STANFORD

Lecture 9

Finish MAC & Broadcast Channels

April 30, 2024

JOHN M. CIOFFI

Hitachi Professor Emeritus of Engineering

Instructor EE379B – Spring 2024

Announcements & Agenda

- Announcements
  - Problem Set #4 due today
  - Midterm in class Thursday

- Agenda (L9)
  - Finish L8 MAC examples
    - mu_mac.m
  - Simultaneous Water-Filling for MAC max rate sum
    - SWF.m and macmax.m
  - **MAC:** Capacity region for frequency-indexed MACs
  - **BC:** Precoder Basics for the Matrix AWGN
  - Scalar Gaussian BC

MAC Examples

Sections 2.7.3-4

Matrix AWGN MAC Example 1 ($L_{x,u} \equiv 1$)

$$\tilde{H} = \begin{bmatrix} 5 & 2 \\ 3 & 1 \end{bmatrix}$$

```
>> Rf=H'*H =
   34    13
   13     5
>> Rbinv=Rf+eye(2) =
   35    13
   13     6

>> Gbar=chol(Rbinv)   =
   5.9161    2.1974
        0    1.0823

>> S0=diag(diag(Gbar))*diag(diag(Gbar)) =
  35.0000         0
        0    1.1714
>> G = inv(diag(diag(Gbar)))*Gbar =
   1.0000    0.3714
        0    1.0000
>> >> b=0.5*log2(diag(S0))  =
   2.5646
   0.1141
>> sum(b) =    2.6788
```

REVERSE ORDER – same commands –other vertex

```
>> H=[   2    5
    1    3] ;
Rbinv =
    6    13
   13    35
Gbar =
 2.4495    5.3072
      0    2.6141
S0 =
   6.0000        0
        0   6.8333
G =
 1.0000    2.1667
      0    1.0000
b =
 1.2925
 1.3863

sum(b) =   2.6788
```

- These are the two vertices for dimension-share (pentagon outer face).
- Two receiver output dimensions for each one-dimensional input $\boldsymbol{x}_u$ (instead of 1 output dimension earlier)

Example 1 continued

- Receiver filters and bias are

Vertex 1

```
>> W=inv(S0)*inv(G') =
    0.0286         0
   -0.3171    0.8537
>> Wunb=S0*inv(S0-eye(2))*W =
    0.0294         0
   -2.1667    5.8333

>> MSWMFu=Wunb*H' =
    0.1471    0.0882
    0.8333   -0.6667
>> Gunb=eye(2)+S0*inv(S0-eye(2))*(G-eye(2)) =
    1.0000    0.3824
         0    1.0000

>> MSWMFu*H =
    1.0000    0.3824
    2.1667    1.0000
```

Vertex 2

```
>> W=inv(S0)*inv(G') =
    0.1667         0
   -0.3171    0.1463
>> Wunb=S0*inv(S0-eye(2))*W =
    0.2000         0
   -0.3714    0.1714
>> MSWMFu=Wunb*H'  =
    0.4000    0.2000
    0.1143    0.1429
>> Gunb=eye(2)+S0*inv(S0-eye(2))*(G-eye(2)) =
    1.0000    2.6000
         0    1.0000
>> MSWMFu*H=
    1.0000    2.6000
    0.3714    1.0000
```

- Not really triangular, why?

Easier with mu_mac.m

```
function [b, GU, WU, S0, MSWMFU] = mu_mac(H, A, Lxu, cb)
```

channel Rxx1/2 #/user xmit antennas 1 cplx, 2 real

```
Per-tonal (temporal dimension) multiuser mac receiver and per-user
bits

  Inputs: H, A , Uind , cb
  Outputs: b, GU, WU, S0, MSWMFU

  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 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
  S0: 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
```

```
H=[5 2 ; 3 1];
[b, GU, WU, S0, MSWMFU] = mu_mac(H, eye(2), [1 1] , 2);

b =   2.5646   0.1141

GU =
   1.0000   0.3824
        0   1.0000

WU =
   0.0294        0
  -2.1667   5.8333
S0 =
  35.0000        0
        0   1.1714

MSWMFU =
   0.1471   0.0882
   0.8333  -0.6667
>> MSWMFU*H =
   1.0000   0.3824
   2.1667   1.0000

>> SNR = 10*log10(diag(S0)) =
  15.4407
   0.6872

>> sum(b) =  2.6788
```

Example 2: 2 x 3 MAC (secondary users)

```
H=[5 2 1
3 1 1];    basically added a 3rd user
[b, GU, WU, S0, MSWMFU] = mu_mac(H, eye(3), [1 1 1] , 2)

b =   2.5646     0.1141    0.1137
GU =    1.0000    0.3824    0.2353
              0    1.0000    0.1667
              0         0    1.0000
WU =
   0.0294       0       0
  -2.1667    5.8333       0
  -1.2857   -0.1429    5.8571
S0 =
  35.0000       0       0
        0    1.1714       0
        0         0    1.1707
MSWMFU =
   0.1471    0.0882
   0.8333   -0.6667
  -0.8571    1.8571
>> sum(b) =    2.7925
>> MSWMFU*H=
   1.0000    0.3824    0.2353
   2.1667    1.0000    0.1667
   1.2857    0.1429    1.0000
>>  SNR10*log10(diag(S0))=
  15.4407
   0.6872
   0.6846
```

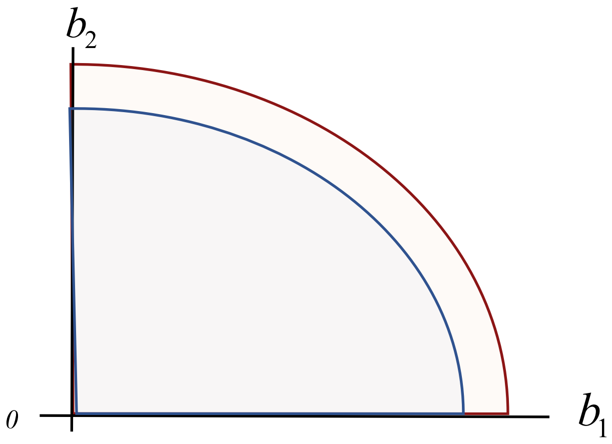
- The channel rank is 2 so at least 1 secondary comp = 3-2.
- But secondary applies to energy-sum MAC (which this is not, yet).
- If original 2 units of energy is spread over 3 users?

```
>> [b, GU, WU, S0, MSWMFU] = mu_mac(H, (2/3)*eye(3), [1 1 1] , 2)

b =   2.0050    0.1009    0.0696
GU =
   1.0000    0.3824    0.2353
        0    1.0000    0.3878
        0         0    1.0000
WU =
   0.0662         0         0
  -2.3878    6.6582         0
  -2.0000   -0.5000    9.8750
S0 =
  16.1111         0         0
        0    1.1502         0
        0         0    1.1013
MSWMFU =
   0.2206    0.1324
   0.9184   -0.3367
  -0.7500    2.2500
>> sum(b) =  2.1755  (lower than 2x2 value of 2.6788)
```

- Relatively more energy on secondary-user comp(s), bsum ↓.

Non-Zero Gap Achievable Region

- Construct $\mathcal{C}(\boldsymbol{b})$ with $\Gamma = 0$ dB.
- Reduce all rates by γ_b relative to boundary points.
- Inscribe smaller region $\mathcal{C}(\boldsymbol{b})$- $(\gamma_b \odot \mathbf{1})$.
- Square constellations instead of spheres (AWGN) loss 1.53 dB in gap above (0.25 bit/dimension).



Simultaneous Water-Filling for MAC max rate sum

Sections 2.7.3-4

Revisit the rate-sum mutual information

$$b = \sum_{u=1}^{U} \tilde{b}_u \le \mathcal{I}\left(\boldsymbol{x};\boldsymbol{y}\right) = \log_2 \frac{\left|H \cdot R_{\boldsymbol{xx}} \cdot H^* + R_{\boldsymbol{nn}}\right|}{\left|R_{\boldsymbol{nn}}\right|}$$

- Maximum rate-sum focuses on the numerator,
  - when optimizing over $R_{\boldsymbol{xx}}$.

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

- Have we seen this problem before?
  - Yes, it is Vector Coding / Waterfilling , except with $\widetilde{H}_u \to R_{noise}^{-1/2} \cdot H_u$ for each u

Simultaneous Waterfilling

$$\mathcal{E}_{u,l} + \frac{1}{g_{u,l}} = K_u \ \forall\ u = 1, \dots, U'$$

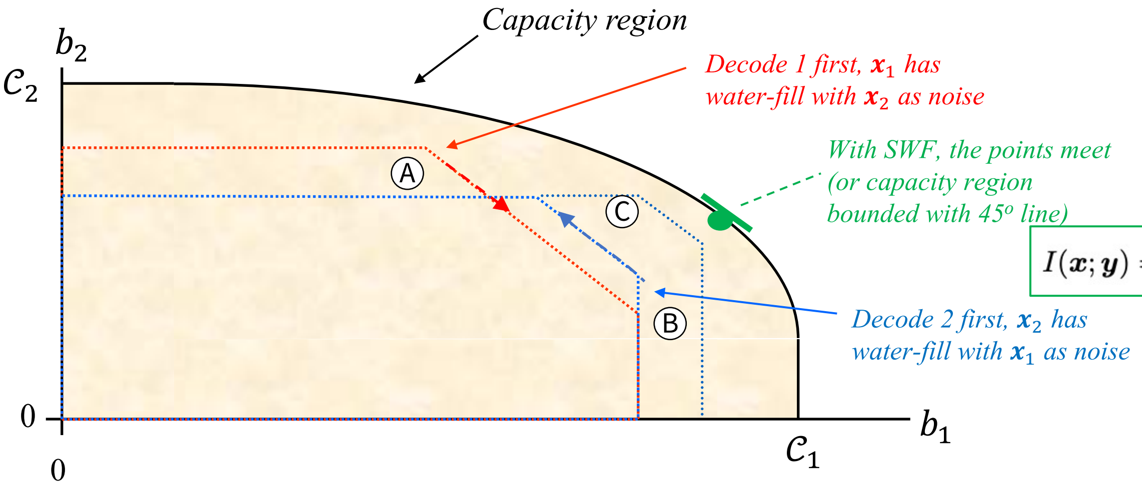
$$\sum_{l=1}^{L_x} \mathcal{E}_{u,l} = \mathcal{E}_u$$

$$\mathcal{E}_{u,l} \ge 0$$

$$\boldsymbol{x}_u = M_u \cdot \boldsymbol{v}_u$$

- But now it repeats U times in the same form for each user.
  - Optimum has each user **simultaneously water-fill** by treating all other users (water-fill) spectra as noise.

Compute Using Iterative Water-filling



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

- SWC problem is convex, and each single-water-fill step is "gradient-like" in improving direction, swf.m
- E-Sum SWC is a saddle point with enlarged region.
  - 2nd optimization is on the allocation of $\mathcal{E}_{x,u} \rightarrow \sum_{u=1}^{U} \mathcal{E}_{x,u} = \mathcal{E}_x$.

SWF.m Program for MAC's max sum rate

```
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
```

- Eu is each user's energy**/sample.**
- For now, $N = 1$, so time/freq are same:
  - H=h .
- Lxu is number of antennas for each user.
- Separate specification of Rnn removes need for noise whitening.
- cb=1 for complex, =2 for real.

Revisit Previous example (slides L8: 26-29)

```
H =
   5    2    1
   3    1    1
>> [Rxx, bsum , bsum_lin] = SWF([1 1 1], H, [1 1 1], eye(2),  2)
Rxx =
   1    0    0
   0    1    0
   0    0    1
bsum =    2.7925
bsum_lin =    1.4349
```

- Same result as L8:29, so each user waterfills with all others as noise; this is trivial when each user has only 1 input dimension. (Why?)
- This is for input energy-vector constraint.
- Note linear solution (no feedback, so matrix MMSE-LE) loses roughly ½ the data rate.
- SWF becomes more interesting when $N > 1$ tones or if $L_{x,u} > 1$ antennas.

For $L_{x,u} = 2$; $u = 1,2$?

```
>> H2=[4 3 2 1
5 6 7 8];
>> [Rxx, bsum , bsum_lin] = SWF([0.5 0.5], H2, [2 2 ], eye(2),  2)
Rxx =
   0.7121   0.4528        0        0
   0.4528   0.2879        0        0
        0        0   0.2876   0.4527
        0        0   0.4527   0.7124
bsum =    5.3434
bsum_lin =   4.0920
>> trace(Rxx) % =    2  (check)
>> trace(Rxx(1:2,1:2))% =    1
>> trace(Rxx(3:4,3:4))  % =    1
```

Energy input is per trace{Rxx,u} per sample!

- Note block-diagonal Rxx.
- Linear-only loses about 25% in data rate (for this channel).

Or use Macmax.m for Esum MAC

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 sum trace of the corresponding users' Rxx (u).
The sum energy is computed 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

- ENERGY-SUM input (per sample)
  - Lxu = numbers of xmit antennas/user
- Time-domain (noise-whitened) h
- This is actually a double loop that:
  - water-fills each and every user for some current set of per-user energies and
  - adjusts energies so they sum to total but increase the rate sum.
- It corresponds to a saddle point.
  - It is not convex (although each sub loop is).
  - It has a solution and converges anyway.
- This will be easier understood later as a dual of a broadcast problem as to why this is true.

Back to Example

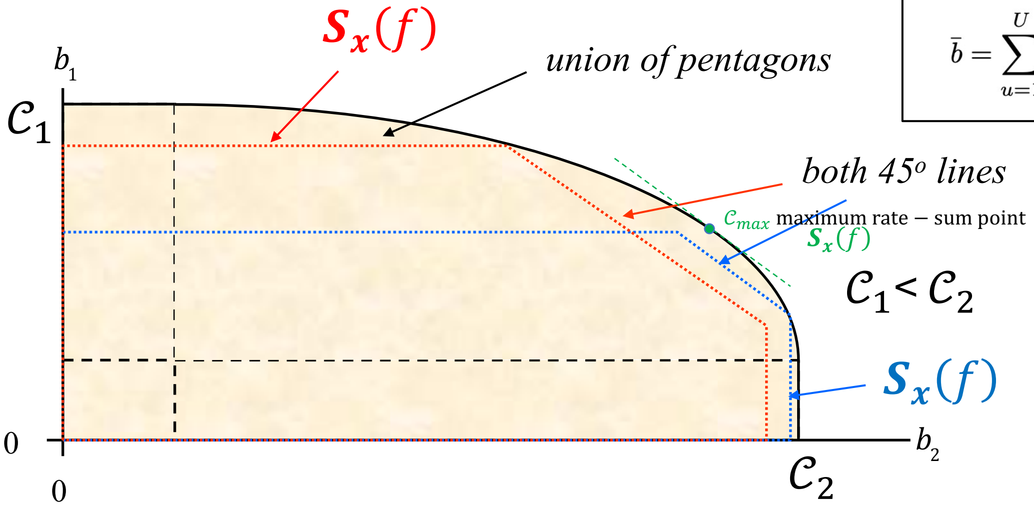
```
>> H3(:,:,1)=H

H =
   5   2   1
   3   1   1
>> [Rxx, bmacmax, bmaclin]=macmax(3/2, H, [1 1 1], 1, 2)
Rxx =
   3.0000        0        0
        0   0.0000        0
        0        0   0.0000
bmacmax =    3.3432
bmaclin =    3.3432
```

- This produces a larger data rate because there is less energy restriction.
- Rxx energizes just user 3! (It's all primary user component, and users 1 and 2 are secondary)
- Linear is the same. Why?

Capacity region for frequency-indexed MACs

Sections 2.7.4.1-2

$\mathcal{C}(b)$ is union of $S_x(f)$-indexed Pentagons



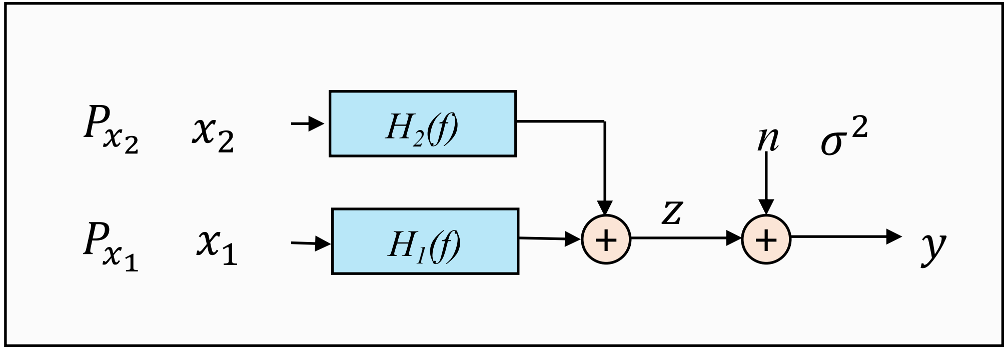
$$\bar{b} = \sum_{u=1}^{U} \bar{b}_u \leq \bar{\mathcal{I}}(\boldsymbol{x};\boldsymbol{y}) = \int_{-\infty}^{\infty} \frac{1}{2} \cdot \log_2 \left[1 + \frac{\sum_{u=1}^{U} S_{x,u}(f) \cdot |H_u(f)|^2}{S_n(f)} \right] df$$

$$S_{x,u}(f) + \frac{\sigma^2 + \sum_{i \neq u} S_{x,i}(f) + |H_i(f)|^2}{|H_u(f)|^2} = K_u$$

Simultaneous water-filling
→ Maximum rate sum

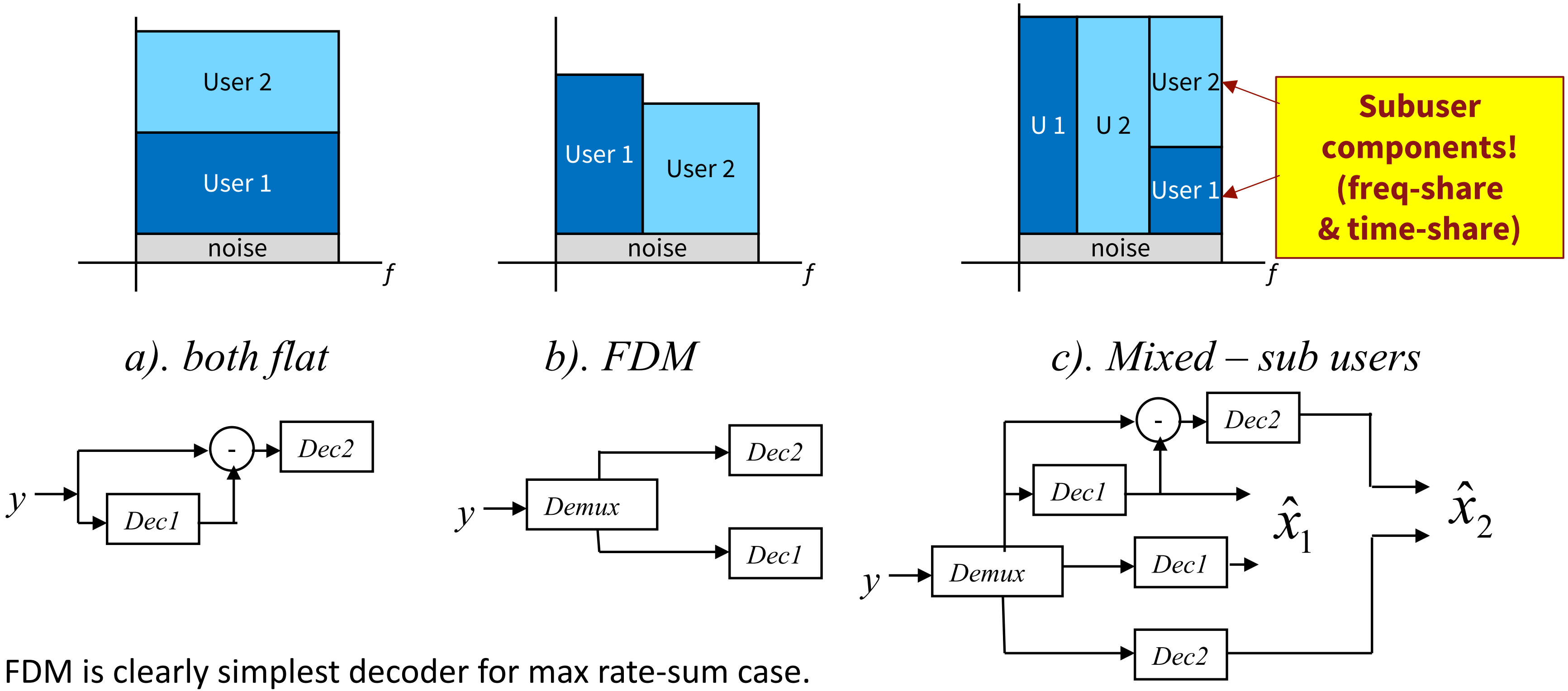
- Each pentagon corresponds to an $\boldsymbol{S_x}(f)$ choice.
  - The pentagons become triangles for the sum-energy MAC.
- The union (convex hull is union when inputs are Gaussian) can dimension-share in frequency as $N \to \infty$.

MT MAC



- The users have continuous-time/frequency channels → use MT on each, theoretically.
- This really means dimensionality is infinite (or very large) so "dimension-sharing" may be inherent.
- SWF applies, but with some interpretation (like power instead of energy and power per dimension instead of power-spectral density, etc.).

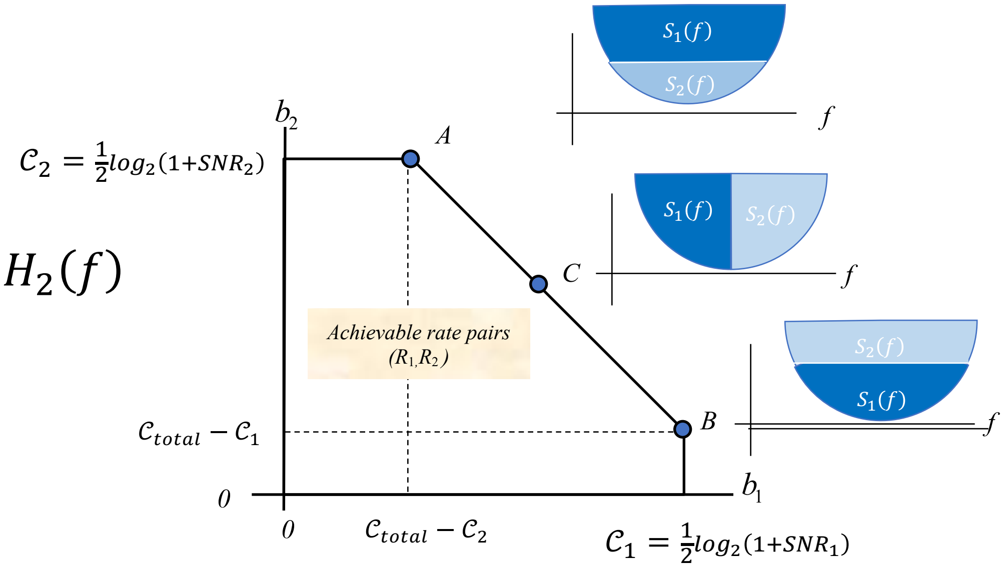
Decoders and SWF



- FDM is clearly simplest decoder for max rate-sum case.
- Both users (and all components in case c) are primary.

Symmetric 2-user channel and SWF

$$H_1(f) = H_2(f)$$

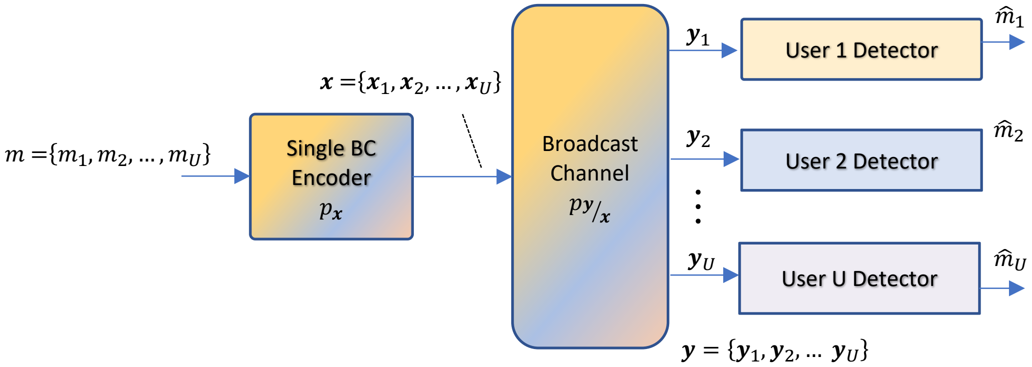


- Symmetric means $H_1(f) = H_2(f)$ (noise is one-dimensional and added to sum)
- Each of points A, B, and C have different SWF spectra – all have same (max) rate sum

Basic Precoders and the Matrix AWGN

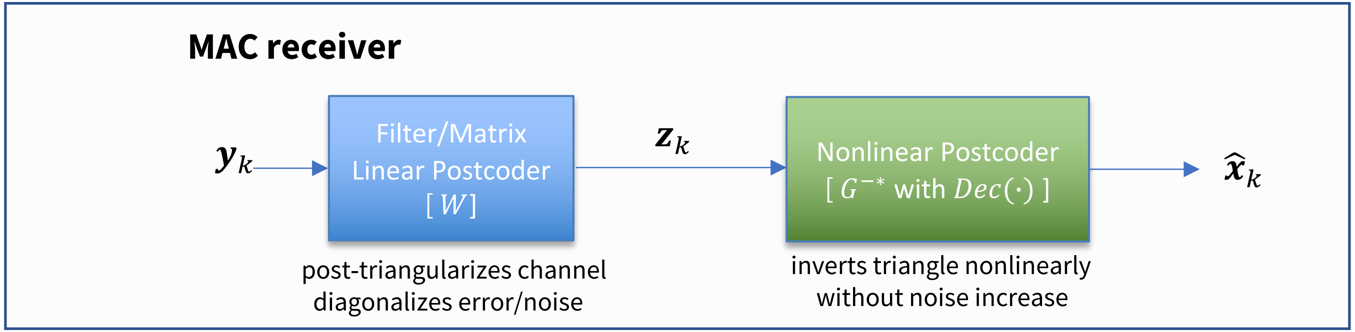
PS5.1 - 2.28 modulo precoding function

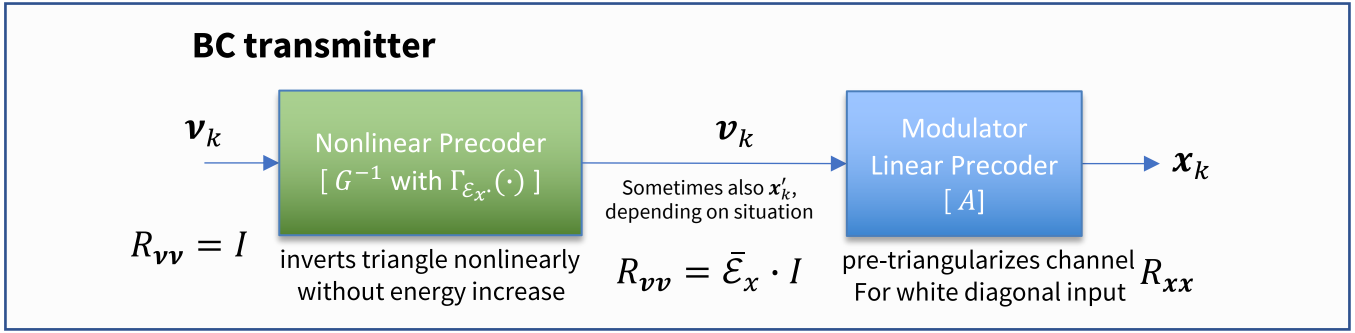
Broadcast Channel (BC)



- The BC is the "Dual" of the MAC.
- Receivers are in different places and so cannot "co-process" $\{\boldsymbol{y}_u\}$.
- Transmitter can co-encode/generate $\boldsymbol{x}$, although input messages remain independent.
  - Who encodes first? (may be at disadvantage)
  - Who encodes last? (knowing other users' signals is an advantage)
  - What then is the **order**?

BC is "reversed" MAC





- The MAC's uncoordinated user input is a kind of "worst case" transmitter, reducing data rate.
  - With only an energy-sum constraint, these worst-case inputs' users best pass as primary user components; secondary components "freeload" on the primary's passage.
- The BC similarly will effectively correspond to a worst-case noise for which receiver coordination is useless, reducing data rate.
  - With worst-case noise, the channel best passes the primary components'; secondary components freeload on the primary's passage.

Triangular Matrices - Innovations and Prediction

- Prediction for some **user order** separates a modulated input to independent message components.

$$\boxed{\boldsymbol{\nu}_u = \boldsymbol{x}_u - \hat{\boldsymbol{x}}_{u/\{\boldsymbol{x}_{u+1} \ldots \boldsymbol{x}_U\}}}$$

Innovations or predictions, but for BC $\boldsymbol{\nu}_u$ become the independent-users' subsymbols, with normalization $R_{\boldsymbol{\nu}\boldsymbol{\nu}}(u) = I$.

- This is a triangular relationship (**inverse of upper triangular is also upper triangular**).

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \\ \vdots \\ \boldsymbol{\nu}_U \end{bmatrix} = \begin{bmatrix} 1 & g_{1,2} & \cdots & g_{1,U} \\ 0 & 1 & \cdots & g_{2,U} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_U \end{bmatrix} = G^{-1} \cdot \boldsymbol{x}$$

- OR, $\boldsymbol{x} = G \cdot \boldsymbol{\nu}$ (G is also upper triangular).
- *Generating $\boldsymbol{x}$ from $\boldsymbol{\nu}$ can increase energy (~ enhance noise in MAC rcvr) if implemented directly (linearly).*

(order reversal is intentional)

Voronoi Regions and Modulo Addition (Sec 2.1)

- A lattice is a (countable) group of vectors $\boldsymbol{\Lambda} = \{\boldsymbol{x}\}$ that is closed under an operation addition, so that
  - If $\boldsymbol{x}_1 \in \boldsymbol{\Lambda}$ and $\boldsymbol{x}_2 \in \boldsymbol{\Lambda}$, then $\boldsymbol{x}_1 + \boldsymbol{x}_2 \in \boldsymbol{\Lambda}$. (Section 2.2.1.1 and Appendix B.2)
  - A constellation is a finite subset of a lattice, plus a constant (coset) $C \subset \boldsymbol{\Lambda} + \boldsymbol{\lambda_0}$. ($\boldsymbol{\lambda_0}$ ensures average value is zero.)

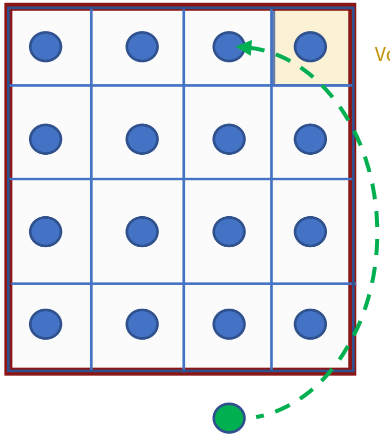
SQ rectangular or $\mathbb{Z}^2$

HEX or A2

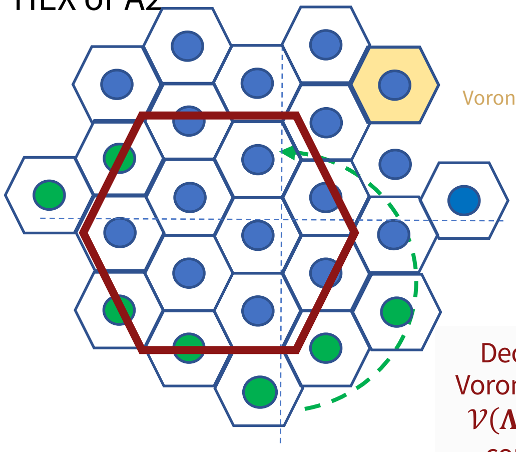
$\boldsymbol{\lambda_0}$ is the "coset leader"

Decision or Voronoi Region $\mathcal{V}(\boldsymbol{\Lambda}' = 4\mathbb{Z}^2)$

Contains 16



Decision or Voronoi Region $\mathcal{V}(\boldsymbol{\Lambda} = \mathbb{Z}^2)$



Decision or Voronoi Region $\mathcal{V} = A_2$

Green indicates Modulo operation (split ties equally)

Decision or Voronoi Region $\mathcal{V}(\boldsymbol{\Lambda}' = \boldsymbol{3A_2})$ contains 9

- Voronoi Region of a lattice, $\mathcal{V}(\Lambda_c)$ is the decision region around any point with volume $V(\Lambda_c)$.
  - Λ_c is the "coding" lattice; codes try to pack more points into limited space (volume/area). – HEX is better than SQ.
- A constellation C typically selects points in one (coding-gain) lattice, Λ_c, within the $\mathcal{V}(\Lambda_s)$ of another (shaping-gain) lattice $\boldsymbol{\Lambda}'$ that is larger (can be scaled versions of one another or possibly different). (Subtract any nonzero vector mean to save energy.)
  - All points in Λ_c outside of $\mathcal{V}(\Lambda_s)$ map into a point inside $\mathcal{V}(\Lambda_s)$ - disguised detector problem.

More general precoder (than Tomlinson/Laroia …)

- Generalize **Modulo Operation:** $\boxed{(\boldsymbol{\nu})_{\Lambda_s} = \boldsymbol{e} \ni \min_{\boldsymbol{\lambda}\in\Lambda_s} \|\boldsymbol{e}\|^2 \;\text{ where }\; \boldsymbol{e} = \boldsymbol{\nu} - \boldsymbol{\lambda}}$

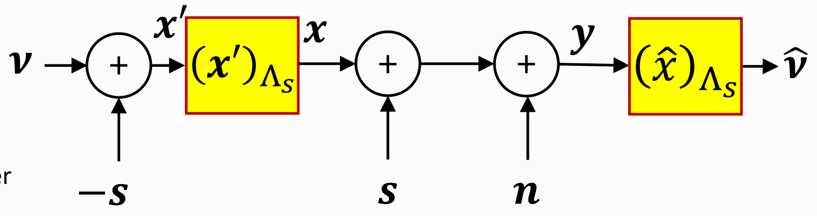
- $\boldsymbol{e}$ does not necessarily need to be a point in Λ_c ; instead, it is a point in $\mathcal{V}(\Lambda_s)$.
  - It's essentially the error between input and output of decoder with decision region equal to $\mathcal{V}(\Lambda_s)$ - sbs in trivial cases of "uncoded" $\mathcal{V}(\Lambda_s)$=$\mathbb{Z}^2$.

- useful

Lemma 2.8.1 (distribution of modulo addition) *Modulo addition distributes as*

$$(\boldsymbol{\mu}+\boldsymbol{\nu})_\Lambda = (\boldsymbol{\mu})_\Lambda \oplus_\Lambda (\boldsymbol{\mu})_\Lambda \quad . \tag{2.371}$$

- Side info is $\boldsymbol{s}$.
  - $\boldsymbol{s}$ *is known (ISI for causal G(D) = in Tomlinson/DFE)*
  - Pre-subtract (precode) and use modulo to set $\mathcal{E}_x$ level (no energy increase for - $\boldsymbol{s}$ **).**
  - Any Λ_s shaping gain also applies here (8.5,$\boldsymbol{s}$=0).
    - $\boldsymbol{\nu}$ then is any (e.g., SQ) input constellation.
    - In shaping case, $\hat{\boldsymbol{\nu}}$ also needs a subsequent detector yet for whatever code is used on $\boldsymbol{\nu}$.
  - $\boldsymbol{x}$ will effectively have continuous uniform distributions over Λ_s.



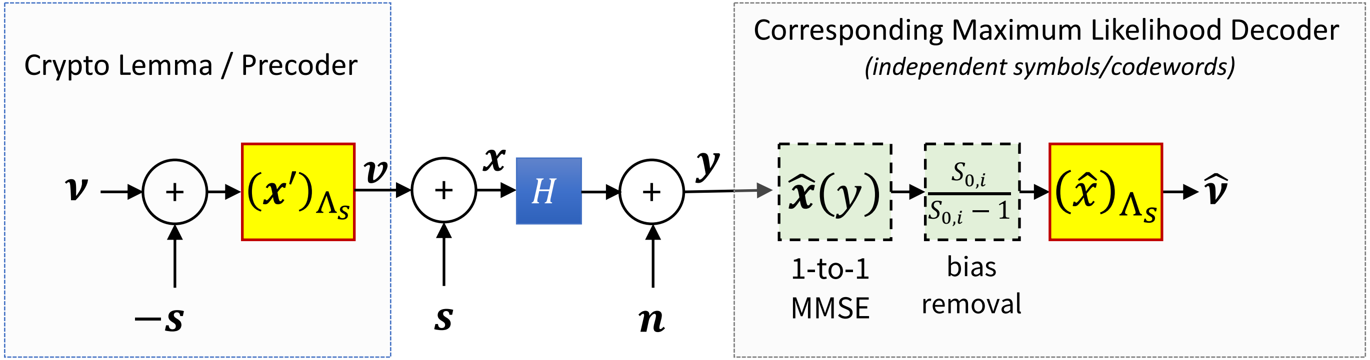
Dirty paper
"noiseless"

(nonlinear)
precoder
…

For BC: $\boldsymbol{s}$ will be the earlier users (but their xtalk cancels) ~ **order.**

With nontrivial channel, need MMSE version

Forney's Crypto Lemma – 2003 (Section 2.8.1.2)



No xmit energy increase
Simplifies ML detection

- The MMSE part can be important in non-trivial cases (often missed in most info-theory texts).
  - It's reshaping the channel crosstalk and/or ISI in MMSE (not zero-forcing) sense.
- When $\boldsymbol{s}$ is uniform over $\mathcal{V}(\Lambda_s)$, then so is $\boldsymbol{v}$, **AND** $\boldsymbol{v}$ is independent of both $\boldsymbol{s}$ and $\boldsymbol{v}$ (like encryption), $\boldsymbol{s}$ is the "key"
  - Or "writing on dirty paper" ($\boldsymbol{s}$ is the dirt, $\boldsymbol{v}$ is the writing, and the second modulo cleans the paper).
- Sometimes the channel adds $\boldsymbol{s}$ (ISI/xtalk), sometimes the transmitter adds $\boldsymbol{s}$ (xmit case, $\boldsymbol{s}$ shares dimensions and energy with $\boldsymbol{x}$).
  - The add-at-xmit case has $\boldsymbol{s}$ as other users, effectively (with a twist .. later).

Non-Causal ?

- Subtly, the lattice Λ_s has a dimensionality N over which $\boldsymbol{s}$ and $\boldsymbol{x}$ are uniformly distributed.

- Wise dimension use with fixed energy $\mathcal{E}_x$ suggests Λ_s has a hyper-spherical boundary, as $N \to \infty$.
  - This infinite-length precoder then also obtains full 1.53 shaping gain.

- Asymtotically, the modulo has infinite number of dimensions, so requires infinite delay for $\boldsymbol{s}$ to be fully known in the formation of $\boldsymbol{x}$; whence "non-causal."
  - Approximated with finite delay in practice, $\boldsymbol{s}$ becomes another user's encoded signal known first (~ non-causal) → **order .**

$$\boxed{(\cdot)_{\Lambda_s}} \rightarrow \boxed{(\cdot)_{\mathcal{E}_x}}$$

- Mod holds energy at $\mathcal{E}_x$ (Gaussian in any finite number of dimensions, uniform in infinite dimensional hypersphere).

- If Λ_s is hypercube, Forney's crypto still holds but with SNR loss of (up to) 1.53 dB (the maximum shaping gain).
  - So reuse code with $\Gamma \to 0$ dB, with QAM constellations and the (up to) 1.53 dB loss remains (greatly simplifies precoder implementation),
    - but everything else works the same.

Midterm Review

- Questions
- Advice

STANFORD

End Lecture 9