0.1018 + 0.0000i
   0.1018 + 0.0000i   0.0178 + 0.0000i
Rxxb(:,:,2,1) =
   0.0116 + 0.0000i  -0.1164 + 0.0000i
  -0.1164 + 0.0000i   1.1642 + 0.0000i
Rxxb(:,:,1,2) =
   0.5712 - 0.0000i   0.2092 + 0.3682i
   0.2092 - 0.3682i   0.3139 + 0.0000i
Rxxb(:,:,2,2) =
   0.3119 - 0.0000i  -0.2111 - 0.3695i
  -0.2111 + 0.3695i   0.5807 + 0.0000i
```

```
Rxxb(:,:,1,3) =
   0.3160 - 0.0000i   0.3152 + 0.2855i
   0.3152 - 0.2855i   0.5723 + 0.0000i
Rxxb(:,:,2,3) =
   0.5729 - 0.0000i  -0.3165 - 0.2849i
  -0.3165 + 0.2849i   0.3165 + 0.0000i
Rxxb(:,:,1,4) =
   0.2184 - 0.0000i   0.3553 + 0.1402i
   0.3553 - 0.1402i   0.6681 + 0.0000i
Rxxb(:,:,2,4) =
   0.6730 - 0.0000i  -0.3572 - 0.1389i
  -0.3572 + 0.1389i   0.2183 + 0.0000i
Rxxb(:,:,1,5) =
   0.1945 + 0.0000i   0.3655 + 0.0000i
   0.3655 + 0.0000i   0.6867 + 0.0000i
Rxxb(:,:,2,5) =
   0.7021 + 0.0000i  -0.3695 + 0.0000i
  -0.3695 + 0.0000i   0.1945 + 0.0000i
Rxxb(:,:,1,6) =
   0.2184 + 0.0000i   0.3553 - 0.1402i
   0.3553 + 0.1402i   0.6681 - 0.0000i
Rxxb(:,:,2,6) =
   0.6730 + 0.0000i  -0.3572 + 0.1389i
  -0.3572 - 0.1389i   0.2183 - 0.0000i
Rxxb(:,:,1,7) =
   0.3160 + 0.0000i   0.3152 - 0.2855i
   0.3152 + 0.2855i   0.5723 - 0.0000i
Rxxb(:,:,2,7) =
   0.5729 + 0.0000i  -0.3165 + 0.2849i
  -0.3165 - 0.2849i   0.3165 - 0.0000i
Rxxb(:,:,1,8) =
   0.5712 + 0.0000i   0.2092 - 0.3682i
   0.2092 + 0.3682i   0.3139 - 0.0000i
Rxxb(:,:,2,8) =
   0.3119 + 0.0000i  -0.2111 + 0.3695i
  -0.2111 - 0.3695i   0.5807 - 0.0000i
```

Proceding with the worst-case-noise program on the sum BC-input autocorrelation matrix (which produces results per real dimension, so double outputs for this complex baseband channel):

```
Rxxbsum=zeros(2,2,8);
  for n=1:8
     Rxxbsum(:,:,n)=Rxxb(:,:,1,n)+Rxxb(:,:,2,n);
   end
Rwcn=zeros(2,2,8);
sumRate=zeros(1,8);
for n=1:8
[Rwcn(:,:,n) sumRate(n)]=wcnoise(Rxxbsum(:,:,n),Hbc(:,:,n),1);
end
sumRate=2*real(sumRate) =
   10.2036   14.8377   15.8959   15.3396   14.1665   15.3396   15.8959   14.8377
sum(sumRate) = 116.5166 > 116.4908
loss-to-single user = 10*log10( (2^(116.6695 /9)-1) / (2^(116.4908 /9)-1) )  =.06 dB
>> 2*bmax = 116.5892 > 116.4908 , but of course less than 116.6695
```

which is a higher data rate, but still less than the single-user maximum.

The program mu_bc now completes the design for the dual BC:

```
A=zeros(2,4,8);
for n=1:8
A(:,:,n)=[ sqrtm(Rxxb(:,:,1,n))  sqrtm(Rxxb(:,:,2,n)) ];
end
[Bu, Gunb, S0, MSWMFunb, B] = mu_bc(Hbc, A, [1 1] , 1);
Bu =   54.1393   62.3515
>> B =  2x8 cell array
    {[3.6736]}    {[7.7329]}    {[7.9256]}    {[6.8322]}    {[5.4843]}    {[6.8322]}    {[7.9256]}    {[7.7329]}
    {[6.5043]}    {[7.1048]}    {[7.9703]}    {[8.5075]}    {[8.6822]}    {[8.5075]}    {[7.9703]}    {[7.1048]}
sum(Bu) =  116.4908   (checks)
GU=zeros(4,4,8);
for n=1:8
GU(:,:,n)=[Gunb{1,n} ; Gunb{2,n}];
end
MSWMFU=zeros(4,1,8);
for n=1:8
MSWMFU(:,:,n)=[MSWMFunb{1,n} ; MSWMFunb{2,n}];
end
```

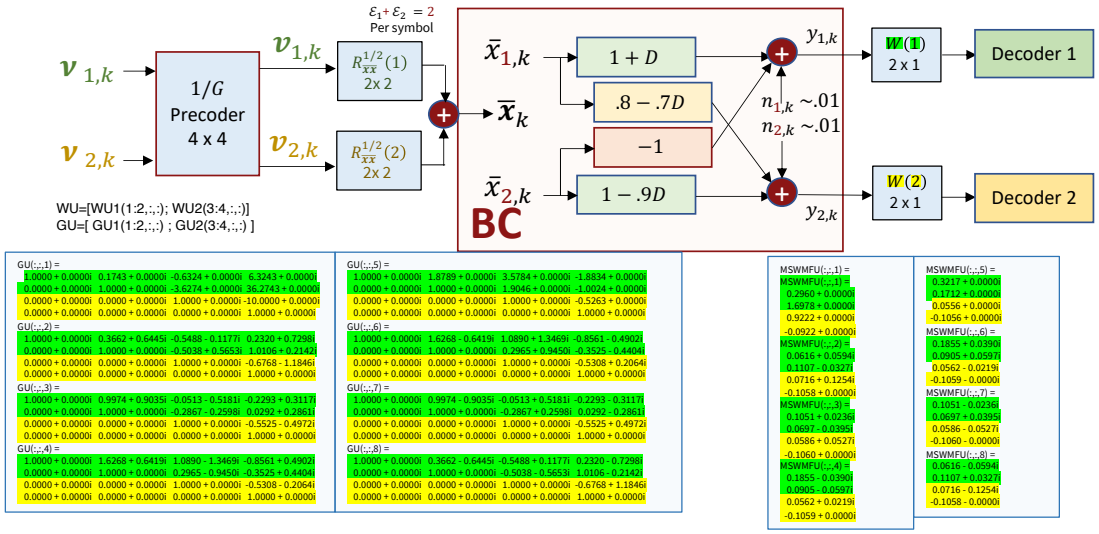


Figure 5.53: 2-user channel as dual BC for Example 5.51.

Figure 5.53 shows the final design:

Realistic design allows for some noise margin on the users and imperfect codes. A reasonable design might allow for a gap of $\Gamma = 1.5$ dB and a margin of $\gamma_m = 1.5$ dB, for a total of 3 dB. This with this complex-baseband channel corresponds to 1 bit/symbol reduction from the bits/symbol of $\tilde{\boldsymbol{b}}' = [b_1 \;\; b_2] = [6 \;\; 7]$ at/near the capacity region boundary to $\tilde{\boldsymbol{b}} = [5 \;\; 6]$. These points are on the slope-equal-one line $b_2 = b_1 + 9$. Figure 5.54 illustrates the operation point within the capacity region. The energies used are those corresponding to the point $\boldsymbol{b}' \in \mathscr{C}(\boldsymbol{b})$.

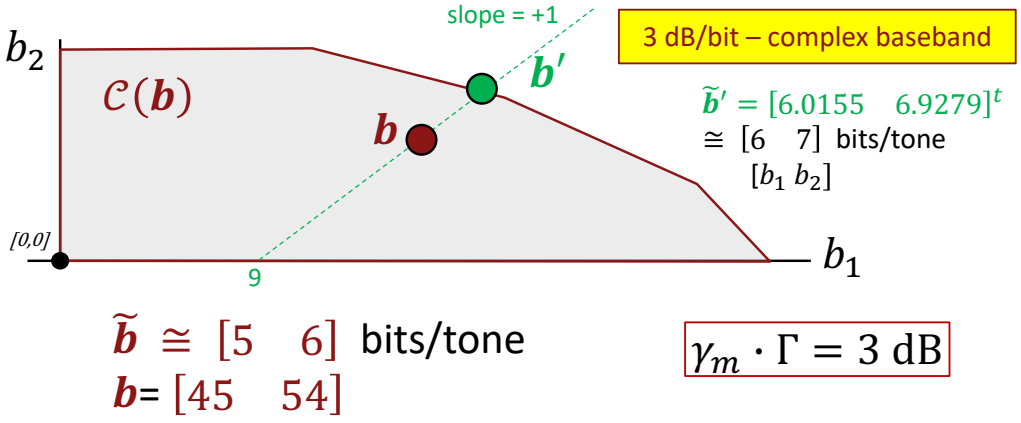


Figure 5.54: Illustration of 1.5 dB margin with 1.5 dB gap BC design 5.51.

Another example that uses space-time only (no tones) appears next. The mu_bc program is not suitable for use in this case, so as n Subsection 2.8.3, a GDFE, with lossless precoder for the vector BC, forms recursively through a series of U GDFE designs, without worst-case noise computation, but using the $R_{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}}(u)$ set that duality (mac2bc.m) creates from a MAC design for the dual (viewed as $\tilde{H}$ and thus

the BC is $\widetilde{H}_{dual}$). However, the multi-step design works for all cases and finds a single precoder through extraction of user u's rows only from the u^{th} GDFE design's unbiased feedback matrix $G_u^{unbiased}$.

EXAMPLE 5.5.5 *[ISI and vector BC via duality]* A single-dimension-output real baseband MAC ($L_y = 1$) has intersymbol interference, as well as the addition of the two users' ISI-distorted outputs before adding AWGN with variance $\frac{\mathcal{N}_0}{2} = .181$. User 1 passes through a $1 + .9 \cdot D$ channel with some input energy on the dual vector MAC channel. User 2 passes through a $1 - D$ channel with some input energy on the dual vector MAC channel, with $\bar{N} = 2$ and $\nu = 1$. The MAC energies correspond (through duality) to the BC users each having unit energies $\mathcal{E}_2 = \mathcal{E}_1 = 3$, so $\mathcal{E}_x^{BC} = 6$. Both dual BC output noises are white with variance $\sigma^2 = 0.181$. With the order reversal, the BC now has user 1 in the preferred or top position with least (no) crosstalk and experiences only ISI, while user 2 now experiences crosstalk from user 1 as well as ISI. In the MAC, user 1 experiences both crosstalk and ISI, while user 2 experiences only ISI.

This example does not use vector DMT, and instead a GDFE in time-domain on each user. This means with $\nu = 1$ that there are 2 channel-output samples for every 3 input samples on each user. The mu_bc program is for vector DMT systems (even with $\bar{N} = 1$) and takes frequency-domain inputs. Thus, this time-domain example uses computeGDFE, which accepts time-domain inputs or one-dimensional/tone inputs with no ISI.

```
>> Hts1=(1/sqrt(.181))*[1 .9 0 ; 0 1 .9] =
    2.3505    2.1155         0
         0    2.3505    2.1155
>> Hts2=(1/sqrt(.181))*[1 -1 0 ; 0 1 -1] =
    2.3505   -2.3505         0
         0    2.3505   -2.3505
```

First, user 1:

```
>> A=[eye(3) eye(3)];
>> [snrGDFEu1, G1U, W1U, S10, MSWMFU1] = computeGDFE(Hts1, A, 2)
snrGDFEu1 =      2.1606
G1U =
    1.0000    0.9000         0    1.0000    0.9000         0
         0    1.0000    0.8006    0.1227    1.0000    0.8006
         0         0    1.0000   -0.5023    0.6591    1.0000
         0         0         0    1.0000    0.4480   -0.4068
         0         0         0         0    1.0000    0.6579
         0         0         0         0         0    1.0000
W1U =
    0.1810         0         0         0         0         0
   -0.1227    0.1610         0         0         0         0
    0.5023   -0.6591    0.9558         0         0         0
   -1.0000   -0.4480    0.4068    1.5842         0         0
   -0.2989   -1.0000   -0.6579   -0.2989    1.7244         0
    0.5262   -0.6376   -1.0000    0.5262   -0.6376    2.6402
S10 =
    6.5249         0         0         0         0         0
         0    7.2107         0         0         0         0
         0         0    2.0463         0         0         0
         0         0         0    1.6312         0         0
         0         0         0         0    1.5799         0
         0         0         0         0         0    1.3788
MSWMFU1 =
    0.4254         0
    0.0522    0.3785
   -0.2137    0.4727
    0.4254   -0.1923
    0.1272    0.3110
   -0.2239    0.4727
>> B1u=0.5*log2(diag(S10)) =
    1.3530
    1.4251
    0.5165
    0.3530
    0.3299
    0.2317
>> sum(B1u(1:3)) =    3.2945
```

The upper 3 GU rows are the feedback/precoder for user 1 on user 1, and user 2 on 1; design ignores bottom 3 rows. This holds for all 6×6 quantities above and for MSWMFU1.

Now for user 2:

```
>> [snrGDFEu2, G2U, W2U, S20, MSWMFU2] = computeGDFE(Hts2, A, 2)
snrGDFEu2 =   2.3736
G2U =
    1.0000   -1.0000         0    1.0000   -1.0000         0
         0    1.0000   -0.8671   -0.1329    1.0000   -0.8671
         0         0    1.0000   -0.4585   -0.5415    1.0000
         0         0         0    1.0000   -0.5415   -0.4585
         0         0         0         0    1.0000   -0.6487
         0         0         0         0         0    1.0000
W2U =
    0.1810         0         0         0         0         0
    0.1329    0.1569         0         0         0         0
    0.4585    0.5415    0.7225         0         0         0
   -1.0000    0.5415    0.4585    1.7225         0         0
    0.3513   -1.0000    0.6487    0.3513    1.7661         0
    0.4784    0.5216   -1.0000    0.4784    0.5216    2.2243
S20 =
    6.5249         0         0         0         0         0
         0    7.3716         0         0         0         0
         0         0    2.3841         0         0         0
         0         0         0    1.5806         0         0
         0         0         0         0    1.5662         0
         0         0         0         0         0    1.4496
MSWMFU2 =
    0.4254         0
   -0.0565    0.3689
   -0.1951   -0.4254
    0.4254    0.1951
   -0.1494    0.2760
   -0.2035   -0.4254
>> B2u=0.5*log2(diag(S20)) =
    1.3530
    1.4410
    0.6267
    0.3302
    0.3236
    0.2678
>> sum(B2u(4:6)) =    0.9217
lower 3 rows are feedback from user 2 into user 2.
ignore top 3 rows
first 3 MSWMF2U rows need not be implemented at receiver 2
```

A final example investigates use of mat2bc for the 64-tone 3-user example, with setting of the $R_{\boldsymbol{xx}}(u)$ at the output of the special square-root matrix A. This process basically absorbs the A_u into the feedback coefficients **between** different users.

EXAMPLE 5.5.6 *[3-user Design revisted]* The 3-user example with a "dual" MAC with design-optimized energy distribution that corresponds to a dual BC that will have the same data rates. This example begins by reforming the 64-tone frequency domain channel transfers for the noise-whitened MAC, which is now the BC's dual. The example also initially reruns minPMAC to find a design for this mac for user rates $[b_3\ b_2\ b_1] = [445\ \ 412\ \ 132$:

```
N=64;
h=cat(3,[1 0 .8 ; 0 1 1],[.9 -.3 0 ; .5 -1 -1],[0 .2 0 ; .4 -.63 0],[0 0 0 ; 0 .648 0])*10;
H64 = fft(h, N, 3);
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H64, [445 412 132]', [1 1 1]',1);
bu_a =  445.0000        412.0000          132.0000
           %  bu_v                          Eun             bun           theta     order   frac      cID
    445.81    411.19       132    {1x3x64 }    {1x3x64 }    {1x3}     {1x3}    0.99      1
    425.68    431.32       132    {1x3x64 }    {1x3x64}    {1x3}      {1x3}     0.01      1
>> info.order{:} =
     3     2     1
     3     1     2
>> info.frac'*info.bu_v =  445.0000  412.0000  131.9999
>> info.frac' =    0.9904    0.0096
>> sum(Eun') =   65.9553   60.2757   40.4453
>> sum(Eun,'all') =  166.6763 < 3 x 64 = 192 (BC will have no individual Es)
```

This design has two vertices shared with the orders and fractions shown. The design first first addresses the vertex with 99% usage:

The next step is form the dual channel in format amenable to the mac2bc program and compute the corresponding dual BC energies and corresponding A_u matrices that specific to each BC user's 3-dimensional input that is summed with the other two users to create the overall $R_{\boldsymbol{xx}}$ (that overall $R_{\boldsymbol{xx}}$ does not explicitly appear in the design). The sqrm matlab command produces warning messages, but still produces valid square-root matrices for the users' discrete modulators.

```
H64mac =reshape(H64,2,1,3,64);
H64bc=zeros(1,2,3,64);
for n=1:64
for i=1:3   H64bc(:,:,i,n) = H64mac(:,:,4-i,n)'; end
end
H64bc1=permute(reshape(H64bc, [2 , 3, 64]), [ 2 1 3]);
Rxxm=zeros(1,1,3,64);
for n=1:64
E=cell2mat(info.Eun);
Rxxm(1,1,:,n)=E(1,:,n);     end
Rxxb=zeros(2,2,3,64);
bbc=zeros(3,64);
for n=1:64   Rxxb(:,:,:,n)=mac2bc(Rxxm(:,:,:,n), H64mac(:,:,:,n)); end
A=zeros(2, 6, 64);
for n=1:64  A(:,:,n)=[ sqrtm(Rxxb(:,:,1,n)) , sqrtm(Rxxb(:,:,2,n)),  sqrtm(Rxxb(:,:,3,n)) ];  end
```

The first vertex' design requires one call of mu_bc.m

```
[Bu, Gunb, S0, MSWMFunb, B] = mu_bc(H64bc1, A, [1 1 1], 1);
Bu =   132.7477  412.8794  445.1264
sum(Bu) = 989.0000
sum(Bu)/67 =   14.7612
Buvertex1=Bu;
```

A subtle point is that the minPMAC energies are not quite perfect match to mu_bc's output rates because of small numerical errors, as the mu_bc essentially is an alternative MMSE calculation. The close match though verifies all the theory behind the design. Because this first vertex occurs 99% of the enlarged symbol/codeword used for sharing, any such small deviation is scaled relative to the 1% point of the second vertex. The corresponding precoder (GU), MSWMFU, and of course the A that was already found constitute the design for vertex one, completed through:

```
GU=zeros(6, 6, 64);
MSWMFU=zeros(6,1,64);
for n=1:64   GU(:,:,n)=[Gunb{1,n} ; Gunb{2,n} ; Gunb{3,n}];
MSWMFU(:,:,n)=[MSWMFunb{1,n}; MSWMFunb{2,n}  ; MSWMFunb{3,n}];  end
GU{:,23}
   1.0000 + 0.0000i   0.0920 - 0.3464i   8.7367 + 1.1630i  10.8139 -15.1709i  -0.6748 - 4.2623i   1.9688 + 0.6639i
   0.0000 + 0.0000i   1.0000 + 0.0000i   3.1234 +24.3936i  48.6591 +18.2925i  11.0106 - 4.8735i  -0.3797 + 5.7850i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   0.9891 - 1.8681i  -1.3553 - 0.3648i   0.4582 - 0.4967i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i  -0.1475 - 0.6474i   0.3091 + 0.0816i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i  -0.2233 + 0.4266i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
A(:,:,23) =
   0.0805 - 0.0000i   0.0074 - 0.0279i   0.2114 + 0.0000i   0.2091 - 0.3950i   0.9368 - 0.0000i  -0.2092 + 0.3996i
   0.0074 + 0.0279i   0.0103 - 0.0000i   0.2091 + 0.3950i   0.9447 + 0.0000i  -0.2092 - 0.3996i   0.2172 - 0.0000i
MSWMFU{:,23} =
   0.8840 - 0.3319i
   1.5285 + 2.1461i
   0.0553 - 0.0808i
   0.0460 + 0.0052i
   0.0535 - 0.0801i
  -0.1988 - 0.0213i
```

Where tone $n = 23$'s precoder, linear transmit filter (square root) and corresponding receiver filters appear.

The second vertex (1%) requires a different **MAC** order [3 1 2] - this corresponds to matlab MAC order [2 1 3], but otherwise follows largely the same steps:

```
H64mac =reshape(H64,2,1,3,64);
H64mac=H64mac(:,:,[2 1 3],:);
```

```
H64bc=zeros(1,2,3,64);
for n=1:64
for i=1:3   H64bc(:,:,i,n) = H64mac(:,:,4-i,n)'; end
end
H64bc1=permute(reshape(H64bc, [2 , 3, 64]), [ 2 1 3]);
Rxxm=zeros(1,1,3,64);
E=cell2mat(info.Eun);
for n=1:64
Rxxm(1,1,:,n)=E(2,[2 1 3],n);     end
Rxxb=zeros(2,2,3,64);
bbc=zeros(3,64);
for n=1:64   Rxxb(:,:,:,n)=mac2bc(Rxxm(:,:,:,n), H64mac(:,:,:,n)); end
A=zeros(2, 6, 64);
for n=1:64  A(:,:,n)=[ sqrtm(Rxxb(:,:,1,n)) , sqrtm(Rxxb(:,:,2,n)),  sqrtm(Rxxb(:,:,3,n)) ];  end
[Bu, Gunb, SO, MSWMFunb, B] = mu_bc(H64bc1, A, [1 1 1], 1);
>> Bu = 131.9999  416.7071  440.2931
>> sum(Bu) =  989.0000
```

This 1% point illustrates the energy finite-precision/numerical deviations' magnification in the two shared users. However, the rate sum remains very accurate. The deviations are evident in the vertices' averaged rate, which is also verfied by direct calculation of the BC users' rates:

```
>> Bu =  131.9999  416.7071  440.2931
>> rate = [Buvertex1 ; Bu]
  131.9999  411.7251  445.2751
  131.9999  416.7071  440.2931
>> info.frac'*rate  =
  131.9999  411.7730  445.2270   versus 412 and 445
  for n=1:64
bbc(1,n)=real(log2(1+H64bc(:,:,1,n)*Rxxb(:,:,1,n)*H64bc(:,:,1,n)'));
bbc(2,n)=real(log2((1+H64bc(:,:,2,n)*(Rxxb(:,:,2,n)+Rxxb(:,:,1,n))*H64bc(:,:,2,n)')/(1+H64bc(:,:,2,n)*Rxxb(:,:,1,n)*H6
bbc(3,n)=real(log2((1+H64bc(:,:,3,n)*(Rxxb(:,:,3,n)+Rxxb(:,:,2,n)+Rxxb(:,:,1,n))*H64bc(:,:,3,n)')/(1+H64bc(:,:,3,n)*(R
end
bvec=sum(bbc') =  131.9999  411.7251  445.2751
bsum=sum(bvec) =  989.0000
```

The vertex share should now be adjusted for the two designs to meet exact the original rates, which is easy by cmputing

```
>> newfrac=inv(rate(1:2,2:3)')*[412 ; 445]  =
    0.9448
    0.0552
```

So the final exact design would better use the two vertices' individual designs in proportion roughly 95% and 5% or basically 19 symbols of 20 for vertex 1 and 1 symbol of 20 for vertex 2.

5.5.5 Generation of the Vector BC Capacity Rate Region

The steps for tracing the vector BC Capacity Region are:

1. Create a dual vector MAC channel (with coefficients $\widetilde{H}$ and noise autocorrelation I).
2. for each $\boldsymbol{b}'$ with $b'_1 = 0, ..., b_{1,max}$, ... $b'_U = 0, ..., b_{U,max}$ with increments selected appropriately and maximums chosen sufficiently large to be outside the rate region (i.e., equal to the single user capacity for all other users zeroed)
   (a) Find the energy vector $\boldsymbol{\mathcal{E}}$ for a given $\boldsymbol{b}$ on the dual vector MAC using the minPMAC program of Section 5.4.
   (b) if $\sum_u \mathcal{E}_u \leq \mathcal{E}$, then the point is in the region, so $c_{new}(\boldsymbol{b}) = \{\boldsymbol{b}' \bigcup c_{old}(\boldsymbol{b})\}$.
3. Trace the boundary for of $\boldsymbol{b}$ in Step 2 for which $\sum_u \mathcal{E}_u = \mathcal{E}$.

5.6 Gaussian IC with GDFE

As in Section 2.6.7 and now focusing on the Matrix-AWGN IC, allocation of energy and information to IC users divides into situations of Section 5.6.1's central and Section 5.6.2's distributed management . Section 2.6.7's consensus management will be viewed here as intermediate and allowing some information passing between users in overhead messages and effectively part of an enhanced distributed management in Section 5.6.3. IC random-access management largely reverts to Section 2.6.7.4's collision avoidance (already studied and not repeated here) that increasingly yields to Section 5.6.2's consensus management alternatives. This section opens with central management description versus distributed management before proceeding to optimization within each situation.

The Centrally Controlled Gaussian IC (CIC) Figure 5.55 describes the CIC. The CIC situation includes the **Distributed Antenna System (DAS)** concept[86]. This concept fundamentally is the same as **cell-free wireless**. The CIC permits central assignment of user data rates, codes, bit distributions and corresponding user input autocorrelation matrices $R_{\boldsymbol{xx}}(u)$ through the central controller that knows all noise autocorrelation matrices $R_{\boldsymbol{nn}}(u)$ and channel H_{ij} entries. This CIC is Chapter 2's IC. For instance, Vector DMT and multi-user GDFE's are possible at each user's CIC receiver. All CIC users synchronize to a common symbol rate, $1/T$.

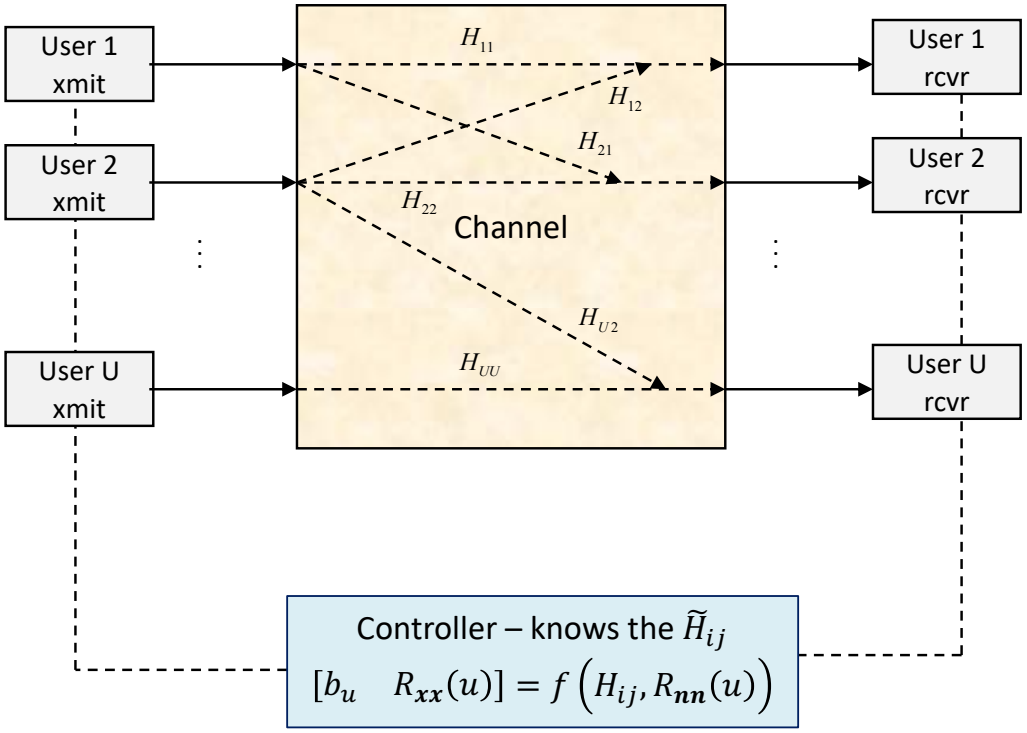


Figure 5.55: **CIC or "DAS" Architecture.**

CIC DAS versions may have $L_{x,u} > 1$ and/or $L_{y,u} > 1$. DAS uses space-division multiplexing of different users' transmitted energy to improve spectrum efficiency. Similarly, users may also use frequency- or time-division multiplexing. IC receivers can employ GDFE receivers for the known codes

[86]The word distributed in DAS refers to the antennas' locations and not to the control; indeed most DAS systems presume and exploit central coordination of their various transmissions, particularly in space.

of all users; these receivers may decode some or all other users first wherever possible according to each receiver's order $\boldsymbol{\pi}_u$. Users' transmitters may employ individual matrix filters that a central controller designs. The controller may separate dimensional energization per user or may allow users to share dimensions.

The Locally Controlled Gaussian IC (LIC) Theoretically, local control is not truly an extra IC constraint because a designer could presumably guess the correct codes (and receivers if given sufficiently long time could presumably ascertain the constellations and codes used on all other users if those codes were not quite, but almost, Gaussian). However, the LIC refines the constraints where the central design of spectra and bit distributions may not be feasible. This may appear then "random" to all other users, possibly stationary or possibly non-stationary.

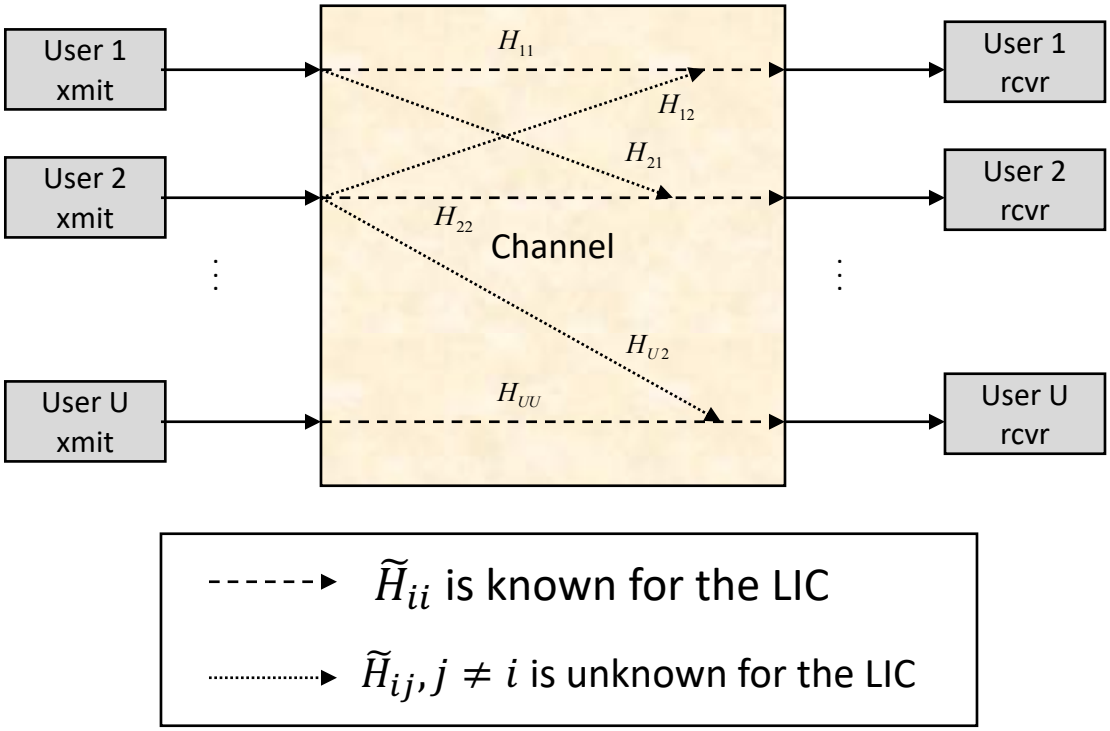


Figure 5.56: **Locally Controlled IC (LIC)**

The LIC may presume only that the transmitter and receiver u know only the following 4 items (see Figure 5.56):

1. the received noise autocorrelation $R_{noise}(u)$ (presumed Gaussian including contributions in aggregate from all other users and the Gaussian noise)
2. its own channel H_{uu} (but not any H_{iu} nor $H_{ui}$where $i \neq u$)
3. The transmit autocorrelation $R_{\boldsymbol{xx}}(u)$ of its own channel
4. the bit distribution and thus total rate of its own channel $b_{u,n}$ and b_u respectively.

Furthermore, multiuser GDFE/successive-decoding will not occur in the LIC, except possibly in single-user form for any particular user, u', with any decisions only within the dimensions that user energizes. Specifically, the LIC strategy does not use receiver crosstalk cancellation nor transmitter precoder-based crosstalk cancellation, but can address its user's intersymbol interference.

5.6.1 CIC Optimization

Section 2.9.1's MAC-set approach views the IC as U parallel MACs. Subsection 5.6.1.1 describes Optimum Spectrum Balancing (OSB), a complex method that applies to the non-convex situation where no user's crosstalk is cancelled and instead all other users' crosstalk is additional noise. This method provides some insight and leaves a computationally complex foundation for understanding Subsection 5.6.1.2's IC approaches that use the higher-performance multiuser GDFE in all receivers. Subsection 5.6.1.2 builds upon the OSB foundation, but then suggests use of Section 5.4.3's MAC optimization methods appropriately modified for Section 2.9.1's MAC-set approach. With the MAC-set approach, the $\boldsymbol{\delta}$ expands to (up to) $(U')^2$ entries $\boldsymbol{\Delta}$, whose positive semidefinite need implies an order.

5.6.1.1 Optimum Spectrum Balancing (OSB)

OSB presumes synchronized DMT modulators with common symbol boundaries.[87] OSB maximizes a weighted energy sum with $\boldsymbol{w} \succcurlyeq \boldsymbol{0}$. The enumeration of possible $\boldsymbol{w}$ traces an OSB achievable region $\mathcal{A}_{OSB}(\boldsymbol{b})$, when each IC receiver has access only to its own signal; it does not attempt to decode any other user. A controller calculates all the users' spectra centrally. OSB's name "optimum" is a misnomer in that OSB is not an overall IC optimum, but is optimum under certain restrictions. OSB otherwise is a highly computationally complex method, but guides development of more practical approaches that later appear in Subsection 5.6.2.

OSB minmizes[88]

$$\min_{\{R_{\boldsymbol{xx}}(u,n)\}} \quad \sum_{u=1}^{U} w_u \cdot \mathcal{E}_u \tag{5.406}$$

$$ST: \quad \text{for} \;\; u = 1, ..., U \tag{5.407}$$

$$0 \leq \sum_{n} \text{trace}\,\{R_{\boldsymbol{xx}}(u,n)\} \leq \mathcal{E}_{u,max} \;\; \text{AND} \tag{5.408}$$

$$b_u = \sum_{n} \log_2 \frac{\mid H_{uu,n} \cdot R_{\boldsymbol{xx}}(u,n) \cdot H^*_{uu,n} + \mathcal{R}_{noise}(u,n) \mid}{\mid \mathcal{R}_{noise}(u,n) \mid} \quad . \tag{5.409}$$

(5.409)'s presumed relationship between user bit distribution and autocorrelation matrices specifically includes all crosstalk as distortion; again OSB uses no successive decoding nor GDFE. This OSB derivation uses $L_x = L_y = 1$. Later, simplified OSB-aproximations generalize more readily to larger L_x and L_y.. Thus, $R_{\boldsymbol{xx}}(u,n) \rightarrow \mathcal{E}_{u,n}$. The interested reader can readily extend to MIMO ($L_{x,u} > 1$ and/or $L_{y,u}$), with attendant computationaly increase to exhorbitently impractical levels. The corresponding scalar tonal Lagrangian forms as the sum:

$$L_n \triangleq L_n(R_{\boldsymbol{xx}}(u,n), \boldsymbol{b}_n, \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{u=1}^{U} [w_u \cdot \mathcal{E}_{u,n} - \theta_u \cdot b_{u,n}] \quad . \tag{5.410}$$

(5.406)'s "minE-OSB" Lagrangian is, with user energy constraints being the diagonal elements of $\boldsymbol{\mathcal{E}}$,

$$L = \sum_{u=1}^{U} \left[\sum_{n} L_n \right] - \theta_u \cdot b_u \quad . \tag{5.411}$$

(5.411)'s Lagrangian is not convex because each user depends on all other users' spectra. However, it does have a solution. The achievable rate region, with no successive decoding, is the convex hull over all $\sum_{u=1}^{U} \theta_u = 1$ with $\boldsymbol{\theta} \succeq \boldsymbol{0}$ and over each's implied constraint $\boldsymbol{w} \succeq \boldsymbol{0}$'s variation. Since there is no

[87] Such synchronization may use GPS timing references outdoors or some other common or learned network clock source generally.

[88] This problem is the same as maximizing one users' rate while all others are each lower-bounded at some desired rate. OSB and ISB were introduced respectively by Dr. Raphael Cendrillon and Prof Wei Yu when the former was a visiting student at Stanford in 2003 and the latter was a PhD student, [3], [2].

crosstalk cancellation, there is no decoder order $\boldsymbol{\pi}_u$. Each user's bits/symbol can include a gap (with $\text{SNR}_{u,n} \triangleq \frac{|H_{uu,n}|^2 \cdot \mathcal{E}_{u,n}}{R_{noise}(u,n)}$) and provides the constraint relating $\mathcal{E}_u$ and b_u.

$$b_u = \sum_n \log_2 \left(1 + \frac{\text{SNR}(u,n)}{\Gamma} \right) \quad . \tag{5.412}$$

Since each user employs its own separate code that treats all others as noise, then the single-user gap approximation applies directly.

The same Lagrangian also applies with fixed user energies to the rate-sum maximization where the trailing term in (5.411) becomes $-w_u \cdot \mathcal{E}(u)$ instead of $-\theta_u \cdot b_u$. Equivalently, the doubly constrained problem can be checked for energy-constraint satisfaction at any given $\boldsymbol{b}$ (the admission problem[89]). This check can be used to generate a new $\boldsymbol{\theta}$, which then can be subsequently used again in the original problem. Thus OSB algorithm has two steps:

energy step minimize L_n for fixed $\boldsymbol{w}$ and $\boldsymbol{\theta}$ by using the known capacity relation in (5.412).

constraint step Optimize using sub-gradient descent (via the ellipsoid, cutting-plane, or other methods) for the value of $\boldsymbol{\theta}$ (or both $\boldsymbol{\theta}$ and $\boldsymbol{w}$ in the admission-problem context).

The second step mirrors the minPMAC's order step. However, the $\boldsymbol{\theta}$ element relation no longer directly reflects a decoding oder. The second step may use the same (elliptical or sub-gradient) algorithms as mimPMAC. The first step differs (from minPMAC) because of the $\mathcal{E}_{u,n}$ interdependencies without successive decoding, a nonconvex functionality. This step instead exhaustively searches all users' possible energy settings.

Consequently, the energy step evaluates all energy increments up to a maximum of

$$M = \frac{\max_u \boldsymbol{\mathcal{E}}(u)}{\Delta\mathcal{E}} \tag{5.413}$$

for some energy search increment $\Delta\mathcal{E}$; then this step evaluates M^U values of L_n for each tone. This energy step might be equivalent to one more bit in the constellation or per dimension. This is a high complexity for more than 2 or 3 users. The actual complexity has order $O(N \cdot U \cdot M^U)$ because each $R_{noise}(u,n)$ calculation itself requires U operations.

The constraint step can use sub-gradients (let $\boldsymbol{\mathcal{E}}_{max}$ be a vector of the diagonal elements of $\boldsymbol{\mathcal{E}}$, and consequently sum of $\boldsymbol{\mathcal{E}}_n$, the vector of energies for the users on any tone n)

$$\Delta\boldsymbol{b} = \boldsymbol{b}_{min} - \sum_n \boldsymbol{b}_n \tag{5.414}$$

$$\Delta\boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{E}}_{max} - \sum_n \boldsymbol{\mathcal{E}}_n \quad . \tag{5.415}$$

These sub-gradients permit a direct update of the Lagrange multipliers according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \cdot \Delta\boldsymbol{b} \tag{5.416}$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \epsilon' \cdot \Delta\boldsymbol{\mathcal{E}} \quad . \tag{5.417}$$

Typically, a more sophisticated elliptical procedure is necessary for reasonable convergence in practice, where the initial condition for $\boldsymbol{\theta}$ (and $\boldsymbol{w}$ in the admission problem) is in the first orthant. The algorithms increment successively and iteratively the energy step and the constraint step until the Lagrangian is sufficiently close to zero (and thus slightly positive).

OSB really is a suboptimum central management system, but focuses only on spectrum setting (and not maximums with best decoders and signal processing/modulation). In so narrowing focus, however, OSB paves a path to simpler methods that ultimately can be largely distributed in implementation and will be effectively optimum in Section 5.6.2's LIC methods.

[89] which results in negative Lagrangian if not feasible

An approach to the energy step: The 2×2 OSB energy step essentially is a matrix inverse for each possible energy increment or equivalently energy increment relative to a single-bit increase. For the 2×2 IC, the SNR's are:

$$SNR_1 = 2^{2\bar{b}_1} - 1 = \frac{\bar{\mathcal{E}}_1 |H_{11}|^2}{\bar{\mathcal{E}}_2 |H_{12}|^2 + \sigma_1^2} \tag{5.418}$$

$$SNR_2 = 2^{2\bar{b}_2} - 1 = \frac{\bar{\mathcal{E}}_2 |H_{22}|^2}{\bar{\mathcal{E}}_1 |H_{21}|^2 + \sigma_2^2} \quad . \tag{5.419}$$

Solving for $\bar{\mathcal{E}}_1$ and $\bar{\mathcal{E}}_2$ produces the linear equation:

$$\begin{bmatrix} |H_{11}|^2 & -SNR_1 \cdot |H_{12}|^2 \\ -SNR_2 \cdot |H_{21}|^2 & |H_{22}|^2 \end{bmatrix} \cdot \begin{bmatrix} \bar{\mathcal{E}}_1 \\ \bar{\mathcal{E}}_2 \end{bmatrix} = \begin{bmatrix} SNR_1 \cdot \sigma_1^2 \\ SNR_2 \cdot \sigma_2^2 \end{bmatrix} \quad , \tag{5.420}$$

which solves by basic matrix inversion for each of the $\bar{b}_{max}^U$ possible solutions.

Generalizing the OSB search: The general set of linear equations is:

$$\begin{bmatrix} |H_{11}|^2 & -SNR_1 \cdot |H_{12}|^2 & \dots & -SNR_1 \cdot |H_{1U}|^2 \\ -SNR_2 \cdot |H_{21}|^2 & |H_{22}|^2 & \dots & -SNR_2 \cdot |H_{2U}|^2 \\ \vdots & \vdots & \ddots & \vdots \\ -SNR_U \cdot |H_{U1}|^2 & -SNR_U \cdot |H_{U(U-1)}|^2 & \dots & |H_{UU}|^2 \end{bmatrix} \cdot \begin{bmatrix} \bar{\mathcal{E}}_1 \\ \bar{\mathcal{E}}_2 \\ \vdots \\ \bar{\mathcal{E}}_U \end{bmatrix} = \begin{bmatrix} SNR_1 \cdot \sigma_1^2 \\ SNR_2 \cdot \sigma_2^2 \\ \vdots \\ SNR_U \cdot \sigma_U^2 \end{bmatrix} \quad , \tag{5.421}$$

The solutions are evaluated as $SNR_i = 2^{2 \cdot [0:b_{max}]} - 1$, so $(b_{max}+1)^U$ possible values' solutions are found and evaluated (for each tone) to determine a best L_n value for the energy step.

The osb.m program Former student Dr. Aakanksha Chowdhery initially provided this program, which has significant edits by the author here.

```
function [S1, S2, b1, b2] = osb(Hmag_sq, No, E, theta, mask, ...
  gap, bitcap, cb)

  This is the main program call to osb.  It sets up calculation of w1
  and calls a subroutine that in turn sets up w2, which in turn calls
  another routine that does the main crosstalk calculations.

  Inputs

  Hmag_sq is a N x 2 x 2 where N is FFT size. N inferred from this.
  No      is a 1 x U white-noise power spectra density matrix.
     If Hmag_sq is complex BB, then No should be the one-sided PSD.
  E       is a 1 x U energy vector.
  theta   is a 1 x U user-rate weighting vector.
  mask    is an N x U PSD maximum allowed.
  gap     is the (non-dB) linear gap (so 1 if 0 dB gap).
  bitcap  is a 1 x U maximum number of bits allowed per tone.
  cb      is 2 for real baseband and 1 for cplex bband

  Outputs

  S1      is user 1's Nx1 PSD
  S2      is user 2's Nx1 PSD
  b1      is user 1's Nx1 bit distribution
  b2      is user 2's Nx1 bit distribution
```

```
calls optimize_l2.m, which calls optimize_s.m
-------------------------------------------------------------
```

This program calls two more programs that Appendix G lists. The optimize_s.m program contains the matrix inversions.

EXAMPLE 5.6.1 *[Simple 2 × 2 IC with osb.m] secondary users components that can be viewed as 1st and 2nd users.*

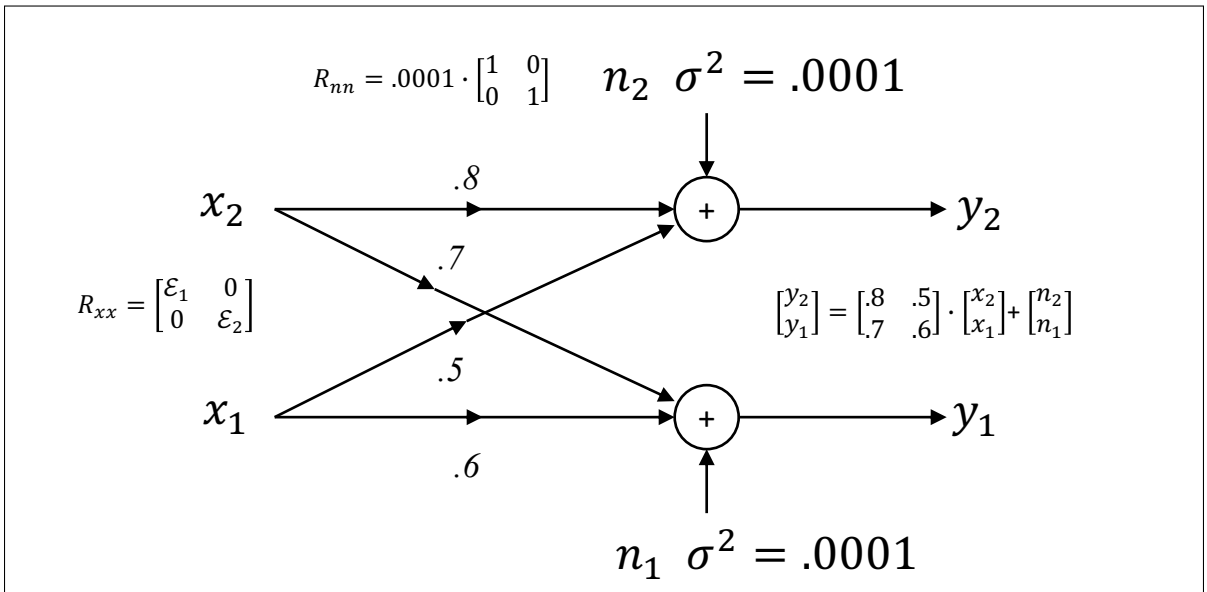


Figure 5.57: Interference-channel for Example 5.57.

This example returns to Chapter 2, Section 9's earlier IC example as in Figure 5.57

$$H = \begin{bmatrix} .8 & .5 \\ .7 & .6 \end{bmatrix} \tag{5.422}$$

with white noise $\sigma^2 = .0001$ The data rates are a function of the weights $\boldsymbol{w}$ chosen as per:

```
>> H2
H2(:,:,1) =    0.6400    0.2500  % note this is squared mag each term
H2(:,:,2) =    0.4900    0.3600
>> Noise  =   1.0e-04 * [    1.0000    1.0000];
>> Ex =[     1     1];
>> mask =[     1     1];
>> gap =     1;
>> bitcap =[    15    15];
>> [S1, S2, b1, b2] = osb(H2, Noise, Ex, [0.5 .5], mask, gap, bitcap,2)

S1 =    0.6398
S2 =     0
b1 =     6    < 6.3 fwith successive decoding
b2 =     0
>> [S1, S2, b1, b2] = osb(H2, Noise, Ex, [0.01 .99], mask, gap, bitcap,2)
```

```
S1 =     0
S2 =    0.1419
b1 =     0
b2 =    4.5000 <5.9 with successive decoding
```

Note that OSB here produces a solution that is typically only one user at a time or per use. The weighting determines which user. This is typical of OSB.

EXAMPLE 5.6.2 *[Simple 2×2 IC with ISI and with osb.m]* This example returns to the early MAC/BC channel of Figure 5.51 with 8 tones and white noise $\sigma^2 = .01$. The data rates are a function of the weights $\boldsymbol{w}$ chosen as per:

```
h = cat(3, [1   .8 ;  -1    1], [-.9   -.7 ;  0   1]  )*10;
He = fft(h, 8, 3);
>> H3=zeros(8,2,2);
>> H3(:,1,1)=He(1,1,:);
>> H3(:,2,1)=He(2,1,:);
>> H3(:,2,2)=He(2,2,:);
>> H3(:,1,2)=He(1,2,:);
>> Noise=ones(8,2);
>> mask=ones(8,2);
>> Ex=8*Ex;
>> [S1, S2, b1, b2] = osb(H3, Noise, Ex, [0.5 .5], mask, gap, bitcap,1);
>> S1' =      0             0           0.7017    0.8272     0    0.8272    0.7017          0
>> S2' =    0.6375    0.7469         0         0             0         0             0

>> b1' =     0     0     7     8     0     8     7     0
>> b2' =     8     8     0     0     0     0     0     8
>> sum(b1) =    30
>> sum(b2) =    24
sum(b1+b2) =    54  < ~116 that MAC, BC, single had for this channel
```

Note that OSB here again produces an FDM solution that is typically only one user at a time or per dimension. The weighting determines which user. This is typical of OSB.

5.6.1.2 GDFE-based CIC optimization

The minimum IC energy sum[90] again corresponds to the set of U input energies that minimize a weighted sum for a MAC-set of canonical (GDFE) designs to that target a common fixed rate vector $\boldsymbol{b}_{min}$. A receiver index will be $u \in \boldsymbol{U}$ while the user index at any receiver will be $i \in \boldsymbol{U}$. Thus $b_{i,u}$ refers to user i's achieved data rate at receiver u. Ultimately, $b_{u,u}$ are the user target data rates.

$$\min_{\{R_{\boldsymbol{xx}}(u)\}} \quad \sum_{u=1}^{U} w_u \cdot \text{trace} \underbrace{\{R_{\boldsymbol{xx}}(u)\}}_{\mathcal{E}_u} \tag{5.423}$$

$$ST: \quad b_{i,u} \geq \left\{ \begin{array}{ll} b_{min,i} & \pi_u(i) \leq \pi_u(u) \\ 0 & \pi_u(i) > \pi_u(u) \end{array} \right\} \triangleq b_{min(\pi_u),u,i}$$

$$R_{\boldsymbol{xx}}(u) \succeq \boldsymbol{0} \qquad .$$

The individual IC receiver i data rate for any user u is equal to $b_{min,u}$ if the user must be decoded (which corresponds to a decoding order $\boldsymbol{\Pi}$ where user u must be reliably decoded at receiver i). If the user need not be decoded, then the data rate is simply nonzero (which corresponds to a decoding order where u's crosstalk is added noise and user u need not be reliably decoded at receiver i). These are U^2 bit-rate

[90] Assumes that all users employ capacity-achieving codes individually with $\Gamma = 0$ dB. With the successive-decoding GDFE now in use, multiuser margin returns, while it was unnecessary in OSB.

constraints are the capacity region's boundary, which is convex. This is thus a convex constraint. When more than one Lagrange multiplier is zero at receiver i, the corresponding users may have any order subsequent to the decoding of user i at receiver i. The order is given during any optimization step, but it can be changed after each such step. Indeed a separate $\boldsymbol{\theta}$ update may lead to a consequential order update before $\{R_{\boldsymbol{xx}}(u)\}$'s convex optimization.

The vector $\boldsymbol{w} = [11...1]^*$ corresponds to the energy sum; more generally $\boldsymbol{w} \succeq \boldsymbol{0}$. Theoretically, (5.423) always has a solution for a given $\boldsymbol{b}$. A difference with the MAC is that there are actually U^2 IC constraints on the U receiver rate vectors $\boldsymbol{b}_u$. Similar to the MAC, (5.423) may not produce a $\boldsymbol{b} \in \mathcal{C}_{IC}(\boldsymbol{b})$ because the designer may not know à priori the capacity region in selecting $\boldsymbol{b}$. When this happens, (5.423)'s minimized energy vector solution $\boldsymbol{\mathcal{E}}$ may exceed the energy limit $\boldsymbol{\mathcal{E}} \not\preceq \boldsymbol{\mathcal{E}}_{\boldsymbol{x}}$. (5.423) describes the capacity region point with lowest $\boldsymbol{w}^*\boldsymbol{\mathcal{E}}$. When $\boldsymbol{w} = [1\ 1\ ...\ 1]^*$, this is the minimum-sum-energy point on the capacity-energy-region boundary $\mathscr{C}_{\boldsymbol{b},IC}(\boldsymbol{\mathcal{E}})$. As in Subsection 5.4.5, an extension determines if $\boldsymbol{b}$ is **admissible**, that is $\boldsymbol{b} \in \mathcal{C}_{IC}(\boldsymbol{b})$ for the given $\boldsymbol{\mathcal{E}}$. Individual IC users' energies satisfy $\sum_n \text{trace}\{R_{\boldsymbol{XX}}(u,n)\} \leq \mathcal{E}_u,\ \forall\ u \in \boldsymbol{U}$ when for at least one user $L_x > 1$. When $L_{x,u} = 1\ \forall\ u \in \boldsymbol{U}$, these terms reduce to the individual-user energies $\mathcal{E}_u$ that arise from summing energy over user u's tones.

Equation (5.423)'s Lagrangian is

$$\begin{aligned} L(R_{\boldsymbol{XX}}(u), \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\Theta}) \quad = \quad & \sum_{u=1}^{U} \left(w_u \cdot \left[\sum_{n=0}^{\overline{N}} \text{trace}\{R_{\boldsymbol{XX}}(u,n)\} \right] \right. \\ & \left. - \sum_{i=1}^{U} \left[\theta_{u,i} \cdot \left\{ \left[\sum_{n=0}^{\overline{N}-1} b_{u,i,n} \right] - b_{min(\pi_u),u,i} \right\} \right] \right) \ \forall\ u \in \boldsymbol{U} \end{aligned} \tag{5.424}$$

(presuming no single-side-band uses of tone $n = \overline{N}$ tone if baseband real). Each receiver's tonal Lagrangian is

$$L_n(R_{\boldsymbol{XX}}(u,n), b_{u,i,n}, \boldsymbol{w}, \boldsymbol{\Theta}) = \sum_{u=1}^{U} \left(w_u \cdot [\text{trace}\{R_{\boldsymbol{XX}}(u,n)\}] - \sum_{i=1}^{U} [\theta_{u,i} \cdot b_{u,i,n}] \right) \ \forall\ u \in \boldsymbol{U} \tag{5.425}$$

The sum over i is the essential difference between the IC and the MAC. Correspondingly, any gradient descent algorithm's Hessian and gradient matrices will have corresponding sums of terms like those in the MAC minPMAC-like optimization. An outer $\boldsymbol{\theta}$ optimization also proceeds similarly, but on U^2 possible $\boldsymbol{\Theta}$ values, some of which (for any current order) may target $b_{u,i} = 0$.

There are two optimum CIC approaches:

minimum weighted energy sum: This is the sum with specified energy weights $\boldsymbol{w} \succeq 0$, where $\boldsymbol{\Theta} \succeq 0$ are the side-constraint multipliers to minimize (maximize with negative sign) rate-constraint impact and determine all receivers corresponding orders, or

maximum weighted rate sum for the order implied by the specified $\boldsymbol{\Theta} \succeq 0$, where $\boldsymbol{w} \succeq 0$ as the side-constraint multipliers to minimize (maximize with negative sign) energy-constraint impact. Essentially this rate is U times the rate sum, because all IC users target the same rate sum $b = \sum_u b_u$, but that constraint is not evident in the dual solution.

Initialize Ellipsoid: To determine the initial ellipsoid that contains $\boldsymbol{\Theta}^o$ on any order step, this step's initialization selects a random order and a user-bit/symbol distribution b_i^o such that

$$b_{j,i\neq u} = b_u^o\ ,\forall\ j \in \boldsymbol{U} \tag{5.426}$$

$$b_{j,u} = b_u^o + 1\ ,\forall\ j \in \boldsymbol{U}\quad . \tag{5.427}$$

Initialization makes a single-pass[91] through Chapter 4's fixed-margin iterative water-filling, for the selected order and bit distribution in (5.335) and (5.336), generates the bit distribution $\boldsymbol{b}^o$, thus generating

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

a set of $\{R_{\boldsymbol{XX}}(u,n)\}$. This set of autocorrelation functions substitutes into the Lagrangian equation, which must always be non-negative for $\boldsymbol{\Theta}^o \succeq 0$, so

$$0 \leq L_{min}(\boldsymbol{\Theta}^o) \leq \left[\sum_{u=1}^{U}\sum_{n} w_u \cdot \text{trace}\left(R_{\boldsymbol{XX}}(u,n)\right)\right] + \left[\sum_{j=1}^{U}\sum_{u=1}^{U}\left(b_{j,u} - \sum_{n} b_{j,u,n}\right)\right] \cdot \theta_u^o \quad . \tag{5.428}$$

Rearranging this equation using (5.427) leads to

$$0 \leq \theta_u^o \leq \sum_{u=1}^{U}\sum_{n} w_u \cdot \text{trace}\left(R_{\boldsymbol{XX}}(u,n)\right) = \sum_{j=1}^{U} \theta_{j,u,max} \quad . \tag{5.429}$$

The step in (5.427) repeats for each user (that is incrementing each user successively by one bit as in (5.426) and (5.427)) to generate a $\theta_{j,u,max}$ for each, as in (5.429). 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) and each of the U possible IC receivers, a box containing $\boldsymbol{\Theta}^o$ is obtained. The eigenvector-axised ellipsoid that covers this box has diagonal Hessian with squared lengths at least $\theta_{j,u,max}^2$ on each of these axes.

$$A_0^{-1} = \begin{bmatrix} \left(\frac{1}{\theta_{1,1,max}}\right)^2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \left(\frac{1}{\theta_{U,U,max}}\right)^2 \end{bmatrix} \quad . \tag{5.430}$$

This step's algorithm runs until $\boldsymbol{\theta}_{j,k+1} - \boldsymbol{\theta}_{j,k} < \epsilon$.

The minPIC.m program awaits a motivated student's project or large extra credit.

5.6.2 LIC and Distributed Management Strategies

Distributed management essentially means that each IC receiver u and transmitter u can measure the channel as if all other users were noise. This reduces performance with respect to the CIC optima in Section 5.6.1.2, and it is common with Subsection 5.6.1.1's OSB. However energy/information is local, so an LIC. Such performance-reducing constraint may be a practical necessity. This section then describes some methods, including distributed-management approximations to the optimum spectrum balancing (OSB) method, with goal to approach best LIC performance.

5.6.2.1 Iterative Water-filling

Chapter 2's Iterative water-filling (IW) addresses the vector MAC's rate sum. However, IW can also apply well to the LIC, where IW improves overall LIC performance in achieving what is sometimes called a "**Nash equilibrium,**" a point where all users inputs are such that none can change individually without degradation. The LIC's use of IW has all receivers treat all other users' signals as noise, Luo and Pang[6]. IW convergence for the IC situations largely occurs. Of interest are diagonally dominant channels. Such channels imply that $|H_{uu}(f)| >> |H_{ui}(f)|$ when $i \neq u$. In effect the transfer function of the user's channel is significantly larger than the transfer function from any other user into this same user. Such may occur in DAS systems that essentially use fixed (or slowly adapting) "beamforming" to impose diagonal dominance. There may be many points to which IW can converge on the LIC (but any may be acceptable improvement over some less adaptive design approaches), unlike IW's unique vector-MAC SWF convergence point.

Essentially, each user being as polite as possible is about the only acceptable LIC-compatible solution. LIC constraints will relax slightly to the improvements in Subsections 5.6.3 and 5.6.3.2.

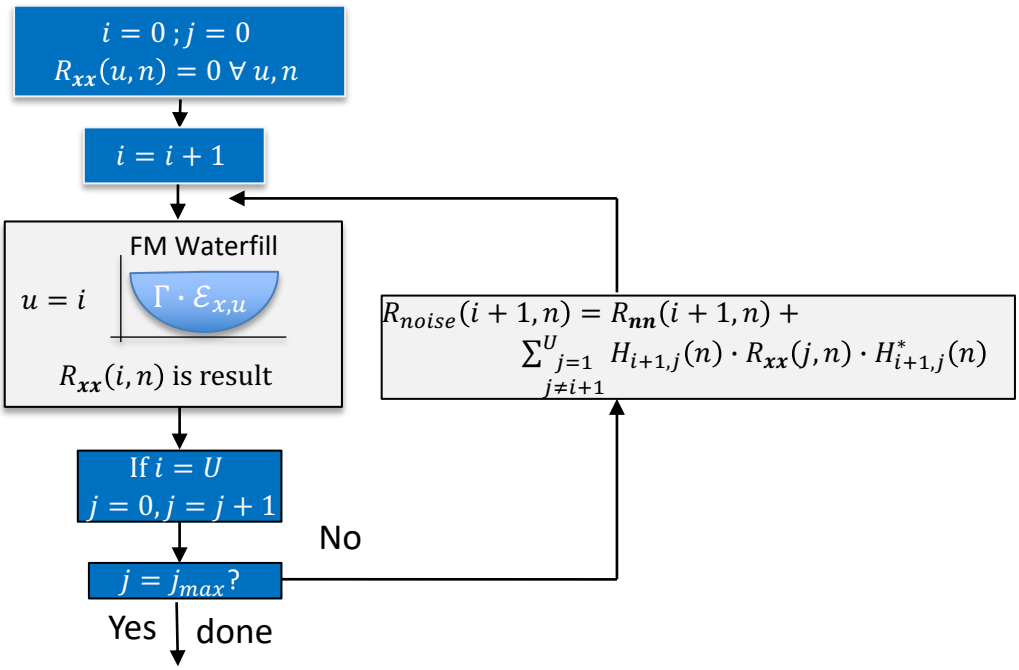


Figure 5.58: **Iterative Water-filling (fixed margin).**

5.6.2.2 The IW Algorithm for the LIC

In Figure 5.58's IW for the IC, each user water-fills energy by treating **all** other crosstalking signals as Gaussian noise. User u's spectrum water-fills using the noise-referred to channel input spectrum curve $\left[\sum_{i \neq u} H_{ui}^2 \cdot S_i(f) + S_n(f)\right] / |H_{uu}|^2$ as (more generally with coding gap Γ and presuming energy-per-dimensional quantities)

$$\lambda_u = \mathcal{E}_{u,n} + \frac{\Gamma \cdot \left[\sigma_n^2 + \sum_{i \neq u} |H_{ui}|^2 \cdot \mathcal{E}_{i,n}\right]}{|H_{uu,n}|^2} \quad . \tag{5.431}$$

In actual use, each user presumably locally implement a water-fill-based loading that treats all other users as noise. All users may adjust simultaneously. Simulations instead usually hold all other users constant and implement a water-fill loading algorithm for the present iteration's user. This simulation then iterates through all users several times until the spectra of all users converge. Field use simply converges if the $\boldsymbol{b}$ is within the FM IW achievable region.

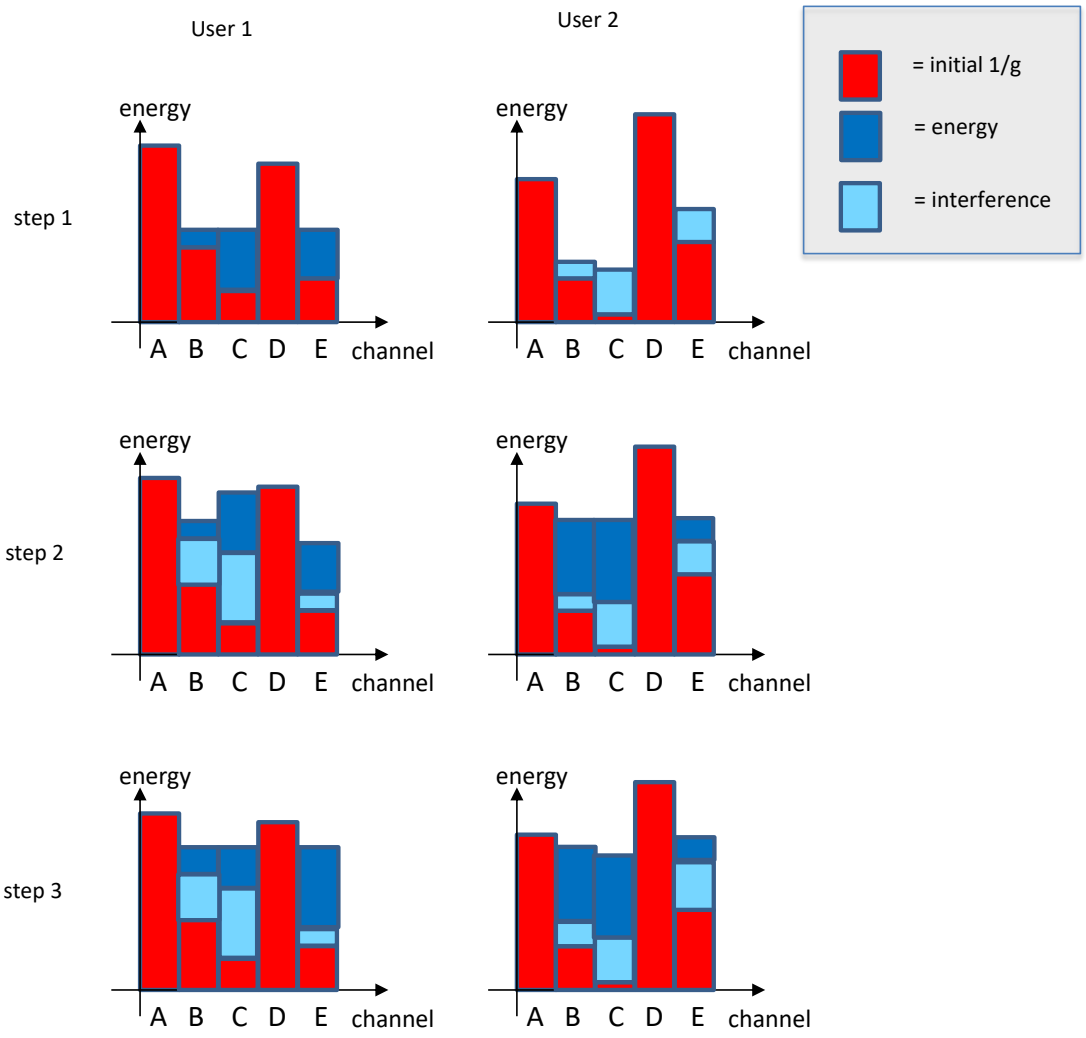


Figure 5.59: **Simple IW Example.**

FM water-filling's use is important because it corresponds to polite energy use where no user has excessive margin. In particular, the non-unique convergence point to which IW converges is usually then a good one. It is not necessarily optimum it all cases, but it is usually close and almost always by definition improves over a static-spectrum choice. Figure 5.59 illustrates a hypothetical 5-dimensional channel (Dimensions have indices A, B, C, D, and E to emphasize that dimensions may be space, time, and/or frequency). Each of two users can use any of the 5 bands. In step one, user 1 water-fills (dark blue), which creates a crosstalk/interference to user 2 on Figure 5.59 step 1's right spectrum (light blue), changing the channel-input referenced spectrum ($1/g$, red color) for user 2. User 2 then waterfills in step 2 (right side) with the dark-blue user 2 energy. However, this then causes user 1 (light blue) to have interference/crosstalk that renders its dark blue energy sub-optimum. User 1 then waterfills again against the new combined noise, but causes user 2 previous water-fill to be slightly suboptimum. At each step, especially with FM IW, both users reduce deviation from optimum. IW eventually leads to a simultaneous waterfill spectra for both users with the other as noise.

5.6.2.3 Near/Far ICs

Figure 5.60 illustrates the "**near/far**" problem where signals from a near transmitter overwhelm signals from a far transmitter at a receiver location. This situation has a $H_{near \to far}$ that approaches or exceeds $H_{far,far}$ in magnitue. The far user's data-rate eduction can be very large.

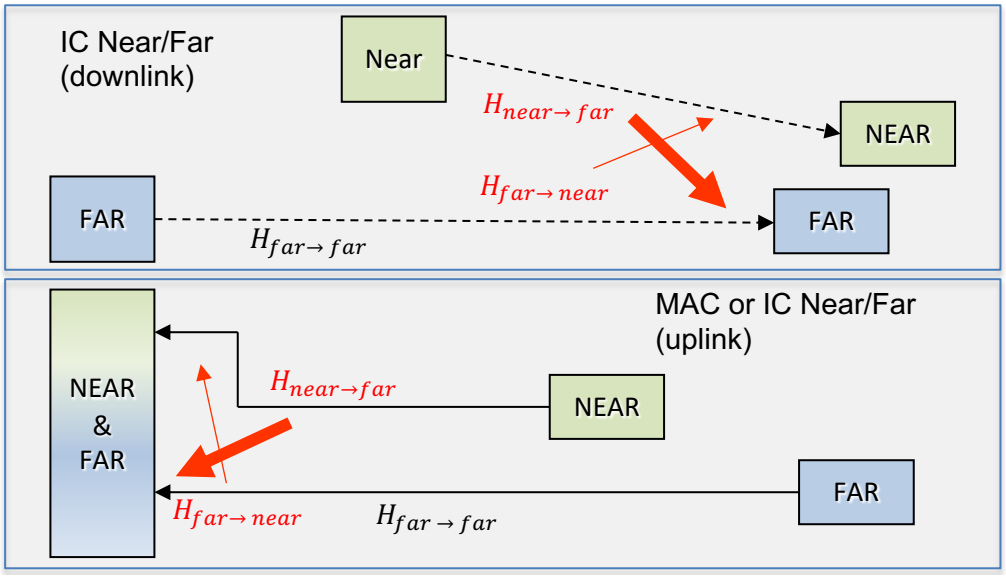


Figure 5.60: **Illustration of Near/Far IC problem.**

The near-far problem occurs often when different systems' users share a common bandwidth. For instance, an "unlicensed" wireless frequency band may allow many systems' simultaneous use (See also Section 2.6.7 on Scheduling and CSMA). It also occurs in wireless systems uplink when a user in one cell may simultaneously transmit on a downlink frequency band used by another cell. Wireline systems with electromagnetically-radiated crosstalk often also experience near-far when one or more transmitters are located much closer to receivers than other transmitters. While the methods of Section 5.6.1 might best address the common-dimension interference, distributed management imposes limitations. Different MIMO systems may spatially target a given direction with intense radiation that overwhelms another systems co-radial user (or one of the spatial sidelobes that always exist with $L < \infty$ might just accidentally be strong at another systems' user-receiver location). Fixed-Margin (FM) water-filling minimizes energy emission to achieve a certain data rate, thereby reducing also interference to other systems. Thus, most practical IW systems use FM. **Power Control** systems are essentially FM IW with a single dimension optimized.

EXAMPLE 5.6.3 *[2-dimensional IW IC example]* Figure 5.61 provides a simple $U = 2$-user IC example helps illustrate the convergence of IW as well as the effects in the following Example. Table 5.1 and Figure 5.59 further detail the steps of IW for the channel of Figure 5.61. Initially user 1 places equal energy on both dimensions. Since the crosstalk transfer into both bands of user 2 is unit gain on this channel, the calculation of the inverse (noise plus crosstalk/ referenced by the unit channel gain) is simply $.1 + .9^2 = .91$ for dimension A and $.1 + .1^2 = .11$ for dimension B. Waterfilling sets user 2's energy on both these bands, plus the normalized noise, equal., which leads to user B energies of 0.6 and 1.4 in the respective dimensions. These now impose crosstalk on user 1. Table 5.1's 4th row then executes water-fill for user 1 and zeros dimension B and places all energy on dimension A. User 2 reacts and places almost all its energy in dimension B (1.81), but still

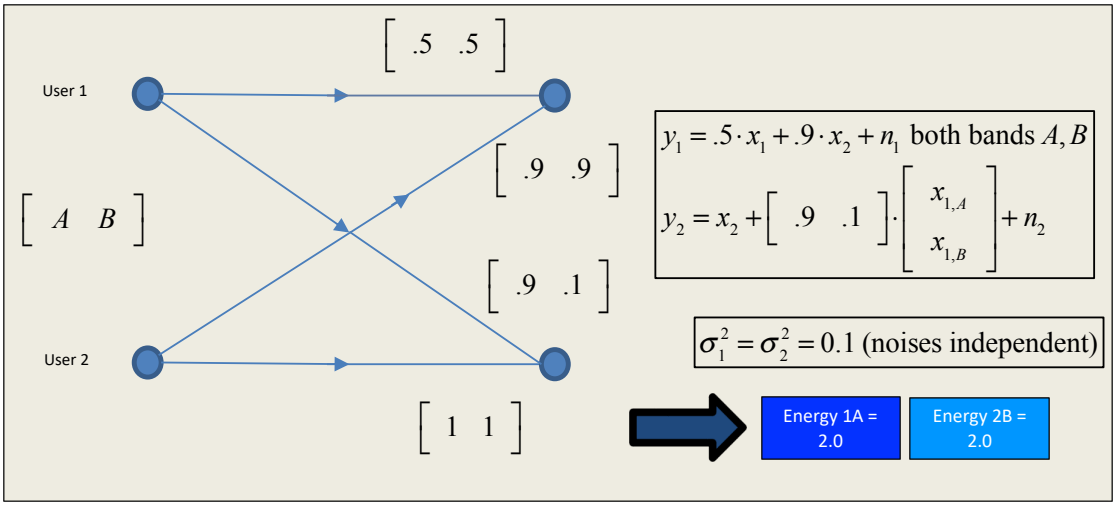


Figure 5.61: **Simple IW Example.**

SIMPLE IW EXAMPLE

	Band A	Band B
User 1	$\mathcal{E}_{1A} = 1$	$\mathcal{E}_{1B} = 1$
User 2	$\frac{1}{g_{2A}} = .1 + (.9)^2 = .91$	$\frac{1}{g_{2B}} = .1 + (.1)^2 = .11$
	$\mathcal{E}_{2A} + .91 = \mathcal{E}_{2B} + .11$ $\mathcal{E}_{2A} + \mathcal{E}_{2B} = 2$ $\mathcal{E}_{2A} = .6 \quad \mathcal{E}_{2B} = 1.4$	
User 1	$\frac{1}{g_{1A}} = \frac{.1 + .6 \cdot (.9)^2}{(.5)^2} = 2.344$	$\frac{1}{g_{1B}} = \frac{.1 + 1.4 \cdot (.9)^2}{(.5)^2} = 4.936$
	$\mathcal{E}_{1A} + 2.344 = \mathcal{E}_{1B} + 4.936$ $\mathcal{E}_{1A} + \mathcal{E}_{1B} = 2$ $\mathcal{E}_{1A} = 2 \quad \mathcal{E}_{2B} = 0$	
User 2	$\frac{1}{g_{2A}} = .1 + 2 \cdot (.9)^2 = 1.72$	$\frac{1}{g_{2B}} = .1 + 0 \cdot (.1)^2 = .1$
	$\mathcal{E}_{2A} + 1.72 = \mathcal{E}_{2B} + .1$ $\mathcal{E}_{2A} + \mathcal{E}_{2B} = 2$ $\mathcal{E}_{2A} = .19 \quad \mathcal{E}_{2B} = 1.81$	
User 1	Remains $\mathcal{E}_{1A} = 2 \quad \mathcal{E}_{2B} = 0$ → IW has converged	
Data rates User 1	$\log_2(1 + 2/2.344) = .89$	0
Total User 1	.89 bits	
Data rates User 2	$\log_2(1 + .19/1.72) = .15$	$\log_2(1 + 1.81/.1) = 4.26$
Total User 2	4.4	
Rate Sum	5.29 bits	

Table 5.1: **Simple IW Example.**

some in dimension A (.19). Subsequent waterfill steps provide no further improvement. User 1 has .89 bits/symbol and user 2 has 4.4 bits/symbol, with sum 5.29 bits/symbol. If instead both users equally energized each dimension with 1 unit of energy, the data rates are lower, summing to .7 bits for user 1 and 1.07 (A) plus 3.18 (B) = 4.25 bits/symbol for user 2. There is a small improvement here, but improvements can be much larger with larger dimensionality

(more degrees of freedom and many users). The IW solution of 5.29 bits/symbol is also better than all user 1 energy on dimension B and all user 2 energy on dimension A, which leads to a sum of almost exactly 5 bits/symbol.

EXAMPLE 5.6.4 *[Bi-directional Crosstalking Wireline Example]* Figure 5.62 illustrates a situation where 25 bi-directional users' channels all have high crosstalk into one another if they try to transmit both directions on the same dimension. Otherwise, all have the same channel and symmetric crosstalk into one another. The channel models are all the same and are

$$H(f,d) = K_{line} \cdot e^{-\alpha(f) \cdot d} \tag{5.432}$$

where the K and the $\alpha(f)$ follow from the standard transmission-line modeling, see for instance Chapter 3 of [5] or the linemod program at this textbook's website. d is the length of the line. The crosstalk model (also [5]) is from each line into adjacent lines:

$$X(f) = .9 \cdot e^{-20} \cdot (25/49)^{0.6} \cdot H(f,d) \cdot d \cdot f^2 \cdot S_x(f) \ , \tag{5.433}$$

Where $S_x(f)$ is the transmit power spectral density. The details of these models are less important than the observation that crosstalk coupling decreases with length, but also increases with frequency squared so more crosstalk at higher frequencies. The channel itself also decreases exponentially with frequency and line length as most engineers would expect, corresponding to a rule of roughly 6dB/1000ft that is often quoted for twisted-pair 24-gauge wires, roughly at all frequencies. No self crosstalk exists between uplink and downlink for the same wire, as this cancels locally using an "echo canceller" in both directions, but of course there is crosstalk from other LIC users at the same end and this is modelled by

$$N(f) = 10^{-13} \cdot (25/49)^{0.6} \cdot S_x(f) \ . \tag{5.434}$$

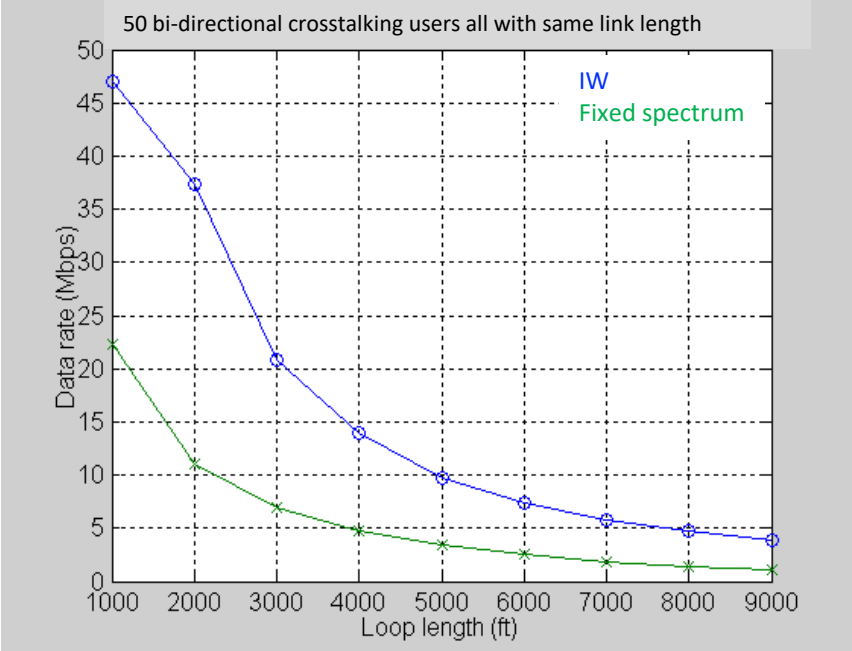


Figure 5.62: Improvement of IW over current static design in symmetric wireline transmission.

In this simulation, the number of users was actually 50 because the lines are all used bi-directionally and of course the upstream and downstream power spectral densities are different for each group of 25 relative to the others, but the same within the groups of 25. but

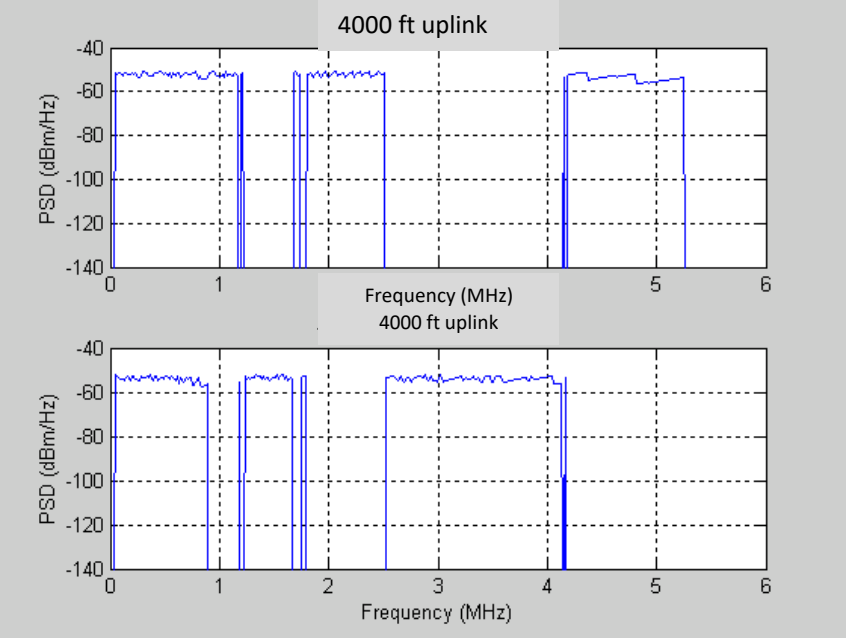


Figure 5.63: **Example PSDs.**

crosstalk into all other loops is presumed. The lower curve uses a 64PAM system with equal energy on all users' dimensions. The upper curve system used IW with up to 4096 4.3125 kHz tones, with no restriction on frequencies used up or down. All links individually and locally use use FM IW and the bandwidth is determined adaptively for each. As is clear, the IW doubles to triples the rate. Such an improvement is solely caused by the better adaptive polite determination of spectrum use. Figure 5.63 shows the up and down spectrum of one user (almost all are the same), which clearly attempts to be an "FDM" solution, but not completely. It is one that is achieved only through distributed management. Coordinated MAC and/or BC management systems will perform yet much better in this situation, providing another factor of 3 roughly in data rate, but the present model is the IC. Nonetheless, the distributed solution is considerably better than the "flat every where" without the IW. Both directions use lower frequencies since crosstalk for lower-frequency tones is weaker for these transmission lines' model. As the frequencies increase, the IW produces a frequency-division-like separation of up and down transmissions because of the model's stronger crosstalk at those frequencies.

5.6.3 Dynamic Spectrum Allocation (DSA)

DSA is a form of consensus management. It allows central spectrum-policy guidance for the distributed-management IC, slightly violating the constraint of no central control, but not allowing the specification of receiver coefficients, channel H_{ij} sharing, or other more coordinated IC management. DSA attempts to gain OSB performance with largely LIC network control. Iterative Spectrum Balancing (ISB) replaces the OSB's intensive exhaustive search with Subsection 5.6.3.1's approximation and often produces OSB's results at a lesser (but still high) complexity a result. OSB/ISB spectrum results often resemble water-fill separately applied in different frequency bands (more generally dimensional groups). These lead to the iterative multi-level water-filling methods of Section 5.6.3.2 that obtain essentially the highest level of performance at a cost essentially no greater than that of IW and with a highly distributed implementation (although a very small amount of central policy appears, which is the message-passing

of consensus-based management).

5.6.3.1 Iterative Spectrum Balancing (ISB)

Iterative Spectrum Balancing (ISB) was simultaneously introduced by Yu and Liu [2]. ISB replaces OSB's exhaustive energy-search step by an iterative approximate algorithm. The approximation optimizes each user separately in a sub step where M values of energy for that user are compared in terms of L_n values while the energies for all other users are held constant. The algorithm cycles through all users (each holding all the others constant). Convergence is assured because the method reduces L_n at each step, and usually in far less than M^U steps. Thus, the complexity reduces from $O(NU(M^U)$ in OSB to $O(NU^2M)$. By comparison, IW requires $O(NU)$ and thus is still considerably less complex yet than ISB, which however is much less complex than OSB and approximates it.

Observation in practice of OSB and its approximations suggest that the best spectra of all users look like water-filling, but in separate bands with different water levels. This makes a strong intuitive theoretical sense since there is no successive decoding and indeed a designer would expect simultaneous water-filling appearance if all other users are noise. However, the water-levels are different in different frequency bands, or more generally across different dimensional sets or clusters. The different levels intuitively correspond to different user priorities or weightings.

5.6.3.2 Multi-Level (Iterative) Water Filling

This text's multi-level iterative water-filling[92] observe that water-filling indeed would appear optimum in certain clusters of dimensions between all users energizing those dimensions. The water-levels can vary, and the clusters of dimensions (frequency bands when there is only one antenna per receiver) can vary, so MLIWF focuses on finding the levels and the clusters (which are cut-off frequencies between the different bands in the single-antenna-receiver case). Summarizing the realization difficulties with all the methods, except iterative water-filling:

1. OSB and ISB require central spectra control and synchronization (as with the MAC or BC) so that crosstalk only occurs between systems independently on each tone. Such synchronization may be difficult and thus the crosstalk will be a function of other adjacent tones (and the spectra of unsynchronized tones only falls as $1/f$ so the other tones' crosstalk will be significant – even windowing can ony reduce this effect slightly as in Section 4.9).

2. In unsynchronized systems, the power-spectral density roll-off prevents very large reductions in power-spectral density on adjacent tones, which is exactly the type of spectra that OSB methods typically produce.

These difficulties encourage the development of more distributed autonomous spectral balancing among the multiple users.

[92] Introduced by Dr. A. Chowdhery in her 2012 Stanford dissertation.

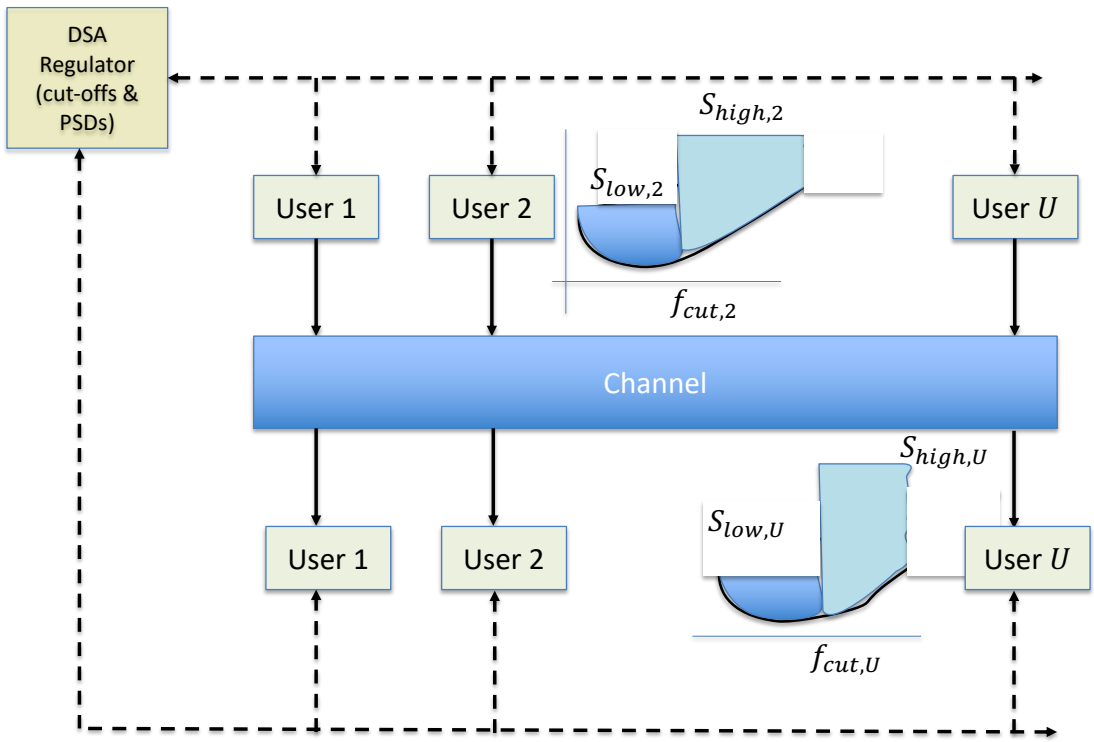


Figure 5.64: **Illustration of 3-user two-band multi-level water filling.**

5.6.3.3 The Multi-Level IW method

Ultimately, the key DSA parameter is a separation or "cut-off" frequency $f_{cut,u}$ between a band of tones in which loading to the maximum level is desired more for politeness and other bands that are preferred less for politeness. Figure 5.64 illustrates the basic concept where a control unit informs users that can easily meet their rate targets are registered and informed to be extra polite. In effect ML IW imposes a maximum (flat) PSD level (which is not quite water-fill but often close as in Chapter 4's discussion around the Separation Theorem) at two levels $S_{low,u}$ and $S_{high,u}$. In effect, each user water-fills in two bands with different water levels. Users that have no ability to be polite are not so asked to be extra polite (and so execute water-filling with one level).

An extra-polite user might have two water-filling bands (the extension to more than two bands is trivial):

$$\lambda_{u,1} = \mathcal{E}_{u,n} + \frac{\Gamma}{g_{u,n}} \forall\, n \in \mathcal{F}_1 \tag{5.435}$$

$$\lambda_{u,2} = \mathcal{E}_{u,n} + \frac{\Gamma}{g_{u,n}} \forall\, n \in \mathcal{F}_2 \quad . \tag{5.436}$$

The DSA regulator determines two bands $\mathcal{F}_1$ and $\mathcal{F}_2$ and tells a (or some) user(s) to be extra polite. on the S_{low} side of its cut-off frequency. The local user transmitter itself determines the two water-filling levels $\lambda_{u,1}$ and $\lambda_{2,u}$ and includes any specified power-spectral density limits. The band with the higher

index 2 is preferred for loading and its water-level is determined by first solving an overall water-filling problem (with one level), and then moving bits one-by-one from band 1 to band 2 until any power spectrum density limit constraints in band 2 would be violated (or until the overall power constraint would be exceeded). This preferred band then effectively water-fills (approximately to the degree that LC approximates water-filling) with a higher water-filling level. The polite band also water-fills but to a lower water-filling level. The energy (and information/coding with gap Γ_u) allocation continues until the b_u and any margin γ_u is achieved. If the system must use the polite band above the requested PSD, then it does and the DSA regulator observes this on a next pass. The cut-off frequency is important. Minimal central control occurs, because the controller would run the multi-level IW for each user for various estimates of the cut-off frequency (or exhaustive search) and then recommends only each user's cut-off accordingly. The users otherwise retain full automony. If noises change such that two different water levels cannot be maintained, then the local user strategy returns to normal water-filling.

The concept easily extends to 3 or more bands with the most preferred band being filled first, then the next highest index and so forth. A simple method to communicate preferences simply sends slight differences in the power-spectral density (PSD) mask to all users (via central control or by message passing sometimes called cell coloring). A change in PSD mask from one tone n to $n+1$ simply implies that the preference index changes and the band with higher PSD mask should have a higher index. Such a method leaves all bands water-filling (and since we know IW converges under wide conditions, then this ML IW will also converge in those same situations). Inspection of the final results for power spectra in Section 5.6.3 certainly confirms that a simple cut-off could easily have been used to obtain the same results with IW. Thus, the only central control required is an indication to the user to load extra polite favoring those bands for which its power spectrum density limit is higher. The power spectrum density limit can be preset in all modems or might possibly have been distributed in a quasi-static upon-start type initialization within the network.

5.6.3.4 ML IW examples and results

The key advantages of the ML IW approach are:

1. distributed low-complexity implementation. There are consequently also no convergence issues, choice of thresholds, choice of elliptic versus sub-gradient, etc complexities even at the controller if one exists.

2. The individual user modems retain the ability to react to changes in the noise or channel. Thus, rapid direct reaction to changes allows robust operation in the presence of any kind of situation not originally anticipated by the controller and/or modems.

3. OSB's tacit "all-are-synchronized" assumption is no longer necessary. The modems react according to the actual noise present and not some presumed synchronized-crosstalk presumption in an optimization algorithm.

4. In the next example, the performance of ML IW matches OSB.

5. If $L > U \cdot \overline{N}$, It approaches optimal CIC.

A few examples will illustrate the advantages.

EXAMPLE 5.6.5 *[Near/Far intererence]* A IC with two transmitter locations appears in Figure 5.65. A noise floor of -140dBm/Hz models background noise. Four users from the NEAR location constitute a strong interference to the single user from the FAR location.

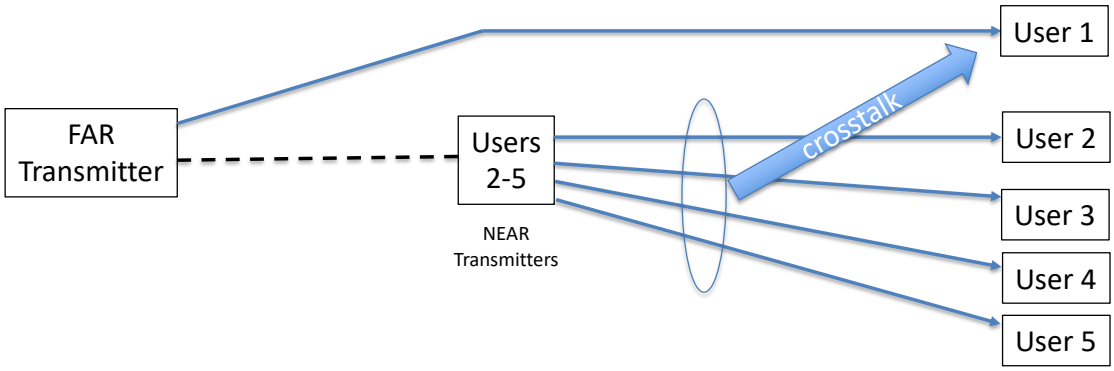


Figure 5.65: **Near Far IC example.**

Figure 5.66 provides the optimal spectrum levels for this multiuser channel. The crosstalk transfer function for the channels again follows Equation (5.433). The far path has length 4km and the near paths have 2km. From this example, it is clear that the masks are very sharp in transition and nearly impossible for implementation. They also illustrate the basic concentration of short-link energy at higher frequencies. T Nonetheless, the shorter links best occupy the very lowest frequencies where the crosstalk is low and the very highest frequencies where they do not inflict harm on the high frequencies that could not be used by the long link.

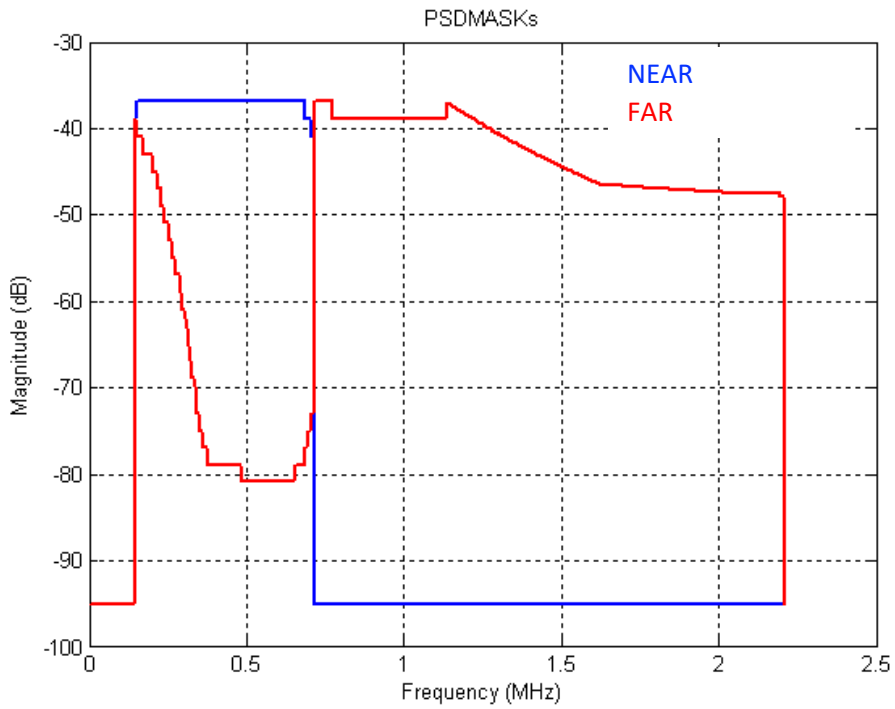


Figure 5.66: **Spectra for situation in Figure 5.65.**

By contrast, the use of ML IW would place 3 bands, low, medium, and high where low and high

can have the same power spectrum density limit levels and the medium band should have slightly lower power spectrum density limit. The low and high bands are then preferred on the 4 short loops. The cut-offs are roughly 300 kHz and 700 kHz.

Figure 5.67 shows the ML IW and OSB achievable rate regions for the situation in Figure 5.65. These two regions are the two largest, which are virtually equal. IW with no power spectrum density limits also appears for reference. Each of the intermediate curves corresponding to relaxing the OSB power spectrum density limit in Figure 5.66 by successive increments of 6 dB to allow for both robustness to channel change and possibility of implementation with less sharp transition bands. ML IW has a power spectrum density limit in the middle band that is just 1 to 2 dB lower than the low and high bands, and thus is very feasible. These show that reasonable relaxation of the central control is highly sensitive. However, the ML IW works without need for such sensitivity.

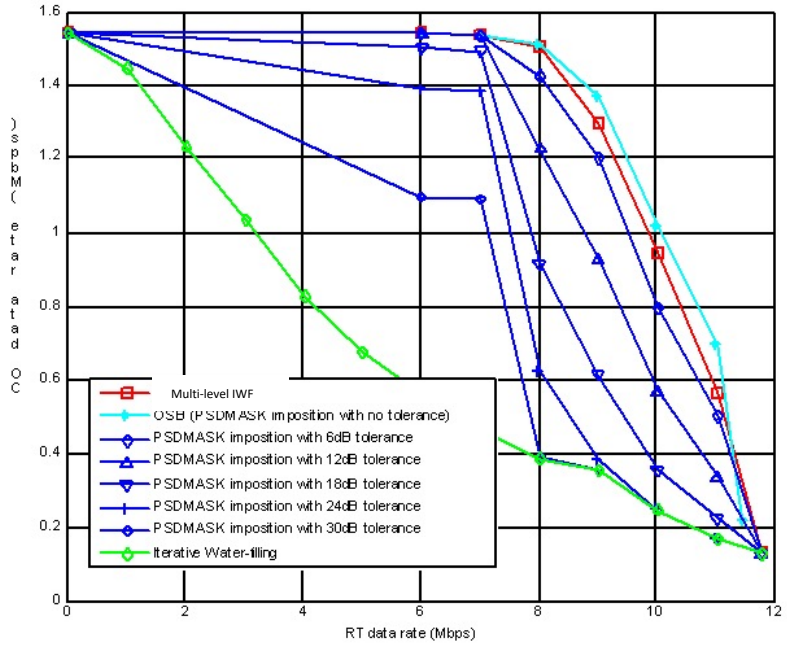


Figure 5.67: **Rate Regions with various levels of tolerance on the OSB power spectrum density limits.**

5.7 Statistically Characterized GDFEs

not yet added.

Exercises - Chapter 5

5.1 channel singularity elimination (8 pts)

(Subsection 5.1.1.1) Given a binary antipodal (2 PAM) white input with $\bar{\mathcal{E}}_{\boldsymbol{x}} = 1$ transmitted through the channel with H matrix

$$H = \begin{bmatrix} .9 & 1 & 0 & 0 \\ 0 & .9 & 1 & 0 \\ 0 & 0 & .9 & 1 \end{bmatrix} \quad , \tag{5.437}$$

and with the input-modulator-matrix choice of $C = I$:

a. (2 pts) Find the projection matrix $P_{\tilde{M}}$.

b. (2 pts) Determine the 4×3 matrix $\tilde{C}$ and the input $\tilde{\boldsymbol{u}}$ that characterize the pass-space description of this channel.

c. (2 pts) Determine the autocorrelation matrix $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}$.

d. (2 pts) How much energy is lost into the null space of this channel?

5.2 Singularity Elimination short problems. (12 pts)

(Subsection 5.1.1.1) For each of the following channels and inputs, determine N, $\tilde{N}$, N^*, and ν.

a. (2 pts) $H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ and $R_{\boldsymbol{xx}} = I$.

b. (2 pts) $H = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ and $R_{\boldsymbol{xx}} = I$.

c. (2 pts) $H = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 1 & 3 & 2 & 4 \\ 4 & 3 & 5 & 4 \end{bmatrix}$ and $R_{\boldsymbol{xx}} = I$.

d. (2 pts) $H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ and $R_{\boldsymbol{xx}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

e. (2 pts) $H = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ and $R_{\boldsymbol{xx}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

f. (2 pts) $H = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 1 & 3 & 2 & 4 \\ 4 & 3 & 5 & 4 \end{bmatrix}$ and $R_{\boldsymbol{xx}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

5.3 MMSE Estimation (8 pts)

(Subsection 5.1.2) Two zero-mean complex Gaussian random variables have probability distributions p_x and p_y with joint probability distribution $p_{x,y}$ and nonsingular autocorrelation matrix

$$R_{x,y} = \begin{bmatrix} \mathcal{E}_x & r_{xy} \\ r_{xy}^* & \mathcal{E}_y \end{bmatrix} \tag{5.438}$$

The minimum mean-square estimate of x given y has variance $\sigma^2_{x/y}$. The orthogonality principle of Appendix 3.A may be useful throughout.

a. (1 pt) Find $\sigma^2_{x/y}$ in terms of $\mathcal{E}_x$, r_{xy} and $\mathcal{E}_y$.

b. (1 pt) Relate the conditional entropy $H_{x/y}$ to $\sigma^2_{x/y}$ and therefore to the MMSE estimate.

c. (2 pts) Interchange the roles of x and y in the results in Parts a and b and compare the SNRs for the two results.

d. (1 pt) Relate the mutual information to the SNR of Part c.

e. (3 pts) Suppose y becomes a complex vector $\boldsymbol{y}$ but x remains a scalar with $\boldsymbol{y} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \boldsymbol{n}$ where $\boldsymbol{n}$ has independent zero-mean Gaussian components all with variance σ^2. Does your answer to part d change? How might you interpret SNR_x in this case in terms of equalizers you know from Chapter 3?

5.4 *DFE - MT connection.* (8 pts)

(Subection 5.2.3 ; Section 2.5 may also be helpful, as well as Subsection 3.12.1) This problem explores the connection between DFE and multi-tone-like concepts for transmitter optimization, essentially extrapolating the Circulant DFE's limit to Chapter 3's MMSE-DFE and DMT's limit to MT.

a. (4 pts) Show that the following are all equivalent expressions for $\mathcal{I}$ for a fixed sampling period $1/T$ equal to the $1/\bar{N}$ times a limiting infinite-length symbol period. (Complex baseband is assumed so the extra factor of 1/2 in front of the log does not appear.) $\bar{N}$ can be assumed to always be an even integer.

$$\widetilde{\mathcal{I}} = \frac{T}{2\pi} \int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} \log_2(1 + SNR(\omega)) \cdot d\omega = \frac{T}{2\pi} \int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} \log_2(1 + SNR_{MFB} \cdot \left|Q(e^{-\jmath\omega T})\right|) \cdot d\omega$$

$$= \frac{T}{2\pi} \int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} \log_2(1 + SNR \cdot \left|H(e^{-\jmath\omega T})\right|^2 \left|\Phi(e^{-\jmath\omega T})\right|^2) \cdot d\omega$$

where $H(e^{-\jmath\omega T})$ is the transform of the sampled channel. This problem assumes that the sampling rate is high enough so that there is no aliasing. Relate these formulas to each of MT and the MMSE-DFE.

b. (4 pts) The integral in Part a can be approximated by the following sum for large $\bar{N}$:

$$\widetilde{\mathcal{I}} = \frac{T}{2\bar{N}T} \cdot \sum_{-\bar{N}/2+1}^{\bar{N}/2} \log_2 \left[1 + SNR \cdot \left|H(e^{j2\pi n/2\bar{N}T})\right|^2 \cdot \left|\Phi(e^{j2\pi n/2\bar{N}T})\right|^2\right]$$

Optimize the above approximation and show that the water-fill equations for this problem are:

$$\frac{1}{SNR \cdot \left|H(e^{j2\pi n/2\bar{N}T})\right|^2} + \left|\Phi(e^{j2\pi n/2\bar{N}T})\right|^2 = \text{a constant}$$

5.5 Input Singularity Elimination. (7 pts)

(Subsection 5.1.1.2) The real-baseband channel-input autocorrelation after channel-singularity elimination and input design is

$$R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} = \begin{bmatrix} 1.0 & 0.5 & 1.0 \\ 0.5 & 1.0 & 0.5 \\ 1.0 & 0.5 & 1.0 \end{bmatrix} \quad . \tag{5.439}$$

a. (1 pts) What is N^* for this channel input? Is $\bar{N}^* = N^*$ on this channel?

b. (2 pts) Find $\tilde{Q}$ and $P_{\tilde{Q}}$.

c. (4 pts) Find at least two A matrices that accept N^*-dimensional (white, so $R_{\boldsymbol{vv}} = I$) input $\boldsymbol{v}$ such that the output autocorrelation is $R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} = A \cdot A^*$. In each case, relate the components of $\boldsymbol{v}$ to those of $\tilde{\boldsymbol{x}}$.

5.6 Complex Baseband Canonical Triangles (13 pts)

(Subsection 5.1.2) This problem illustrates easy derivation of a number of important mathematical relationships in the GDFE through the canonical triangles associated with the canonical forward and backward channel models, which are found in Figure 5.8.

a. (2 pts) The forward channel in Figure 5.8 has two right subtriangles embedded within it. From the one describing a decomposition of the noise vector $\boldsymbol{n}'$, one notes that the MMSE estimation filter for $\boldsymbol{n}'$ given $\boldsymbol{z}$ is $R_{\boldsymbol{n}'\boldsymbol{z}} R_{\boldsymbol{z}\boldsymbol{z}}^{-1} = (I - R_f R_b)$. Use this relation, and note that $R_{\boldsymbol{n}'\boldsymbol{z}} = R_{\boldsymbol{n}'\boldsymbol{n}'} = R_f$, to prove that $R_f^{-1} = R_b + R_{\boldsymbol{z}\boldsymbol{z}}^{-1}$.

b. (2 pts) Use the triangle for decomposition of the error vector in the backward triangle to note that $R_b = -R_{\boldsymbol{e}\boldsymbol{n}'} R_{\boldsymbol{n}'\boldsymbol{n}'}^{-1}$ to prove easily that $R_{\boldsymbol{e}\boldsymbol{e}} = R_b$.

c. (2 pts) Follow parts a and b to prove that $R_b^{-1} = R_f + I$.

d. (2 pts) Use a convenient triangle to show that $R_{\boldsymbol{e}\boldsymbol{n}'}{}^* = -R_f R_b$.

e. (2 pts) Prove that $R_b R_f = (I - R_b)$.

f. (3 pts) The matrix SNR for a matrix MMSE-LE is $\boldsymbol{SNR} = R_{\boldsymbol{vv}} \cdot R_{\boldsymbol{e}\boldsymbol{e}}^{-1}$. Use the Pythagorean relationship $I = R_b \cdot R_{\boldsymbol{z}\boldsymbol{z}} \cdot R_b + R_{\boldsymbol{e}\boldsymbol{e}}$ to prove the "similar" triangles relationship $\boldsymbol{SNR}$ is also equal to $R_{\boldsymbol{n}'\boldsymbol{n}'}^{-1} R_{\boldsymbol{z}\boldsymbol{z}}$. Why is $\mathcal{I}(\boldsymbol{x};\boldsymbol{y}) = \mathcal{I}(\boldsymbol{v};\boldsymbol{z}) = \log_2 |\boldsymbol{SNR}|$?

5.7 Complex Baseband Matrix Bias. (8 pts)

(Subsection 5.1.2) For the matrix additive Gaussian noise channel, the bias relationships for the matrix MMSE-LE and the GDFE generalize easily from their scalar counterparts. Let $\hat{\boldsymbol{v}} = R_b \cdot \boldsymbol{z}$ and $\boldsymbol{SNR} = R_{\boldsymbol{e}'\boldsymbol{e}'}^{-1}$. The forward and backward are as in Section 5.1.

a. (1 pt) Show that $E[\hat{\boldsymbol{v}}/\boldsymbol{v}] = R_b \cdot R_f \cdot \boldsymbol{v} \neq \boldsymbol{v}$ unless $\boldsymbol{v} = 0$.

b. (1 pt) Show that $E[\boldsymbol{e}/\boldsymbol{v}] = (I - R_b \cdot R_f) \cdot \boldsymbol{v}$.

c. (2 pts) Show that unbiased matrix SNR defined as $\boldsymbol{SNR}_{unb} \triangleq R_{\hat{\boldsymbol{v}}\hat{\boldsymbol{v}}} \cdot R_{\boldsymbol{e}'\boldsymbol{e}'}^{-1}$ is equal to $\boldsymbol{SNR} - I$.

d. (2 pts) Show that another expression for the unbiased matrix SNR is $\boldsymbol{SNR}_{unb} = R_{\boldsymbol{n}'\boldsymbol{n}'}^{-1} \cdot R_{\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}}$ where $\hat{\boldsymbol{z}} = R_f \cdot \boldsymbol{v}$.

e. (2 pts) Show that $\mathcal{I}(\boldsymbol{v};\boldsymbol{z}) = \log_2 |\boldsymbol{SNR}|$ for complex signals.

5.8 Non-white and white inputs to the $1 + .9D^{-1}$ channel. (16 pts)

From Example 5.1.2, assume that the $N + \nu$-dimensional binary PAM input symbol possibilities are mutually independent and the eight possible 3-dimensional channel-input vectors are $\boldsymbol{x} = [\pm 1,\ \pm 1,\ \pm 1]$. The noise-equivalent channel is

$$\frac{1}{\sigma} \cdot \widetilde{H} = \frac{1}{\sqrt{.181}} \cdot \begin{bmatrix} .9 & 1 & 0 \\ 0 & .9 & 1 \end{bmatrix} = \begin{bmatrix} 2.1155 & 2.3505 & 0 \\ 0 & 2.1155 & 2.3505 \end{bmatrix} \quad ,$$

with singular value decomposition:

$$\underbrace{\frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}_{F} \underbrace{\begin{bmatrix} 3.8694 & 0 & 0 \\ 0 & 2.2422 & 0 \end{bmatrix}}_{\Lambda} \underbrace{\begin{bmatrix} .3866 & -.6671 & -.6368 \\ .8161 & -.0741 & .5731 \\ .4295 & .7412 & -.5158 \end{bmatrix}^*}_{M^*} \quad . \tag{5.440}$$

The first two columns of M, or $\widetilde{M}$, span the pass space of the channel, so $P_{\tilde{M}} = \widetilde{M} \cdot \widetilde{M}^*$. The original PAM modulator corresponds to $C = I$, so

$$\tilde{C}_{temp} = P_{\widetilde{M}} \cdot C = P_{\widetilde{M}} = \begin{bmatrix} .5945 & .3649 & -.3285 \\ .3649 & .6715 & .2956 \\ -.3285 & .2956 & .7340 \end{bmatrix} \quad , \tag{5.441}$$

which in this special case of white PAM modulation corresponds to a rank-2

$$R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}} = P_{\tilde{M}} \cdot I \cdot P_{\tilde{M}} = \tilde{C}_{temp} \quad . \tag{5.442}$$

a. (4 pts) Complete the steps in Figure 5.5 for this input to find $R_{\boldsymbol{u}\boldsymbol{u}}$.

b. (4 pts) Compute a formula for the channel inputs $\tilde{\boldsymbol{x}} = \tilde{C} \cdot \boldsymbol{u}$ using the O matrix from Part a. Use it to compute the 8 $\boldsymbol{u}$ and associated $\tilde{\boldsymbol{x}}$ values that correspond to the 8 possible ± 1 2-PAM inputs? Are the equal? Why or Why not? Do this first without fixmod program, then check results using that program.

c. (2 pts) How much of $R_{\boldsymbol{x}\boldsymbol{x}}$'s energy is lost into null space?

d. (2 pts) Continuing, what are the 8 possible noise-free channel outputs?

e. (2 pts) What kind of detector would achieves lowest P_e for this (backward canonical) channel? Please provide detector-type answers for both uncoded 2-PAM and a more general description for the presumed use of a zero-gap code.

f. (2 pts) Compare $2^{2 \cdot \overline{I}(\boldsymbol{x};\boldsymbol{y})}$ to SNR_{GDFE} for some $\boldsymbol{v}$ that whitens this $\boldsymbol{u}$. What can you say about error probability if the original 2-PAM $\boldsymbol{x}$ is used in this GDFE?

5.9 GDFE's and White Inputs. (15 pts)

(Subsection 5.1.3) A real baseband linear-ISI channel with discrete-time response $1 + D$ has AWGN with $\frac{\mathcal{N}_0}{2} = .001$. The transmission scheme uses 8PAM inputs on all dimensions that are independent with $\mathcal{E}_{\boldsymbol{x}} = 1$.

a. (4 pts) For $N = 3$ and $\nu = 1$, find the CDFE on this channel, including the unbiased G, feedforward filter matrix , mean-square-whited-matched filter matrix, $\text{SNR}_{cdfe,u}$ and $\bar{P}_e$ with equal-energy 16-level PAM on each of the 3 used subchannels' dimensions.

For this problem, the geometric SNR can be considered the same on all subchannels, although that is only true asymptotically with CDFE. (Error propagation may be ignored.)

b. (2 pts) Compare Part a's error probability with the asymptotic error probability for an 8 PAM MMSE-DFE with $N_f = 3$ and $N_b = 1$ on this same channel. The program dfecolorsnr.m from Chapter 3 may be helpful. Comment on why the latter appears better.

c. (4 pts) Repeat Part a for a GDFE (with a guard band of $\nu = 1$ but not necessarily a cyclic prefix) that instead places 4/3 units of energy on each of the 3 useful input dimensions. Why does this work better than Part a? Compare with part b and comment as to which system is actually better.

d. (5 pts) Repeat Part c for N increasing to 10, 20, 100, 200 except only compute the unbiased SNR (and not all the matrices). Estimate to which value this SNR converges as N becomes infinite. What is the number of levels of PAM that would be used as N increases at $\bar{P}_e = 10^{-6}$. Which system in this problem is best and why?

5.10 Comparing the GDFE and the MMSE-DFE. (14 pts)

(Subsection 5.1.3) For the $1 + .9 \cdot D^{-1}$ with $SNR_{mfb} = 10$ dB, the GDFE performs worse than the MMSE-DFE for $N = 2$ and $\nu = 1$. One might conjecture that this is caused by a large $(\nu/(N+\nu) = 1/3)$ bandwidth loss due to too small N. This problem investigates increasing N and assumes the student has access to the dfecolor.m Matlab program.

Initially let $R_{\boldsymbol{x}\boldsymbol{x}} = I$ and let N increase.

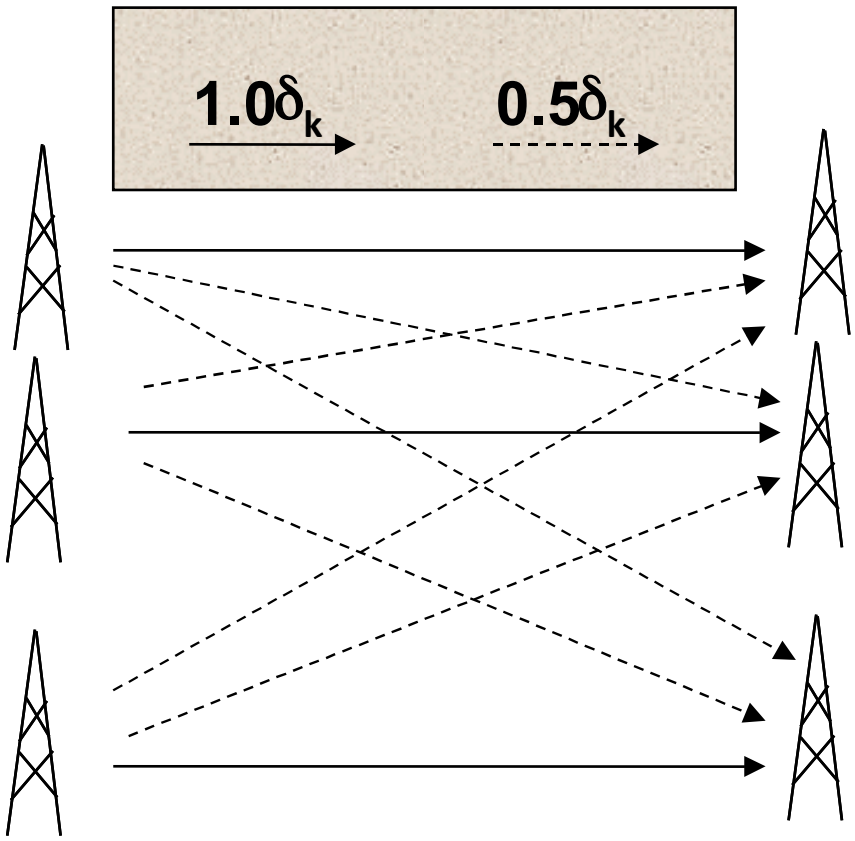


Figure 5.68: MIMO Antennas for Problem 5.11

a. (5 pts) Use Matlab to compute the GDFE performance for variable N and $\nu = 1$. Plot $SNR_{gdfe,u}$ versus N for $N = 2, ..., 40$. (Recall $SNR_{gdfe} = 2^{2 \cdot \overline{\mathcal{I}}(X,Y)}$. Is there an easy way to compute $\overline{\mathcal{I}}(X,Y)$?, and think of vector coding)

b. (2 pts) Repeat Part a for Chapter 3's MMSE-DFE and plot $SNR_{mmsedfe,u}$ versus N with the number of feedback taps fixed at $\nu = 1$. Compare with part a results by plotting on the same graph.

c. (2 pts) Plot Vector Coding's performance with waterfilling, which equals the performance of the GDFE with optimized input, for $N = 2, ..., 40$ with $\nu = 1$ and compare to parts (a) and (b).

d. (5 pts) Draw conclusions from your plots in parts (a), (b), (c). This part has a large number of points for a reason, so perhaps think about any initial transmission-system conditions in terms of what was being plotted.

5.11 Cyclic Antennas (7 pts)
(Subsection 5.2.3)

Figure 5.11's 6-real-dimensional cyclic channel has $H = \begin{bmatrix} 1 & .5 & .5 \\ .5 & 1 & .5 \\ .5 & .5 & 1 \end{bmatrix}$ with noise variance (2-sided power spectral density) equal to .01 and a gap of 0 dB, the energy per real dimension is 1. Hint: Cholesky factorization of a matrix that is nearly white is trivial.

a. (2 pts) Find a water-filling input covariance matrix $R_{\boldsymbol{xx}}$. Also find the corresponding Cholesky factorization $R_{\boldsymbol{xx}} = G_x \cdot S_x \cdot G_x^*$.

b. (2 pts) Find the forward and backward canonical-channel-characterizing matrices R_f and R_b^{-1}.

c. (2 pts) Find the feedback matrix G and the combined matched-filter feedforward filter matrix.

d. (1 pt) What is the SNR of this CDFE-equalized channel and what is the corresponding largest number of bits per dimension that can be transmitted at $P_e = 10^{-6}$?

5.12 MAC-like Vector DMT (9 pts)

(Section 5.2.1) Two transmission lines share a common cable and are not shielded with respect to one another and so thus experience crosstalk as well as ISI, as in Figure 5.69. The two transmitters are not co-located, but each uses DMT with the same clocks and cyclic-prefix lengths - that is the symbols are aligned on the two channels. The inputs to the transmit IFFT's are $X_{1,n}$ and $X_{2,n}$ respectively, and each such component is independent of all others (including all frequency indices on the other input). The input energy per dimension for each tone of these two transmitters on each such input is denoted $\mathcal{E}_{1,n}$ and $\mathcal{E}_{2,n}$. The two outputs are received in the same box at the end of the cable, so that coordinated signal processing/detection of the two input streams (as if they were one) is possible. The discrete-time matrix impulse response of the channel is

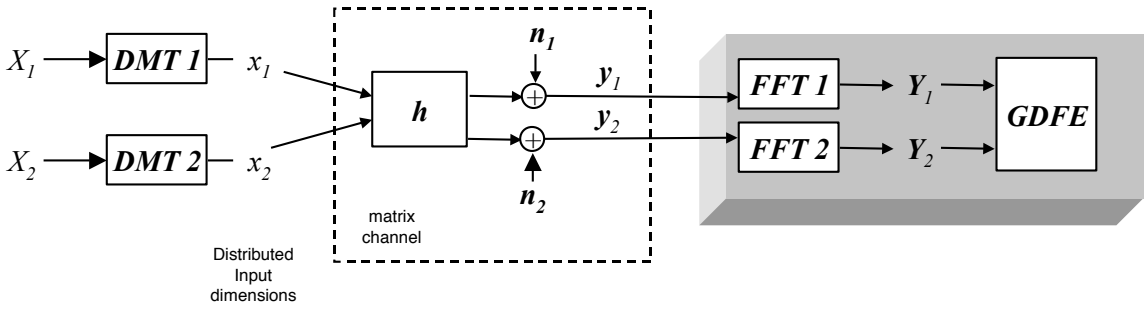


Figure 5.69: Illustration of bonded DMT for Problem 5.12

$$\boldsymbol{h}_k = \begin{bmatrix} h_{11,k} & h_{12,k} \\ h_{21,k} & h_{22,k} \end{bmatrix} \tag{5.443}$$

where each entry is an FIR channel and

$$h_{ij,0} + h_{ij,1} \cdot D + ... + h_{ij,\nu_{ij}} \cdot D^{\nu_{ij}} \quad . \tag{5.444}$$

and $h_{ij,0} \neq 0$ and $h_{ij,\nu_{ij}} \neq 0$. The channel output is thus

$$\boldsymbol{y}(D) = \begin{bmatrix} y_1(D) \\ y_2(D) \end{bmatrix} = \boldsymbol{h}(D) \begin{bmatrix} x_1(D) \\ x_2(D) \end{bmatrix} + \boldsymbol{n}(D) \quad , \tag{5.445}$$

where $R_{\boldsymbol{nn}}(D) = \sigma^2 \cdot I$.

a. (1 pt) What is the shortest-length cyclic prefix that can be used to ensure there is no intersymbol interference on or between any of the discrete-time paths?

b. (2 pts) Write a relationship for each DMT-output (i.e. tone n for both of the outputs $Y_{1,n}$ and $Y_{2,n}$) of a receiver in terms of ALL the tone inputs to both channels. What is the dimension of the channels created?

c. (4 pts) For the resultant channels in part b, determine GDFE settings in terms of the given quantities above and $\boldsymbol{H}_n$ and/or its elements? What happens if any of the inputs have zero energy on a given frequency? You should use the defined quantities $r_{1,n} = |H_{11,n}|^2 + |H_{21,n}|^2$, $r_{2,n} = |H_{22,n}|^2 + |H_{12,n}|^2$, and $r_{12,n} = H^*_{11,n} \cdot H_{21,n} + H^*_{21,n} \cdot H_{22,n}$ to simplify derived expressions. The total feedforward-filter/matched-filter may be expressed as a product of 3 matrices, and not simplified further.

d. (2 pts) If the two inputs have mutual information $\overline{\mathcal{I}}_1(\boldsymbol{x}_1; \boldsymbol{y})$ and $\overline{\mathcal{I}}_2(\boldsymbol{x}_2; \boldsymbol{y}/\boldsymbol{x}_1)$, what is the overall SNR of the DMT/GDFE (bonded DMT) system?

5.13 Generalized Cholesky. (10 pts)

(Subsection 5.2.4.1) Given the discrete-time channel $H(D) = 1 + .8 \cdot D$ with $\bar{\mathcal{E}}_{\boldsymbol{x}} = 1$ and $\frac{\mathcal{N}_0}{2} = .164$ and a gap of 0 dB:

a. (4 pts) Find the best $R_{\boldsymbol{xx}}$ and corresponding $\text{SNR}_{gdfe,u}$ for $N = 4$ and $\nu = 1$.

b. (5 pts) Compute all the matrices in the GDFE for your answer in part a if the designer is allowed to construct a white input and wants a triangular or "causal" implementation. Provide a block diagram of both transmitter and receiver.

c. (1 pt) Why can't a CDFE or DMT obtain quite this same performance with $N = 4$ and $\nu = 1$?

5.14 Some review questions. (6 pts)

(Section 5.3)

a. (1 pt) For what types of $R_{\boldsymbol{xx}}$ can a CDFE be implemented? Equivalently, for what types of channel matrices H?

b. (1 pt) For what types of $R_{\boldsymbol{xx}}$ can a GDFE be implemented? Equivalently, what are the restrictions on H?

c. (1 pt) What is the characteristic that defines a "canonical channel"?

d. (1 pt) What is the difference between "canonical factorization" and a "canonical channel"?

e. (1 pt) Why does the appropriate use of the CDEF result allow suboptimum equalizers to play at highest performance levels?

f. (1 pt) Generally on a channel with à prior unknown or severe intersymbol interference, if you had the option to implement DMT or a CDFE/MMSE-DFE, what would you choose if complexity/cost was your dominant concern? Why? (Give two reasons).

5.15 A multi-band design.

A channel has $H(D) = 1 - D^3$ with $\frac{\mathcal{N}_0}{2} = .01$ and $\bar{\mathcal{E}}_{\boldsymbol{x}} = 1$.

a. (5 pts) Suppose this channel were modelled as 3 independent $1 - D$ channels in parallel. Find the CDFE for $N = 8$ and $\nu = 1$ and the best data rate with $\Gamma = 3$ dB.

b. (1 pt) Now instead model this channel directly with $N = 24$ and $\nu = 3$. How may independent CDFE bands are there, $M =?$.

c. (5 pts) Design the CDFE and compute its SNR for the overall design in part b. You need not provide the feedforward and feedback matrices, but do provide the set of SNR's for all the subchannels implemented. Use a water-filling rate-adaptive input design for the best covariance.

d. (2 pts) Can you guess an eventual setting for the CDFE's feedforward and feedback sections from part c. If so, provide it.

e. (1 pt) What is the SNR of a DMT with $N = 24$ and $\nu = 3$?

f. (3 pts) Draw a block diagram of the DMT implementation and of the CDFE implementation and compare (you can label boxes with letters and do not have to provide the settings for each and every matrix entry).

5.16 A Tonal Channel (16 pts)

A complex baseband 2×2 channel D-Transform is

$$H(D) = \begin{bmatrix} 1+D & -.5-.4D \\ .9 & 1-D \end{bmatrix}$$

with white (spatially and in frequency) noise (per-tone) variance $\mathcal{N}_0 = .01$ and the energy per input sample is $= 1$. The energy per each time-domain symbol (sum over both inputs/sample) is thus $\mathcal{E}_x = 2$.

a. (5 pts) What is ν for this matrix channel for the time-domain symbols? For $\overline{N} = 8$ using the same sampling rate/bandwidth for the channel, find each tonal noise-whitened channel $\widetilde{H}_n$. To what level does the symbol energy increase when $\overline{N} = 8$? What is the transmit energy per each tone with $\overline{N} = 8$? (The answer should include ν value that adjusts for prefix-lost input energy.)

b. (3 pts) For Vector DMT on this channel, first as a single-user channel, find the maximum data rate possible in bits/symbol. Can the gap approximation be well used on this design? Comment on bits/tone and bits/real-dimension.

c. (4 pts) For a $U = 2$-user MAC with the energy per user from Part a, find a GDFE solution/realization with the order such that $\pi(2) = 2$ (top of matrix) and $\pi(1) = 1$ (thus the order is as shown). That is, find the 8 unbiased feedback sections and associated (unbiased) feedforward matrices, along with the total data rates for users 2 and 1. Describe heuristically what you might do if the gap were increased from 0 dB to 3 dB. Also find the design for the other vertex with the same sum data rate.

d. (1 pt) Find the loss of using only a linear receiver for this channel in dB.

e. (1 pt) Find γ_{MAC}.

5.17 A GDFE Design Problem 5.16's Channel (8 pts)

This problem continues the previous Problem 5.17. The complex baseband 2×2 channel D-Transform is again

$$H(D) = \begin{bmatrix} 1+D & -.5-.4D \\ .9 & 1-D \end{bmatrix}$$

with white noise (per-tone) variance $\mathcal{N}_0 = .01$ and $\mathcal{E}_x = 2$.

a. (5 pts) Is the rate vector $\boldsymbol{b} = [60\ 50\ ]^*$ feasible if the energy is equally split between the two users? If so, find a design for $\overline{N} = 8$. If not, find a design for $\overline{N} = 8$ with minimum sum energy. The answer need only include the energy and bit distributions for this problem (no need to design the 8 GDFE's, although an implementation would need them).

b. (3 pts) Repeat Part a for $\boldsymbol{b} = [65\ 55\ ]^*$. If infeasible, find the best design with minimum energy sum.

5.18 GDFE BC Design with Duality (20 pts)

This exercise returns to Problem 5.16's MAC

$$H(D) = \begin{bmatrix} 1+D & -.5-.4D \\ .9 & 1-D \end{bmatrix}$$

with white (spatially and in frequency) noise (per-tone) variance $\mathcal{N}_0 = .01$ and $\mathcal{E}_x = 2$ with one energy unit for each MAC user. This noise whitened MAC channel is now the dual of another BC, $\widetilde{H}_{BC}(D)$.

a. (3 pts) Find ν. Find the number of (complex) channel input dimensions (per symbol) on $\widetilde{H}_{BC}$ if $\overline{N} = 8$? How many (complex) input dimensions are there for the $R_{\boldsymbol{XX}}$ used in the tonal channel, so after any transmit linear matrix (A)? How many dimensions total for all users at the input to any A matrix and/or lossless precoder input/output on this channel?

b. (3 pts) Using the MAC's FFT, $\widetilde{H}_n$, find the dual $\widetilde{H}_{BC,n}$. List the $\overline{N}$ values for $\widetilde{H}_{BC,n}$.

c. (6 pts) Find the set of dual-BC autocorrelation matrices $\{R_{\boldsymbol{XX}}(u,n)\}$ that correspond to a white (equal energy on all dimensions) input over all users on the MAC, and the corresponding dual channel's bit distribution, users' bits/symbol, and overall rate sum. (Hint: use the mac2bc.m program).

d. (2 pts) Find the maximum sum rate for this channel. (Hint: use the bcmax program.) Show how this equals the max energy-sum rate sum.

e. (6 pts) Find the lossless precoders (G_n) and linear-matrix (A_n), along with the receivers (unbiased) MSWMF (W_n) for the BC design corresponding to Part c. (hint: use the mu_bc.m program)

5.19 Specific rate-point GDFE BC Design with Duality (20 pts)

For Problem 5.17's channel

$$H(D) = \begin{bmatrix} 1+D & -.5-.4D \\ .9 & 1-D \end{bmatrix}$$

with white noise (per-tone) variance $\frac{\mathcal{N}_0}{2} = .01$ and $\mathcal{E}_x = 2$. The target data BC rate vector for $\overline{N} = 8$ is $\boldsymbol{b} = [50 \;\; 40]^*$.

a. (6 pts) Find a design that for this data rate that minimizes the BC transmit energy. Find a design using minPMAC to set the min-sum energy levels used. Show the A matrix and the precoder G, as well as the (unbiased) MSWMF (W) for each tone.

b. (5 pts) Are there other designs that achieve (at least) the same rate point? For instance, consider a design using admMAC. Provide the design in the same way as Part a. Comment on the design relative to Part a.

c. (2 pts) Find the loss γ_{BC} for this channel.

d. (2 pts) Find the margin for the design in Part a.

5.20 An Interference Channel Comparison (8 pts)

For the (memoryless) IC

$$H(D) = \begin{bmatrix} 50 & 50 \\ 0 & 1 \end{bmatrix}$$

with white (spatially and in frequency) noise (per-tone) variance $\frac{\mathcal{N}_0}{2} = 1$ and $\mathcal{E}_x = 2$:

a. (4 pts) Find an iterative (simultaneous) water-filling solution with no GDFE's and each user as crosstalk to the other.

b. (4 pts) Find the best rate sum when both receivers may use a GDFE for successive decoding and compare to Part a.

Bibliography

[1] John M. Cioffi and G. David Forney, Jr. *"Generalized Decision-Feedback Equalization for Packet Transmission with ISI and Gaussian Noise"*. Communications, Computation, Control, and Signal Processing, a tribute to Thomas Kailath. Springer, Boston, MA ISBN 978-1-4615-6281-8. Editors: Paulraj A., Roychowdhury V., Schaper C.D. pp. 79-127.

[2] Wei Yu and R. Lui. *"Dual methods for nonconvex spectrum optimization of multicarrier systems"*. IEEE Transactions on Communications Vol: 54, Issue: 7, July 2006. pp 1310-1322.

[3] Raphael Centrillon, Wei Yu, M. Moonen, J. Verlinden, and T. Bostoen. *" Optimal multiuser spectrum balancing for digital subscriber lines"*. IEEE Transactions on Communications Vol: 54, Issue: 5, May 2006. pp 922-933.

[4] G. Ginis, W. Yu, C. Zeng, and J.M. Cioffi "Dynamic Digital Communication System Control" US patent 7,158,563 family (multiple patents issued) Filed June 8, 2001, Issued (first) 1/2/2007. See also "Vectored Transmission for Digital Subscriber Lines" IEEE Journal on Selected Areas in Communications Vol. 20 No. 5, June 2002, pp. 1085-1104.

[5] Thomas Starr, John M. Cioffi, and Peter. J. Silverman *"Understanding Digital Subscriber Line Technology"*. Prentice Hall, New Jersey, 1999.

[6] Zhi-Quan Luo and Jong-Shi Pang "Analysis of Iterative Waterfilling Algorithm for Multiuser Power Control in Digital Subscriber Lines". EURASIP Journal on Applied Signal Processing. DOI 10.1155/ASP/2006/24012 December 2006, pp. 1-10.

Index