STANFORD

Lecture 10

Broadcast Channels Continued

May 8, 2023

JOHN M. CIOFFI

Hitachi Professor Emeritus of Engineering

Instructor EE392AA – Spring 2023

Announcements & Agenda

- Announcements
  - Problem Set #5 due Wednesday May 17
  - Midterms grades (and PS4?) at canvas
    - Feedback/assessment from exam
  - Section 2.8 continues
  - Projects List (slide 3)

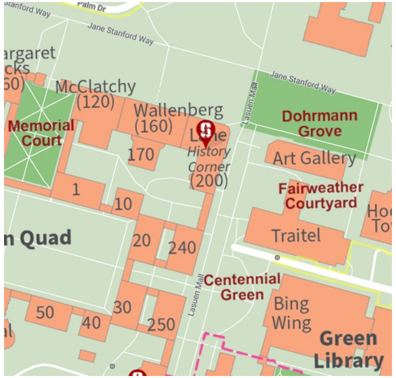
- Problem Set 5 = PS5 (due **May 17**)
  1. 2.28 modulo precoding function
  2. 2.29 scalar BC region
  3. 2.30 vector BC design
  4. 2.31 2-user IC region
  5. 2.32 bonded relay channel

- Agenda
  - Vector Gaussian BC Design
  - Worst-case-Noise BC Design
  - Vector WCN-BC Design
  - Maximum BC rate sum

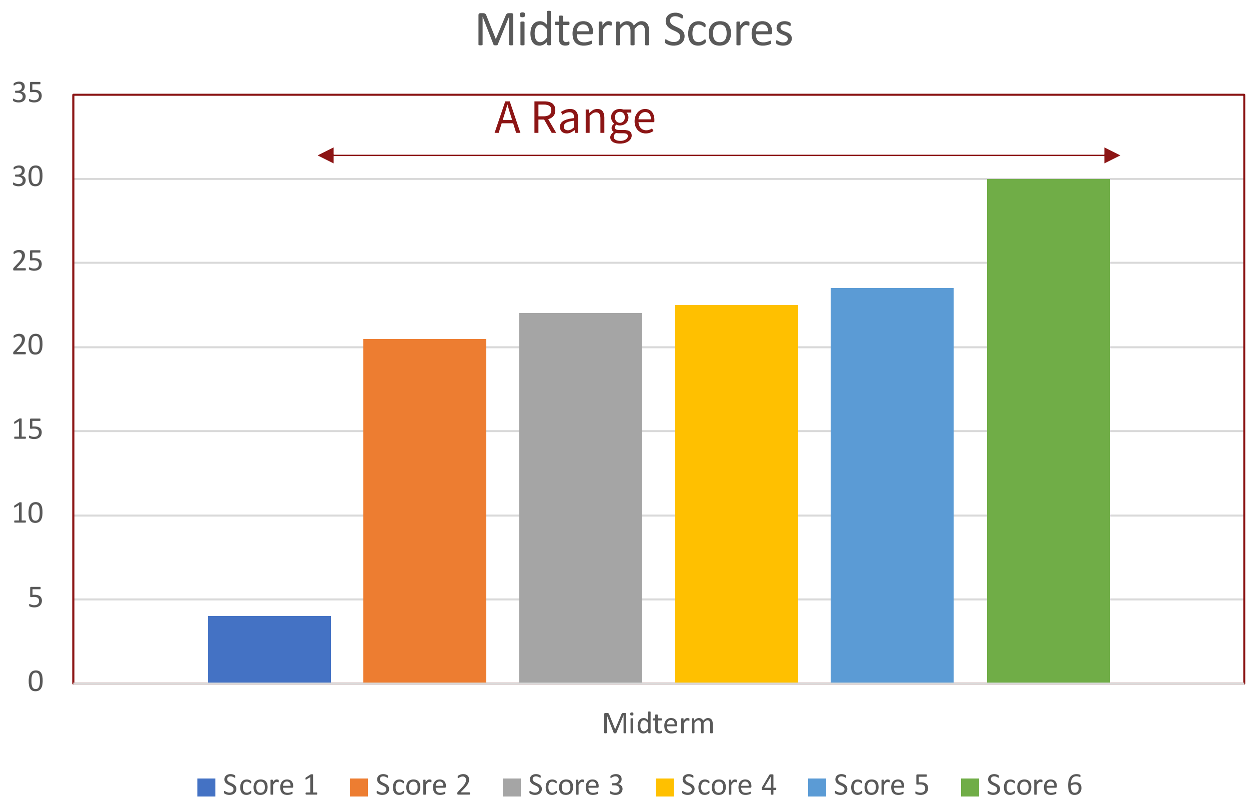
NO CLASS – MONDAY May 15

MAKE-UP -- FRIDAY May 19

3:00 **200-003**



Midterm



[Midterm Link]

[Solutions Link]

Projects – Very Open to Student Proposals too!

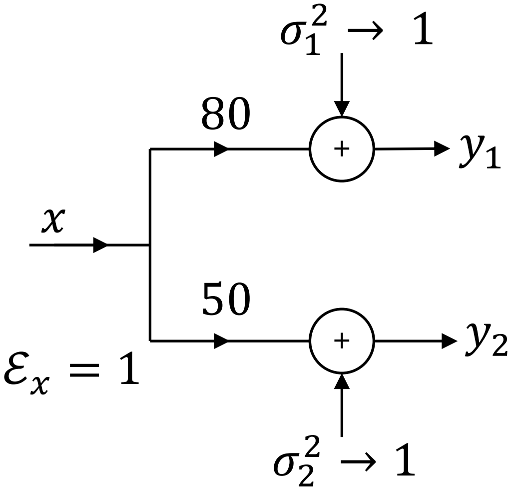
- Adaptive statistical-loading (maybe matlab program?)
- Wireless averaging
  - Single-user
  - MAC
  - BC
- Iterative water-filling program for interference channel (MA is better)
- Multi-level IW program
- Mapping/scheduling of queue-depths to best point (with margin) in capacity region
  - Probably use QPS (queue-proportional scheduling at end of Section 2.6)
- Intelligent Reflective Surface (IRS, or RIS)
  - Alternating optimization routine to optimize the reflecting elements (modelled as unitary matrix)
- minPIC.m program
- Others?

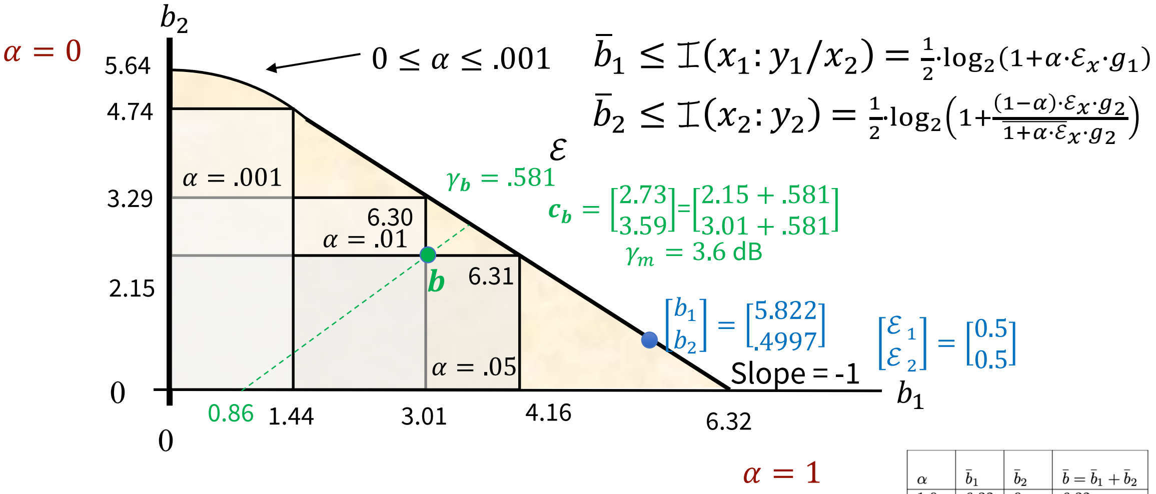
Vector MMSE BC Design

Known $R_{xx}(u)$ Section 2.8.3.1

Revisit Scalar Example – see L9:28

- $h_1 = 0.8\,; h_2 = 0.5; \sigma_1^2 = \sigma_2^2 = .0001$

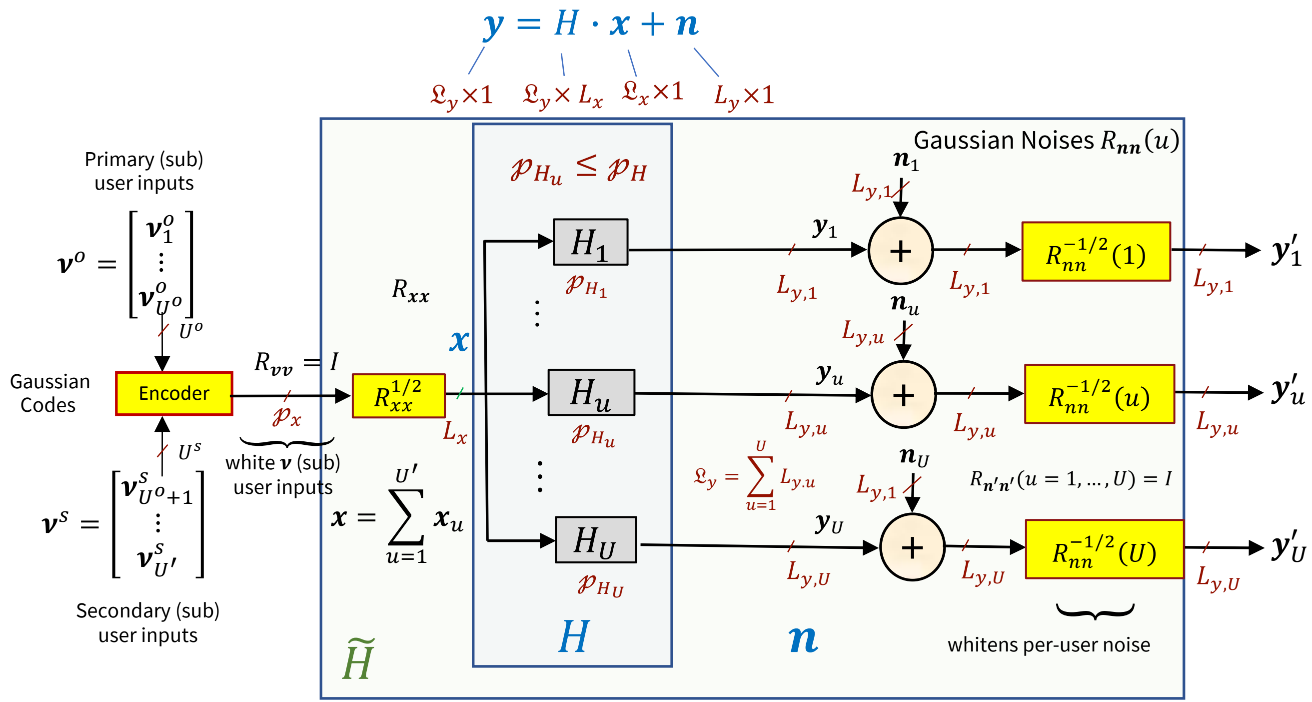




α	$\bar{b}_1$	$\bar{b}_2$	$\bar{b} = \bar{b}_1 + \bar{b}_2$
1.0	6.32	0	6.32
.75	6.12	.20	6.32
.50	5.82	.50	6.32
.25	5.32	1.0	6.32
.10	4.66	1.66	6.32
.05	4.16	2.15	6.31
.01	3.01	3.29	6.30
.001	1,44	4.74	6.18
0	0	5.64	5.64

- User 1 has highest sum rate when User 2 has zero energy
  - User 1 is a primary user/component (precoder subtracts user 2, mods sum to unit energy)
    - And adds user 2 at channel input also (modulo in receiver 1)
  - User 2 is a secondary user/component (decode with user 1 as noise)

Vector Gaussian BC



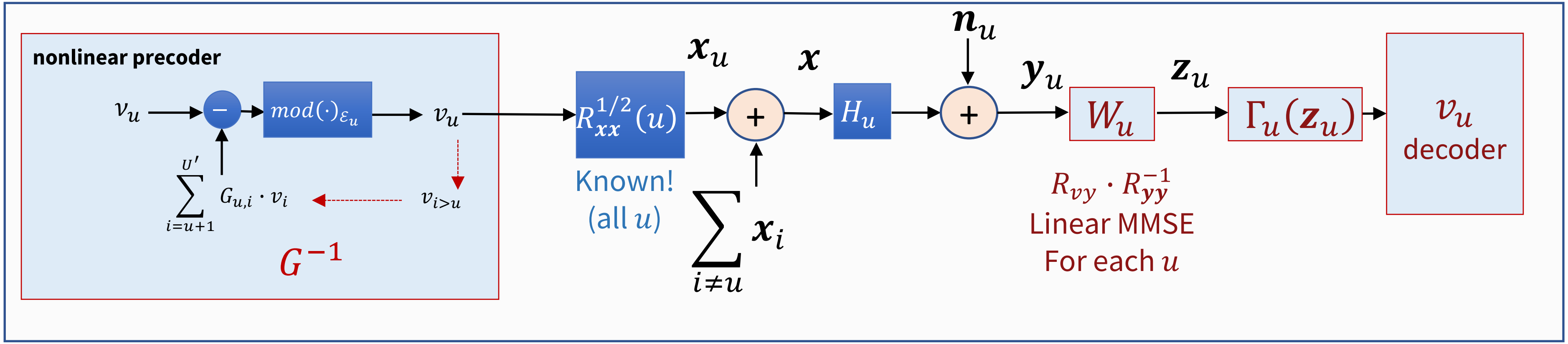
- The users' independent message subsymbol vectors sum to a single BC input $\boldsymbol{x}$

$$\text{\# of subusers} = U' \leq \sum_{u=1}^{U} \min\left(\mathcal{P}_x, \mathcal{P}_{H_u}\right) \leq \mathfrak{L}_y \cdot L_x$$
$$\leq U \cdot L_x \text{ (our designs)}$$

Modulator $A = R_{xx}^{1/2}$ need not be square because it includes the sum

MMSE – BC and Mutual Information – user u

- $\mathcal{I}(\boldsymbol{x}_u; \boldsymbol{y}_u / \boldsymbol{x}_{u+1,\ldots,U}) = \frac{1}{2} \cdot \log_2 \frac{|R_{\boldsymbol{xx}}(\boldsymbol{u})|}{|R_{\boldsymbol{ee}}(\boldsymbol{u})|}$ corresponds to a MMSE problem (like MAC, except $\boldsymbol{y}_u$).

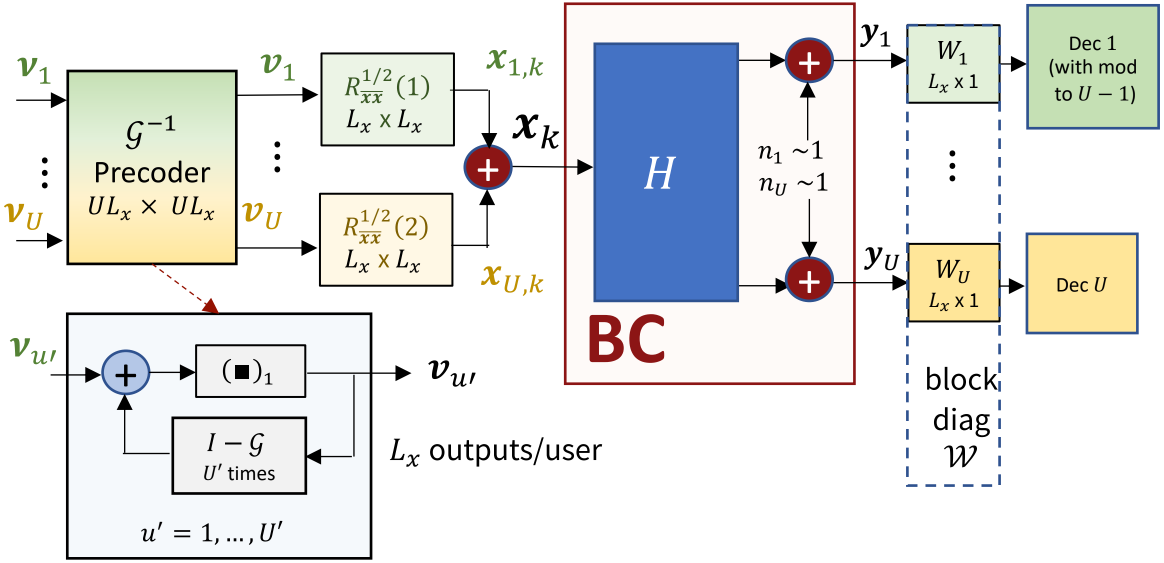


- There is successive-decoding ("GDFE") canonical performance (up to U' components)
  - BC implements the G^{-1} with a lossless precoder at the transmitter

- This structure reliably achieves highest rate for given input $R_{\boldsymbol{xx}}(u)$, and order $\boldsymbol{\pi}_u$.

- The catch? Designer must know $\{R_{\boldsymbol{xx}}(\boldsymbol{u})\}$ and order beforehand.

Structure for all user components $u \in U'$



- This structure needs a little more interpretation when channel rank < number of energized users.

The program mu_bc.m

```
function [Bu, GU, S0, MSWMFunb , B] = mu_bc(H, AU, Lyu , cb)
 --------------------------------------------------------------------------
 Inputs: Hu, AU , Usize, cb
 Outputs: Bu, Gunb, Wunb, S0, MSWMFunb

 H: noise-whitened BC matrix [H1 ; ... ; HU] (with actual noise, not wcn)
   sum-Ly x Lx x N
 AU: Block-row square-root discrete modulators, [A1 ... AU]
    Lx  x (U * Lx) x N
 Lyu: # of (output, Lyu) 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 precoder matrices: (Lx U) x (Lx U) x N
    For each of U users, this is Lx x Lx matrix on each tone
 S0: sub-channel dimensional channel SNRs: (Lx U) x (Lx U) x N
 MSWMFunb: users' unbiased diagonal mean-squared whitened matched matrices
   For each of U cells and Ntones, this is an Lx x Lyu  matrix
 Bu - users bits/symbol 1 x U
   the user should recompute SNR if there is a cyclic prefix
 B - the user bit distributions (U x N) in cell array
```

Set $N = 1$ (for now)

```
>> H =
   80
   50
>> Lyu=[1 1];
>> [Bu, Gunb, S0, MSWMFunb] = mu_bc(H, [1/sqrt(2) 1/sqrt(2)], Lyu, 2)
Bu =
   5.8222    0.4997
>> Gunb{:,:} =
   1.0000    1.0000
---
        0         1
S0 =
  2 x 1 cell array
    {[3.2010e+03]}    User 1 SNR
    {[    1.9992]}    User 2 SNR
MSWMFunb =
  2 x 1 cell array
    {[0.0177]}        multiplies y1
    {[0.0283]}        multiplies y2
>> sum(Bu) =   6.3219
>> [Bu, GU, S0, MSWMFunb , B] = mu_bc(H, [1 0], 1 , 2);
>> B =  1 × 1 cell array   {[6.3220]}
```

Equal energy on both users

G adds user 2 to user 1 – we already knew this

Add nothing to user 2

Receivers are each simple scaling

- Same values as blue rate vector on L9:28
- For this channel the rate sum is already close to maximum which occurs at $b = 6.3220$

More Examples

```
 H=[50 30
10 20];
>> A =
   0.5000        0   0.5000        0
        0   0.5000        0   0.5000
[Bu, Gunb, S0, MSWMFunb] = mu_bc(H, A, [1 1], 2);
Bu =
   4.8665   0.4971
>> Gunb{:,:} =
   1.0000   0.6000   1.0000   0.6000
        0   1.0000   1.6667   1.0000
 -----------------
        0        0   1.0000   2.0000
        0        0        0   1.0000
>> MSWMFunb{:,:} =
   0.0400
   0.0667
------
   0.2000
   0.1000
```

user 1's own xtalk and user 2 also

user 2's own xtalk

Each receiver estimates 2 input dimensions for its user, each a subuser.

```
>> >> S0{:,:} =
  626.0000        0
         0   1.3594
------------------
    1.1984        0
         0   1.6623

>> sum(Bu) =   5.3636
```

User 1's dimensional SNR's

User 2's dimensional SNR's

- mu_bc.m solves two MMSE problems here (for receiver 1 and receiver 2).
- It also aggregates them into right places in single matrix (cell array) of feedback/precoder, receiver filters.
- The receiver filters' rows apply to only their specific user/component (subuser) through MSWMFunb.

Worst Case Noise BC Design

Sections 2.8.3.2 and 2.8.3.5

$L_{y,u} = 1$ Case: finding the primary components

- Find each user's normalized channel $\tilde{H}_u \triangleq R_{\boldsymbol{nn}}^{-1/2}(u) \cdot H_u$

- Later → a general $(L_{y,u} > 1)$ way that uses worst-case noise; however, $L_{y,u} = 1$ is simpler to describe

$$y_u = \tilde{h}_{u,1} \cdot x_1 + \cdots + \tilde{h}_{u,L_x} \cdot x_{L_x}$$

- Find largest BC element
  - That is user i_1 and is first in order π

$$h_{max} = \max_{i,j} \left| \tilde{h}_{i,j} \right| \; i \in I_{BC} \wedge j \in J_{BC}$$

- Find next largest channel gain with user 1 as noise

$$\tilde{h}_{max} = \max_{i,j} \left| \tilde{h}_{i,j} \right|^2 / (\left| \tilde{h}_{i,i_1} \right|^2 + 1) \; \forall \; i \in \{I_{BC} \setminus i_1\} \wedge j \in \{J_{BC} \setminus \{i_1\}\}$$

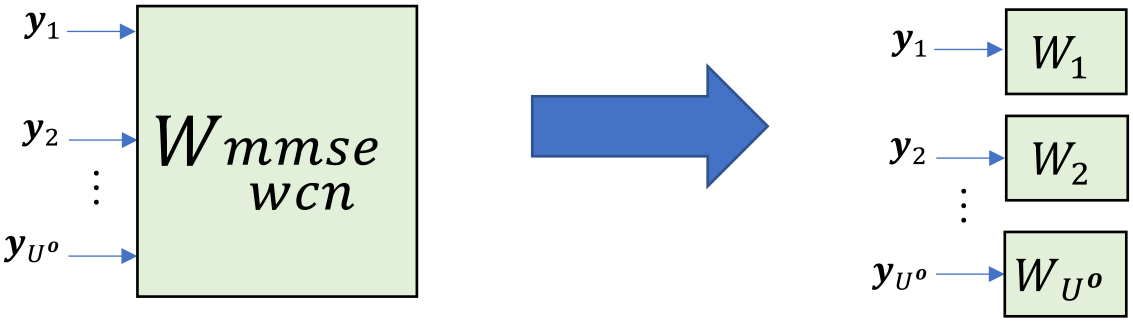
  - That is user i_2 and is second in order π

- Continue recursively $U^o = \wp^H$ times

- Any energy on users $\{\min(L_x, \wp_H) + 1, \ldots , U\}$ reduces rate sum and comes from secondary components.

Worst-Case Noise (2.8.3.3)

- "Worst-case" noise has covariance $R_{\boldsymbol{nn}}$ that minimizes $\mathcal{I}(\boldsymbol{x};\boldsymbol{y})$ for a fixed $R_{\boldsymbol{xx}}$
  - Only the receivers' local noises $R_{\boldsymbol{nn}}(u)$ are fixed, but the correlation between different user/receivers' noise may vary
- Thm: $R_{wcn}^{-1} - [H \cdot R_{\boldsymbol{xx}} \cdot H^*]^{-1} + R_{psd} = \mathcal{S}_{wcn}$ where $\mathcal{S}_{wcn}$ is $\mathfrak{L}_y \times \mathfrak{L}_y$ block (sub-block sizes $L_{y,u}$) diagonal
  - Further: $\wp_{R_{wcn}} = \wp_{\mathcal{S}_{wcn}}$ = # of primary components = U^0 .
  - Further: The secondary components correspond to $\mathcal{S}_{wcn}$'s zeroed diagonal elements (equivalently nonzero elements correspond to primary components).
  - R_{psd} is a Lagrange constraint for the positive semidefinite nature of the worst-case noise, and $\mathcal{S}_{wcn}$ is the Lagrange constraint for the block diagonal noises. R_{psd} is absent unless R_{wcn} is singular – many treatments ignore the singular case.
- Proof: See notes (also Appendix C on matrix calculus)
- When noise has R_{wcn} , the MMSE estimate $\hat{\boldsymbol{v}} = W \cdot \boldsymbol{y}$ has W that **IS DIAGONAL (block)**

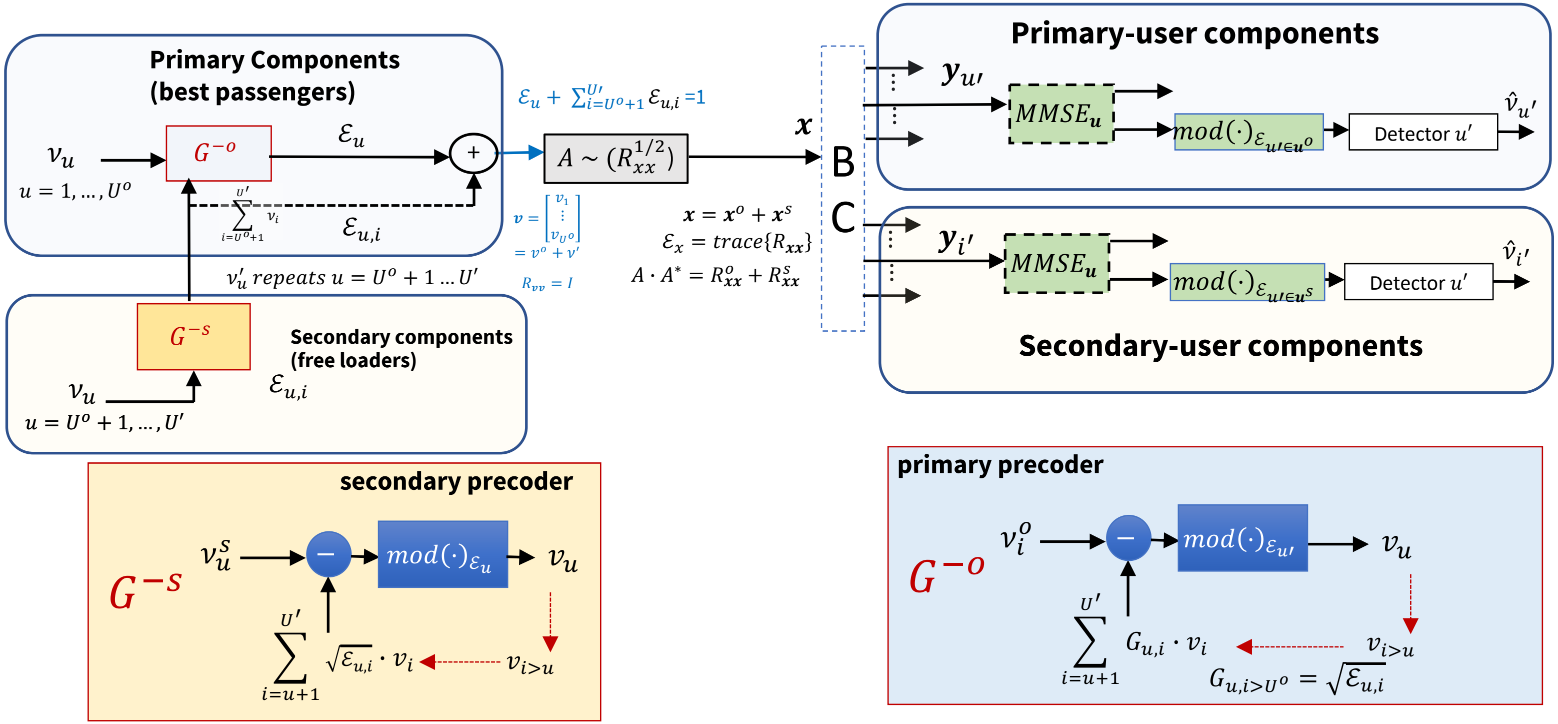


So, WCN corresponds to best BC receiver(s)
$b = \mathcal{I}_{wcn}(\boldsymbol{x};\boldsymbol{y})$

- Energized secondary components simply "free load" on the primary-components' dimensions

Remove Singularity and now Precode

- Now that primary (sub-) users are known, SVD on original channel (no noise-whiten) $H \cdot R_{xx}^{1/2} \triangleq [F \quad \boldsymbol{f}] \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} M^* \\ \boldsymbol{m}^* \end{bmatrix}$
- Input transforms to $\boldsymbol{x} \to M \cdot \boldsymbol{v}$ (M becomes part of square root and relabel); $H \cdot R_{xx}^{1/2} \to F\Lambda$



Mohseni's nonsingular-WCN Program (no CVX)

```
function [Rwcn, bsum] = wcnoise(Rxx, H, Ly, dual_gap, nerr)

 inputs
 H is U*Ly by Lx, where
    Ly is the (constant) number of antennas/receiver,
    Lx is the number of transmit antennas, and
    U is the number of users. H can be a complex matrix
 Rxx is the Lx by Lx input (nonsingular) autocorrelation matrix
    which can be complex (and Hermitian!)
 dual_gap is the duality gap, defaulting to 1e-6 in wcnoise
 nerr is Newton's method acceptable error, defaulting to 1e-4 in
    wcnoise

 outputs
 Rwcn is the U*Ly by U*Ly worst-case-noise autocorrelation matrix.
 bsum is the rate-sum/real-dimension.

 I = 0.5 * log(det(H*Rxx*H'+Rwcn)/det(Rwcn)), Rwcn has Ly x Ly diagonal blocks
   that are each equal to an identity matrix
```

- Example

```
>> H=[80
50] =
   80
   50          (noise-whitened/normalized channel)
>> [Rwcn,b]=wcnoise(1,H,1,1e-6,1e-4)
Rwcn =
   1.0000    0.6250
   0.6250    1.0000
b =    6.3220.
```

```
>> Htilde=inv(Rwcn)*H =
  80.0000
   0.0000
>>Swcn=inv(Rwcn)-inv(H*H' + Rwcn). =
   0.9998    0.0000
   0.0000    0.0000
>> Ryy=H*H'+Rwcn =   1.0e+03 *
   6.4010    4.0006
   4.0006    2.5010
>> SNRp1=det(Ryy)/det(Rwcn)  = 6.4010e+03
>> 0.5*log2(SNRp1)  =    6.3220 (checks!)
```

- $S_{wcn}(u,u)$ = sensitivity to $R_{\boldsymbol{nn}}(u)$ change, if = 0 ➔ secondary user
- Note WCN program permits easy sketch of the capacity region
  - Hint: you know noise-free user 1, and you also know the sum, so user 2's rate is the difference (PS5.3)

Singular 3x3 BC, nonsingular WCN

```
>> H =

   80   60   40
   60   45   30
   20   20   20

>> Rxx =

    3    0    0
    0    4    0
    0    0    2

>> [Rwcn, b]=wcnoise(Rxx, H, 1, 1e-5 , 1e-4);

 >>Rwcn
   1.0000   0.7500   0.0016
   0.7500   1.0000   0.0012
   0.0016   0.0012   1.0000
>> b
  11.3777
>> Swcn=inv(Rwcn)-inv(H*Rxx*H'+Rwcn) =

   0.9995   0.0000   0.0000
   0.0000  -0.0000   0.0000
   0.0000   0.0000   0.9948
```

- Works for any Rxx if WCN produced is nonsingular

```
>> sr=[5 0 0
7 8 9
10 11 12];
>> Rxx=sr*sr' =
   25   35   50
   35  194  266
   50  266  365
>> [Rwcn, b]=wcnoise(Rxx, H, 1, 1e-5 , 1e-4);
>> Rwcn
   1.0000   0.7500   0.0002
   0.7500   1.0000   0.0001
   0.0002   0.0001   1.0000
>> b = 16.4029
>> Swcn=inv(Rwcn)-inv(H*Rxx*H'+Rwcn)

Swcn =

   0.9999   0.0000   0.0000
   0.0000  -0.0000  -0.0000
   0.0000  -0.0000   0.9996
```

- Note position of zeros – user 2 is a single secondary component
- Wcnoise.m is sufficient if WCN is nonsingular
- Nonsingular R_{xx} forces ID of secondary components (WCN nonsingular)

Liao's Generalized Worst-Case Noise (uses CVX)

function [Rnn, sumRatebar, S1, S2, S3, S4] = cvx_wcnoise(Rxx, H, Lyu)

cvx_wcnoise This function computes the worst-case noise for any given input autocorrelation Rxx and channel matrix.

Arguments:
- Rxx: input autocorrelation, size(Lx, Lx)
- H: channel response, size (Ly, Lx)
- Lyu: number of antennas at each user, scalar/vector of length U

Outputs:
- Rnn: worst-case noise autocorrelation, with white local noise
- sumRatebar: maximum sum rate/real-dimension
- S1 is the lagrange multiplier for the real part of Rnn diagonal elements. Zero values indicate secondary users.
- S2 is the imaginary part
- S3 is for the positive semidefinite constraint on Rwcn
- S4 is for a larger Schur compliment used in the optimization

also allows variable number of antennas/user – not so in wcnoise.m

S1 and S2 together are S_{wcn} ; S2 prevents quantization-error accumulation on imaginary part

S3 plus S4 together in upper $U \times U$ positions equal S_{wcn}

- This program accommodates singular WCN (which is common in well-designed systems)
- The S3 plus S4 are the R_{psd} value, which must add to the S_{wcn} (block) diagonal = S1, see text
- Secondary users only identified when $R_{\boldsymbol{xx}}$ is nonsingular or if singular, the best or "water-filling for WCN" case.

Singular WCN

- Can occur if input is lower rank (optimization)

```
>> H =
  0.4054 - 0.1990i   0.3641 + 0.6869i   3.6004 + 0.5569i   0.5318 + 0.0080i
  1.8406 + 1.3469i  -1.3014 - 1.1630i   2.7217 + 1.0820i   0.0947 - 1.0710i
 -2.3367 + 1.1594i  -0.3949 + 0.7899i  -1.4024 + 0.8380i   0.8085 + 0.3019i
  1.1210 + 1.4423i   0.2611 + 1.6376i   2.9534 - 0.3945i   0.2962 - 0.8347i

>> Rxx =
  0.1382 + 0.0000i   0.0077 - 0.0664i  -0.0701 + 0.0449i  -0.0594 - 0.0207i
  0.0077 + 0.0664i   0.2005 + 0.0000i  -0.0155 - 0.0152i   0.0281 - 0.0130i
 -0.0701 - 0.0449i  -0.0155 + 0.0152i   0.0522 + 0.0000i   0.0262 + 0.0288i
 -0.0594 + 0.0207i   0.0281 + 0.0130i   0.0262 - 0.0288i   0.0331 + 0.0000i
>> rank(H) =    3
>> rank(Rxx) =    2
>>[V, D]=eig(Rxx);
diag(D) = -0.0001  0.0001   0.1418    0.2821
>> Rxx=V(:,3)*V(:,3)'*D(3,3)+ V(:,4)*V(:,4)'*D(4,4);
rank(iRxx) = 2
[F,L,M]=svd(H);
diag(L) =  6.7116   3.1056   2.0988   0.0000
H=F(:,1)*M(:,1)'*L(1,1)+ F(:,2)*M(:,2)'*L(2,2)+F(:,3)*M(:,3)'*L(3,3);
rank(H) =    3
```

```
[>> [Rwcn, b , S1, S2, S3, S4] = cvx_wcnoise(Rxx, H, ones(1,4))
Rwcn =
  1.0000 + 0.0000i  -0.5084 + 0.0458i   0.1251 + 0.5916i  -0.0042 + 0.3065i
 -0.5084 - 0.0458i   1.0000 + 0.0000i  -0.4394 + 0.2473i  -0.6341 + 0.0932i
  0.1251 - 0.5916i  -0.4394 - 0.2473i   1.0000 + 0.0000i   0.7215 + 0.3923i
 -0.0042 - 0.3065i  -0.6341 - 0.0932i   0.7215 - 0.3923i   1.0000 + 0.0000i
>> rank(Rwcn) =    3
b =    0.8241
S1 =  4x1 cell array
   {[    0.2288]}
   {[    0.3149]}
   {[    0.3073]}
   {[5.9295e-09]}
S2 =  4x1 cell array
   {[0]}
   {[0]}
   {[0]}
   {[0]}
S3+S4(1:4,1:4) =
   0.2288        0        0        0
        0   0.3149        0        0
        0        0   0.3073        0
        0        0        0   0.0000
>> pinv(Rwcn)*H =   1.0e+09*
  1.0647 + 0.7898i  -0.1492 + 0.2605i   2.1940 + 0.5313i   0.1632 - 0.5248i
  0.4982 + 1.1166i  -0.2378 + 0.1418i   1.5228 + 1.4199i   0.3687 - 0.3479i
 -0.6578 + 1.1464i  -0.2754 - 0.1173i  -0.2699 + 2.2345i   0.5387 + 0.1003i
 -0.0000 - 0.0000i   0.0000 - 0.0000i  -0.0001 - 0.0000i  -0.0000 + 0.0000i
```

- It turns out though that 0 sensitivity with singular WCN may not identify secondary user
  - Looks like last user here
- It will if also maximum rate sum (next lecture)

3 Cases

Perfect MIMO: $U^o = U'$. This case is non-degraded since $U' = \varrho_H = \varrho_x$; there are no secondary components. Perfect MIMO users each have equal number of used transmitter dimensions (antennas) and total number of receiver dimensions (antennas). Perfect MIMO's (all-primary-component) dimensions each have a path largely (MMSE sense) free of other users' crosstalk, as becomes evident shortly. Essentially, all the user components get their own dimension.

Degraded (NOMA): $U^o < U'$. In NOMA, $U^o = \min(\varrho_h, \varrho_x) <= U'$ and secondary components have no dimensions (antennas) to themselves. In this case, energy-sharing (or sharing the secondary components' data rates on common dimensions) is necessary if the secondary components must carry non-zero information. However, the $H_{u \in \boldsymbol{u}^s}$ determine these secondary components' maximum reliably decodable rates, even though the primary-component receivers could reliably decode a higher rate for these secondary components.

Enlarged MIMO: $U^o > U'$. This case corresponds to at least one individual user's receiver having $L_{y,u} > 1$; there are multiple receiver dimensions (antennas) per user. There are two sub-cases:

1. $U < U^o < U'$ (degraded enlarged MIMO). In this case some components may share dimensions to receivers, and there are secondary components.
2. $U < U^o = U'$ (non-degraded enlarged MIMO). In this case, each component has at least one dimension that is largely free (MMSE sense) of crosstalk from other components.

When $U^o >> U$, enlarged (non-degraded) MIMO often has the name "**Massive MIMO**."

- Note Wi-Fi, 4G/5G, Vector DSL (sometimes called MU-MIMO) are all "perfect MIMO"
- Time-sharing (TDMA) really just increases channel rank until perfect MIMO
- NOMA is more general; Internet of Things is probably going to force it (if not huge # antennas)

Vector WCN-BC Design

PS5.3 - 2.30 Vector BC Design

WCN Design focuses on primary users

- Any secondary-user components“freeload” on the dimensions best used by primary-user components
- Delete the secondary-user components’ rows from $\tilde{H}$
- The precoder-coefficients’ design, which pre-inverts channel, depends on those primary components
  - Any energized secondary components “dimension-share” those primary dimensions (and reduce overall rate sum)
- This can provide BC insights
- Chapter 5 will find a way for any desired $\boldsymbol{b}' \in \mathcal{C}(\boldsymbol{b})$ to derive the $\{R_{\boldsymbol{xx}}(u)\}$, but the choice of $\boldsymbol{b}'$ (scheduling) may want to know about primary/secondary components

Only for primary users (→ nonsingular WCN)

- Use R_{wcn} directly
  - General $\mathcal{S}_{wcn}$ is block diagonal
  - Find A

$$R_{wcn}^{-1} - \left[H \cdot A \cdot A^* \cdot H^* + R_{wcn}\right]^{-1} = \mathcal{S}_{wcn}$$

- Indeed, that is the backward MMSE channel in there!
  - Q_{wcn} is also block diagonal

$$\mathcal{S}_{wcn} = R_{wcn}^{-1} - \left[R_{wcn}^{-1} - R_{wcn}^{-1} \cdot H \cdot A \left(I + A^* \cdot H^* \cdot R_{wcn}^{-1} \cdot H \cdot A\right)^{-1} A^* \cdot H^* \cdot R_{wcn}^{-1}\right]$$

$$Q_{wcn}^* \cdot \mathcal{S}_{wcn}' \cdot Q_{wcn} = R_{wcn}^{-1} \cdot H \cdot A \boxed{\underbrace{\left(I + A^* \cdot H^* \cdot R_{wcn}^{-1} \cdot H \cdot A\right)^{-1}}_{R_b \triangleq G^{-1} \cdot S_0^{-1} \cdot G^{-*}}} A^* \cdot H^* \cdot R_{wcn}^{-1} \qquad (2.432)$$

$$\underbrace{\mathcal{S}_{wcn}'}_{\text{diagonal}} = \underbrace{Q_{wcn} \cdot R_{wcn}^{-1} \cdot H \cdot A}_{\text{triangular inverse}} \cdot R_b \cdot \underbrace{A^* \cdot H^* \cdot R_{wcn}^{-1} \cdot Q_{wcn}^*}_{\text{triangular inverse}} \;, \qquad (2.433)$$

- QR factorization (primaries' channel)
  - Extract $A = R_{\boldsymbol{xx}}^{1/2}$ from tri inverse,
  - which is the forward channel

$$Q_{wcn} \cdot R_{wcn}^{-1} \cdot H = \left[\underbrace{\mathbf{0}}_{(L_x - U^o)\times U^o} \quad \underbrace{R}_{U^o \times U^o}\right] \left[\begin{array}{c} \boldsymbol{q} \\ \underbrace{Q^*}_{U^o \times L_x} \end{array}\right] = R \cdot Q^*$$

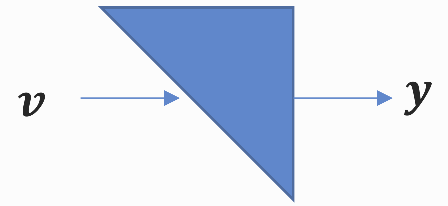
- Cholesky factorization (input)

$$\Phi \cdot \Phi^* = Q^* \cdot R_{\boldsymbol{xx}} \cdot Q \quad .$$

- A special square root!
  - "pre-triangularizes" the channel,
  - which becomes $R \cdot \Phi$

$$R_{\boldsymbol{xx}}^{1/2} = A = Q \cdot \Phi$$

Goal!



The precoder

- Want monic G for precoder

$$D_A \triangleq \mathrm{Diag}\{R \cdot \Phi\}$$

Find diagonal values

$$\begin{aligned} G &= D_A^{-1} \cdot R \cdot \Phi \\ S_0 &= D_A \cdot (S')^{-1}_{wcn} \cdot D_A \end{aligned}$$

Monic Equivalent

- Check = R_b for G and S_0

$$\begin{aligned} G^{-1} \cdot S_0^{-1} \cdot G^{-*} &= \left(\Phi^{-1} \cdot R^{-1} \cdot D_A\right) \cdot \left(D_A^{-1} \cdot S_{wcn} \cdot D_A^{-1}\right) \cdot \left(D_A \cdot R^{-*} \cdot G^{-*}\right) && (2.441) \\ &= \Phi^{-1} \cdot R^{-1} \cdot S_{wcn} \cdot R^{-*} \cdot \Phi^{-*} && (2.442) \\ &= \Phi^{-1} \cdot R^{-1} \cdot \left[Q_{wcn} \cdot R_{wcn}^{-1} \cdot H \cdot A \cdot R_b \cdot A^* \cdot H^* \cdot R_{wcn}^{-1} \cdot Q_{wcn}^*\right] \cdot R^{-*} \cdot \Phi^{-*} \\ &= \Phi^{-1} \cdot R^{-1} \cdot R \cdot Q^* \cdot Q \cdot \Phi \cdot R_b \cdot \Phi^* \cdot R^* \cdot R^{-*} \cdot A \cdot Q^* \cdot \Phi^{-*} && (2.443) \\ &= R_b \ . && (2.444) \end{aligned}$$

- Check SNR and mutual-info

$$\begin{aligned} 2^{\mathcal{I}_{wcn}(\boldsymbol{x};\boldsymbol{y})} &= \frac{|\, H \cdot R_{\boldsymbol{xx}} \cdot H^* + R_{wcn} \,|}{|\, R_{wcn} \,|} \\ &= |\, R_{wcn}^{-1/2} \cdot H \cdot R_{\boldsymbol{xx}} \cdot H^* \cdot R_{wcn}^{-*/2} + I \,| \\ &= |\, R_{wcn}^{-1/2} \cdot H \cdot A \cdot A^* \cdot H^* \cdot R_{wcn}^{-*/2} + I \,| \\ &= |\, A^* \cdot H^* \cdot R_{wcn}^{-1} \cdot H \cdot A + I \,| \quad \text{follows from SVD of } R_{wcn}^{-1/2} \cdot H \cdot A \\ &= |\, R_b^{-1} \,| \\ &= |\, S_0 \,| \\ \mathcal{I}_{wcn}(\boldsymbol{x};\boldsymbol{y}) &= \log_2\left(|S_0|\right) \ \text{ bits/complex subsymbol.} \end{aligned}$$

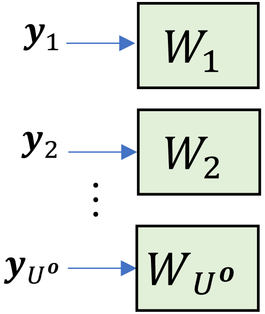
Works!

The Receiver

- The MMSE receiver is block diagonal!
  - For WCN only
  - Just what the BC needs

$$
\begin{aligned}
W &= \underbrace{S_0^{-1} \cdot G^{-*}}_{1-to-1} \cdot \underbrace{A^* \cdot H^* \cdot R_{wcn}^{-1}}_{\text{noise-white-match}} \cdot \underbrace{Q_{wcn}^* \cdot Q_{wcn}}_{I} \\
&= S_0^{-1} \cdot G^{-*} \cdot \Phi^* \cdot Q^* \cdot Q \cdot R^* \cdot Q_{wcn} \\
&= S_0^{-1} \cdot G^{-*} \cdot \Phi^* \cdot R^* \cdot Q_{wcn} \\
&= S_0^{-1} \cdot G^{-*} \cdot G^{-1} \cdot D_A \cdot Q_{wcn} \\
&= S_0^{-1} \cdot D_A \cdot Q_{wcn} \quad ,
\end{aligned}
$$

- Same bias removal as with all MMSE



BC WCN-Design Steps Summary (2.8.3.3)

Special Square Root

- Find R_{wcn} - this step also finds $\mathcal{S}_{wcn}$ and also the primary/secondary users and $b_{max}(R_{\boldsymbol{xx}})$
  - Delete rows/columns (secondary sub user dimensions) with zeros from $\mathcal{S}_{wcn}$, and correspondingly then in R_{wcn}
- If $\mathcal{S}_{wcn}$ is non-trivial (block diagonal), form $\mathcal{S}_{wcn}$=$Q^*_{wcn} \cdot \mathcal{S}'_{wcn} \cdot Q_{wcn}$ (eigen decomp)
- Perform QR factorization on $Q_{wcn} \cdot R^{-1}_{wcn} \cdot H = R \cdot Q^*$ where R is upper triangular, and Q is unitary
- Perform Cholesky Factorization on $Q^* \cdot R_{\boldsymbol{xx}} \cdot Q = \Phi \cdot \Phi^*$ where Φ is also upper triangular
- And now, the special square root is $R^{1/2}_{\boldsymbol{xx}} = Q \cdot \Phi$ (see diagram last page = A)

Precoder and Diagonal Receiver

- Find the diagonal matrix D_A =Diag$\{R \cdot \Phi\}$
- Find the (primary sub-user) precoder $G = D_A^{-1} \cdot R \cdot \Phi$ (monic upper triangular)
- Find the backward MMSE (block) diagonal matrix $S_0 = D_A \cdot (S')^{-1}_{wcn} \cdot D_A$ (note, $R_b^{-1} = G^* \cdot S_0 \cdot G$)
- Block diagonal (unbiased) receiver is $W_{unb} = (S_0 - I)^{-1} \cdot D_A \cdot Q_{wcn}$
- Can check but $b_{max}(R_{\boldsymbol{xx}})$ from WCN will be $\mathcal{I}_{wcn}(\boldsymbol{x};\boldsymbol{y})$=$\log_2|S_0| = \sum_{u=1}^{U^o} \log_2\left(1 + SNR_{BC,wcn,u}\right)$

Other data rate vectors *b* then share this system between primary/secondary

Example – all primary

- Energy $\mathcal{E}_x$ =2 , $L_x = 2$

```
>> H = [   80    70
           50    60 ];
>> Rxx=[1 .8
        .8 1];
```

```
>> [Rwcn,b]=wcnoise(Rxx,H,1)
Rwcn =
    1.0000    0.0232
    0.0232    1.0000
b = 9.6430
>> Swcn = inv(Rwcn)-inv(H*Rxx*H'+Rwcn) =
    0.9835    0.0000
    0.0000    0.9688
>> Htilde=inv(Rwcn)*H =
   78.8817   68.6440
   48.1687   58.4064
>> [R,Q,P]=rq(Htilde)
R =
  -12.4389  -74.6780
         0  -104.5673
Q =
    0.6565   -0.7544
   -0.7544   -0.6565
P =    2    1
```

Nonsingular Rwcn

ORDER IS REVERSED SO SWITCH USERS!

```
>> Rxxrot=Q'*Rxx*Q;
>> Phi=lohc(Rxxrot) =
    0.4482    0.0825
         0    1.3388
>> DA=diag(diag(R*Phi));
>> G=inv(DA)*R*Phi  =
    1.0000   18.1182
         0    1.0000
>> A=Q*inv(R)*DA*G =
    0.2942   -0.9557
   -0.3381   -0.9411
>> S0=DA*inv(Swcn)*DA =   1.0e+04 *
    0.0032   -0.0000
   -0.0000    2.0229
>> Wunb=(inv(S0)-eye(2))*DA =
    5.3983   -0.0000
   -0.0000  139.9857
```

Indeed Diagonal!

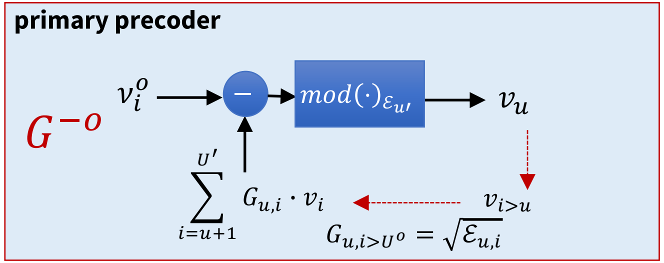
```
>> Gunb=eye(2)+S0*inv(S0-eye(2))*(G-eye(2)) =
    1.0000   18.7103
         0    1.0000
>> b=0.5*log2(diag(S0))' =   2.4909    7.1521
>> sum(b) =    9.6430 (checks
```

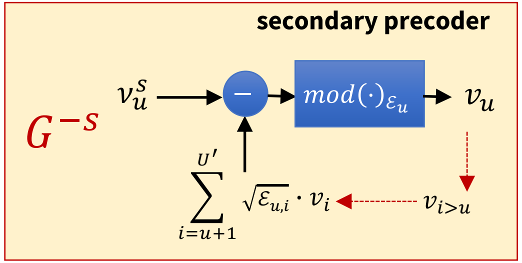
```
Wunb*H*A*inv(G) =
  -0.6987 -755.7235
 -780.3821 -454.9625
```

Try different
Input Rxx,
See text

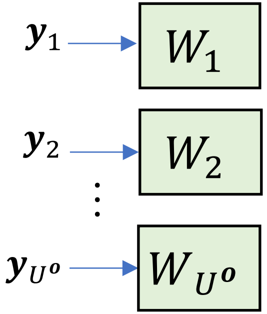
Return to Design

- The design can allocate R_{xx} energy to secondary and primary users as





- The receivers are easy



Another example – singular 3x3 BC (Ex 2.8.8)

```
>> H=[80 60 40
60 45 30
20 20 20];
>> rank(H)  =     2
>> Rxx=diag([3 4 2]);
>> [Rwcn1, b]=wcnoise(Rxx, H, 1, 1e-5 , 1e-4);
>> Rwcn1
    1.0000    0.7500    0.0016
    0.7500    1.0000    0.0012
    0.0016    0.0012    1.0000
>> b =   11.3777
>> Swcn=inv(Rwcn1)-inv(H*Rxx*H'+Rwcn) =
    0.9995    0.0000    0.0000
    0.0000   -0.0000    0.0000
    0.0000    0.0000    0.9948
```

User 2 is secondary – remove for now

```
>> H1=[H(1,1:3)
H(3,1:3)] =
   80   60   40
   20   20   20
>> [Rwcn, b]=wcnoise(Rxx, H1, 1, 1e-5 , 1e-4);
>> Rwcn =
   1.0000   0.0016
   0.0016   1.0000
>> b =  11.3777
>> Swcn=inv(Rwcn)-inv(H1*Rxx*H1'+Rwcn) =
   0.9995   0.0000
   0.0000   0.9948
```

Primary/Secondary

```
>> [R,Q,P]=rq(inv(Rwcn)*H1)
R =
      0   9.1016  -33.2537
      0        0 -107.6507
Q =
   0.4082  -0.5306  -0.7429
  -0.8165   0.1517  -0.5571
   0.4082   0.8340  -0.3713
P =    2    1
```

ORDER IS REVERSED (Here it is order of users 1 and 3 since 2 was eliminated)

```
>> R1=R(1:2,2:3);
>> Q1=Q(1:3,2:3);
>> Rxxrot=Q1'*Rxx*Q1  =
   2.3275   0.2251
   0.2251   3.1725
>> Phi=lohc(Rxxrot);
>> DA=diag(diag(R1*Phi)) =
  13.8379        0
        0 -191.7414
>> G=inv(DA)*R1*Phi  =
   1.0000   -4.1971
        0    1.0000
>> A=Q1*inv(R1)*DA*G =
  -0.8067   -1.3902
   0.2306   -0.9730
   1.2679   -0.5559
>> A*A'  =
   2.5833   1.1667  -0.2500
   1.1667   1.0000   0.8333
  -0.2500   0.8333   1.9167
```

Sq Root & Precoder

Not equal to Rxx
Energy not inserted into null space (same on part that is in pass space)

Rcvr & Data Rate

```
>> S0=DA*inv(Swcn)*DA =   1.0e+04 *
   0.0192   0.0000
   0.0000   3.6957
>> MSWMFunb=(inv(S0)-eye(2))*DA  =
  -13.7657    0.0000
   -0.0000  191.7362
>> Gunb=eye(2)+S0*inv(S0-eye(2))*(G-eye(2))  =
   1.0000   -4.2191
  -0.0000    1.0000
>> b=0.5*log2(diag(S0))'  =
   3.7909   7.5868
>> sum(b) =   11.3777 checks
>> H*A =
   0.0219 -191.8333
   0.0164 -143.8749
  13.8379  -58.3825
```

See Example 2.8.8 or details of below

Assign 1 energy unit to User 1, 1/3 to user 3, and now squeeze in 2/3 energy on user 2

```
>> b=0.5*log2(diag([ 1 1/3]) *diag(S0))  =
   3.7909
   6.7943
```

Crosstalk is >> ct=1/3*143.9^2 = 6.8928e+03

```
>> b2=0.5*log2(1+(2/3)*60^2/6892.8) =    0.2155
>> b2+sum(b) = 10.8007 < 11.3777
```

Energy on secondary reduces rate sum

System Diagram for this WCN design

$$
\begin{aligned}
v_1 &= \sqrt{\mathcal{E}_1}\cdot v_1^o + \sqrt{\mathcal{E}_{1,2}}\cdot v_2 \\
v_3 &= \sqrt{\mathcal{E}_3}\cdot v_3^o + \sqrt{\mathcal{E}_{3,2}}\cdot v_2
\end{aligned}
$$

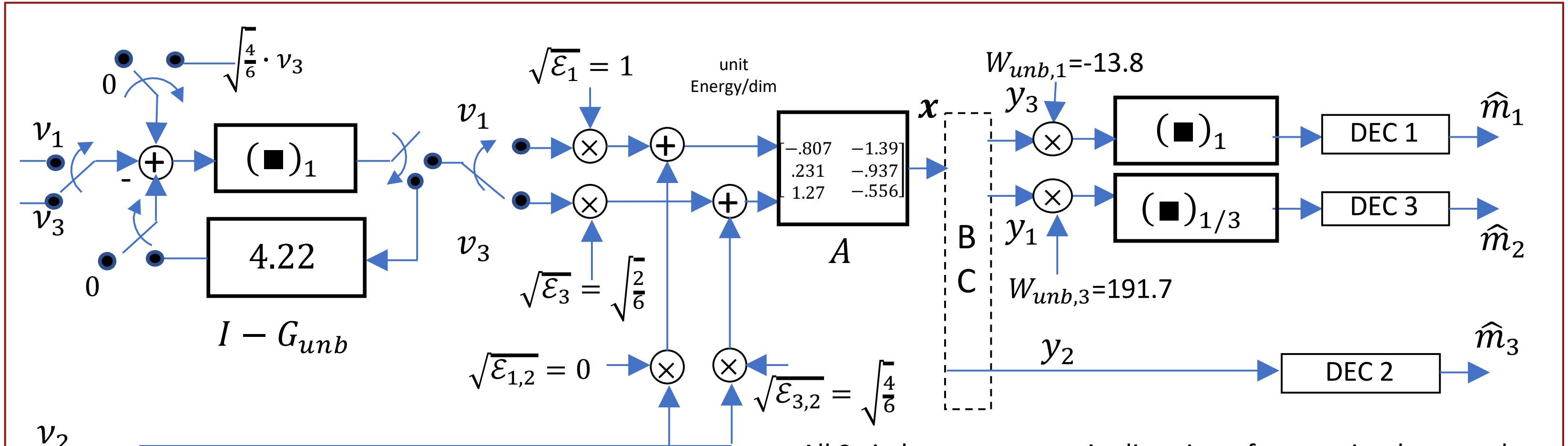
```
GU=   1.0000  -4.2191
     -0.0000   1.0000

MSWMFU=
 -13.7657   0.0000
  -0.0000 191.7362
```

See Ex 2.8.8

$$
\boldsymbol{x} = \underbrace{\begin{bmatrix} -0.8067 & -1.3902 \\ 0.2306 & -0.9370 \\ 1.2679 & -0.5559 \end{bmatrix}}_{A} \cdot \begin{bmatrix} \sqrt{\mathcal{E}_1} & \sqrt{\mathcal{E}_{12}} & 0 \\ 0 & \sqrt{\mathcal{E}_{23}} & \sqrt{\mathcal{E}_3} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}
$$

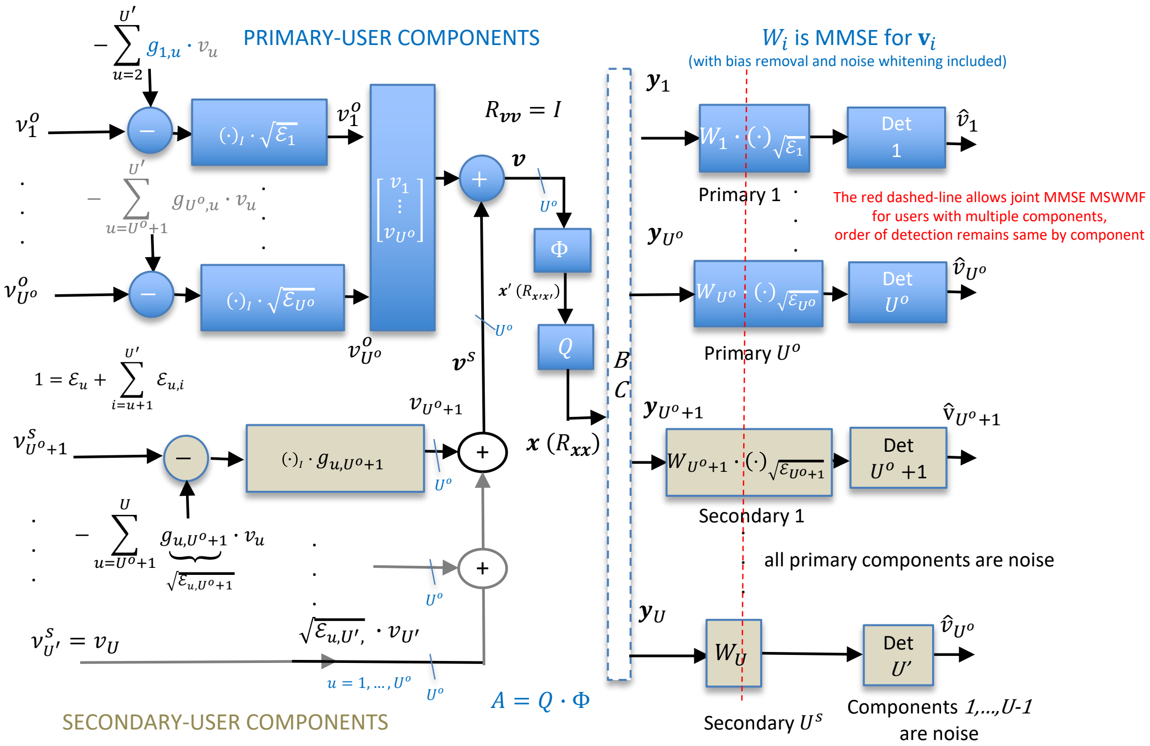
Try: $\mathcal{E}_1 = 1 \;\; and \;\; \mathcal{E}_{12} = 0$

$\mathcal{E}_3 = \frac{2}{6} \;\; and \;\; \mathcal{E}_{32} = \frac{4}{6}$



All Switches move once in direction of arrow simultaneously

Gaussian Vector BC System Diagram



- Works for any R_{xx} , but the square root $Q \cdot \Phi$ is very special and unique, and the design is for the R_{wcn} , no matter the real correlation between receiver noises; U' is number of user components

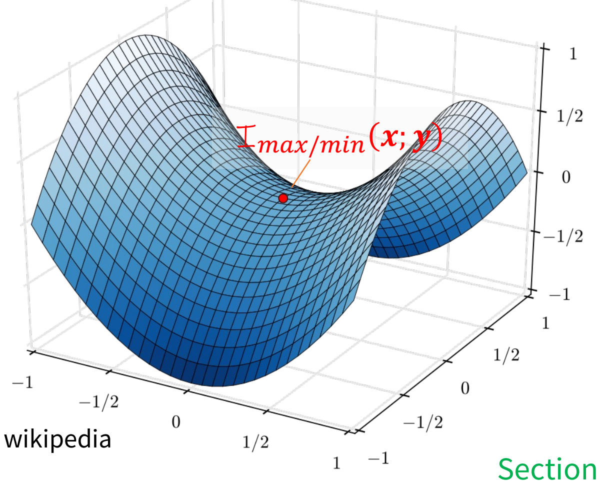
Maximum BC rate sum

Maximum BC rate sum

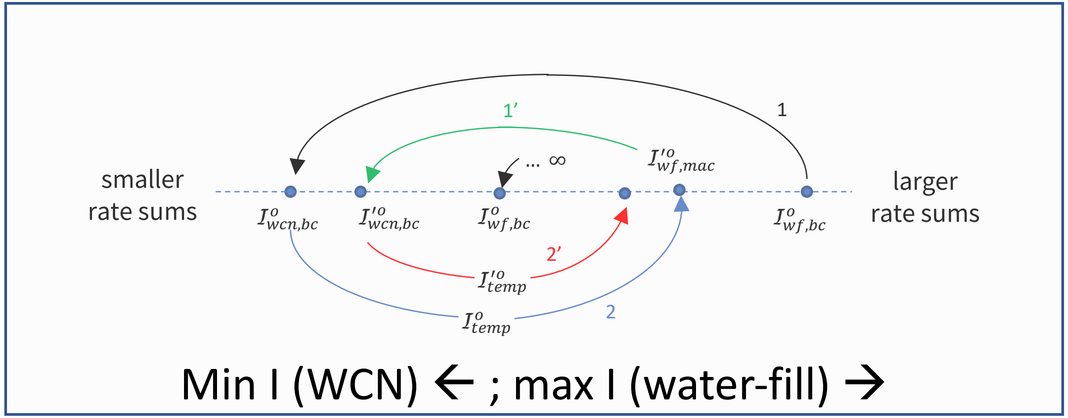
- **Maximize** $\mathcal{I}(\boldsymbol{x};\boldsymbol{y})$ through water-filling (but … presumes receivers can coordinate)
  - This is concave problem that always can be solved for the best input autocorrelation $R_{\boldsymbol{xx}}$

- **Minimize** $\mathcal{I}_{min}(\boldsymbol{x};\boldsymbol{y})$ through worst-case-noise to get $\mathcal{I}_{wcn}(\boldsymbol{x};\boldsymbol{y})$
  - This is also convex problem that always can be solved for worst noise autocorrelation R_{wcn}

- This is a saddle-point problem that produces a max-min = min-max



wikipedia



bcmax.m

```
function [Rxx, Rwcn, bmax] = bcmax(iRxx, H, Lyu)

 Uses cvx_wcnoise.m and rate-adaptive waterfill.m (Lagrange
 Multiplier based)
  Inputs:
    - iRxx: initial input autocorrelation array, size is Lx x Lx x N.
      Only the sum of traces matters, so can initialize to any valid
      autocorrelation matrix Rxx to run wcnoise.
      needs to include factor N/(N+nu) if nu ~= 0
    - H: channel response, size is Ly x Lx x N, w/o sqrt(N)
    normalization
    - Lyu: array number of antennas at each user; scalar Lyu means same for all

  Outputs:
    - Rxx: optimized input autocorrelation, Lx x Lx x N
    - Rwcn: optimized worst-case noise autocorrelation, with white local noise
        Ly x Ly x N
        so IF H is noise-whitened for Rnn, then actual noise is
        Rwcn^(1/2)*Rnn*Rwcn^(*/2)
    - b: maximum sum rate/real-dimension - user must mult by 2 for
       complex case
```

- Revisit example from slide 29

```
iRxx =
    3    0    0
    0    4    0
    0    0    2
>> H =
   80   60   40
   60   45   30
   20   20   20
[RxxA, RwcnA, bmax] = bcmax(iRxx, H, 1)
RxxA =
   3.7515   1.5032  -0.7451
   1.5032   1.5019   1.5007
  -0.7451   1.5007   3.7465
RwcnA =
   1.0000   0.7500   0.0008
   0.7500   1.0000   0.0006
   0.0008   0.0006   1.0000
bmax =  12.1084  (> 11.3777 that occurred earlier)
```

- No secondary components

Another example

```
H = [    80    70
         50    60 ];
>> iRxx=[1 .8
       .8 1];
>> [Rxx, Rwcn, bmax] = bcmax(iRxx, H, Lyu)

Rxx =
   1.0001    0.0082
   0.0082    0.9999
Rwcn =
   1.0000    0.0049
   0.0049    1.0000
bmax =   10.3517 > 9.6430
```

- Usually converges pretty quickly, not always though – CVX can get finicky when singularity involved

New Example – Singular Rwcnopt

```
>> H
   1.0719  -0.8627  -0.1901   0.2952
   1.0498  -0.7245   0.2568   0.2757
  -0.4586   0.5595   1.0027   0.0530
   0.4107   0.0496   0.3965  -0.7740
[F,L,M]=svd(H);
L =
   2.0671        0        0        0
        0   1.1449        0        0
        0        0   0.8130        0
        0        0        0   0.0000
H=F(:,1)*M(:,1)'*L(1,1)+ F(:,2)*M(:,2)'*L(2,2)+F(:,3)*M(:,3)'*L(3,3);

[Rxxopt, Rwcnopt, bmax] = bcmax(eye(4), H, 1)

Rxxopt =
   1.1559  -0.7381   0.0995  -0.0691
  -0.7381   0.6399   0.2862  -0.2207
   0.0995   0.2862   1.3091  -0.0326
  -0.0691  -0.2207  -0.0326   0.8951
Rwcnopt =
   1.0000   0.9163  -0.4792  -0.0191
   0.9163   1.0000  -0.0991   0.1251
  -0.4792  -0.0991   1.0000   0.1044
  -0.0191   0.1251   0.1044   1.0000
bmax =    2.1911
```

Nonsingular Rwcn

```
inv(Rwcnopt) - inv(H*Rxxopt*H'+Rwcnopt)
 -69.9931   62.4186  -26.7687   -6.3512
  62.4186  -54.9656   23.8385    5.6560
 -26.7687   23.8385   -9.5873   -2.4256
  -6.3512    5.6560   -2.4256   -0.1050
Rwcnopt, sumRatebar, S1, S2, S3, S4] =
cvx_wcnoise(Rxxopt, H, [1 1 1 1])
Rwcnopt =
   1.0000   0.9163  -0.4792  -0.0191
   0.9163   1.0000  -0.0991   0.1251
  -0.4792  -0.0991   1.0000   0.1044
  -0.0191   0.1251   0.1044   1.0000
sumRatebar =    2.1911
rank(H) =    3
>> S3+S4(1:4,1:4) =
   0.0978        0        0        0
        0   0.6204        0        0
        0        0   0.6360        0
        0        0        0   0.4705
rank(H) =    3
>> Htilde=pinv(Rwcnopt)*H;
>> [R,Q,P]=rq(Htilde);
R =
   0.0000   0.0211  -0.2595   0.1213
        0  -0.9411  -0.0435   0.0071
        0        0  -1.0542   0.0496
        0        0        0  -1.1499
P =    1    4    2    3
```

```
>> [V,D]=eig(Rxxopt)

V =

  -0.5342  -0.7457  -0.3635  -0.1626
  -0.7873   0.6052  -0.0344  -0.1124
   0.2071   0.2556  -0.9213   0.2073
  -0.2278  -0.1108   0.1336   0.9581


D =

   0.0000        0        0        0
        0   1.7105        0        0
        0        0   1.3639        0
        0        0        0   0.9256
```

End Lecture 10