Lecture 15

BC-MAC Duality Design

May 20, 2026

JOHN M. CIOFFI

Hitachi Professor Emeritus of Engineering

Instructor EE379B – Spring 2026

Announcements & Agenda

- Announcements
  - PS7 – last homework 6/4
  - Section 5.5
  - admMAC nominally works, but can run very long time
    - Use minPMACmimo if you experience this.

- Problem Set 7 = PS7 (due 5/30. extend to 6/4)
  1. 5.16 A tonal channel
  2. 5.17 GDFE MAC Design
  3. 5.18 Dual computations
  4. 5.19 GDFE BC design via duality
  5. 5.20 IC with/without GDFE

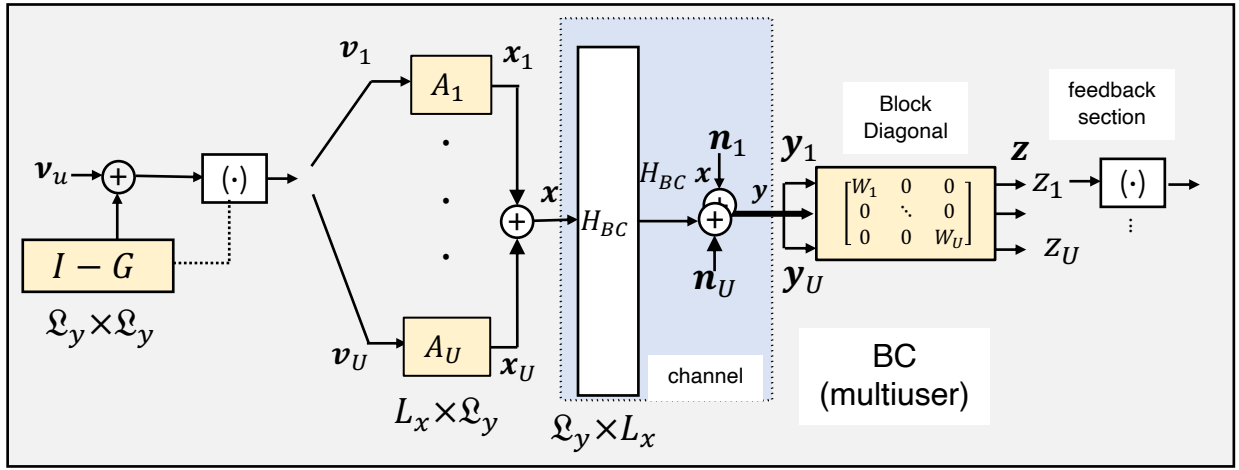
- Agenda
  - MAC/BC Duality Basics
  - Vector MAC/BC Duality
  - MAC-dual Design

MAC / BC Duality Basics

Section 5.5

BC Input Addition & Design

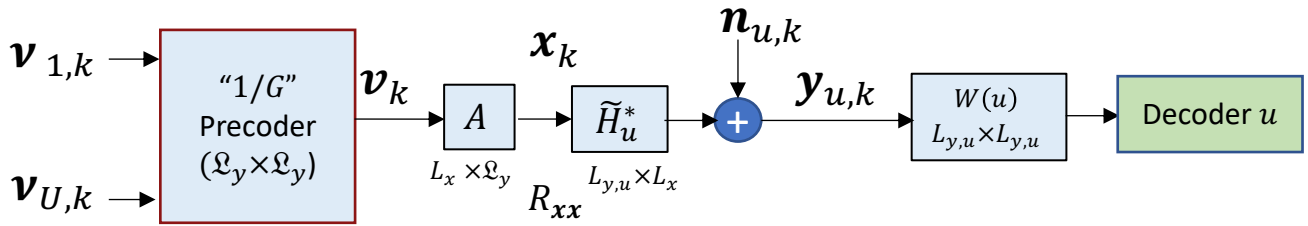
- The matrix-AWGN BC design adds user input symbols $\boldsymbol{x}_u$ before transmission.
  - $\boldsymbol{x} = \sum_{u=1}^{U} \boldsymbol{x}_u$ tends to "hide" independent input contributions.
    - Remember the "secondary-user component" precoder – "freeloading" compounds the hidden-subuser complexity – it is $\mathfrak{Q}_y \times \mathfrak{Q}_y$.
    - Primary users (or really any ϱ_H users) more productively separate.
  - $R_{\boldsymbol{xx}} = \sum_{u=1}^{U} R_{\boldsymbol{xx}}(u)$ - the contributions get "mixed" on the dimensions.
  - Key result: $R_b^{-1} = H_{BC}^* \cdot R_{\boldsymbol{nn}}^{-1} \cdot H_{BC} + I = G \cdot S_0 \cdot G^*$.



- Design:
  - Generates a single precoder, with
    - no concern for primary / secondary,
  - which is MMSE based,
  - which finds A's and G, (and W's), &
  - derives from **dual** MAC's design.

Determines essential set of subuser components

GDFE per user (useful for BC), mu_bc.m



k is subsymbol index.

Up to $\mathcal{Q}_y$ subuser components can affect decoder u (if $\mathcal{Q}_y > U^2$, design simplifies).

- Design achieves canonical performance for user u *for the given set of inputs and $\{\tilde{H}_u^*\}$ that is:*
  - *only a max rate sum if all- primary users with a corresponding special worst-case-noise-designed square root A.*
  - *But, this design is for specific $\{R_{\boldsymbol{xx}}(u)\}$.*

- The receiver $W(u)$ is indeed MMSE for these inputs and $\tilde{H}_u$.

- Max rate is $\mathcal{I}_{BC}(u) = \log_2 \left(\frac{\left| I + \sum_{i=1}^{u} H_u^* \cdot R_{\overline{xx}}(i) \cdot H_u \right|}{\left| I + \sum_{i=1}^{u-1} H_u^* \cdot R_{\overline{xx}}(i) \cdot H_u \right|} \right)$ and corresponds to the individual- $\tilde{H}_u$ MMSE GDFE.

- This is NOT a chain rule form because the H subscript is u, not i (but is a mini chain rule for rcvr u).
  - Thus, $\mathcal{I}(\boldsymbol{x}, \boldsymbol{y}) \neq \sum_{u=1}^{U} \mathcal{I}_{BC}(u)$; indeed $\mathcal{I}(\boldsymbol{x}, \boldsymbol{y}) \geq \sum_{u=1}^{U} \mathcal{I}_{BC}(u)$.

Specific to MAC and BC

- MAC Channel has normal notation:

$$\widetilde{H}_{MAC} = [\widetilde{H}_U \quad \cdots \quad \widetilde{H}_1] \qquad \mathcal{J}_{x\ or\ y} \triangleq \begin{bmatrix} 0 & 0 & I_{L_{x\ or\ y,1}} \\ 0 & \iddots & 0 \\ I_{L_{x\ or\ y,U}} & 0 & 0 \end{bmatrix}$$

- Dual BC Channel transposes each user channel and reorders outputs and inputs so 1 is at top/left:

$$\widetilde{H}_{BC} = \mathcal{J}_x \cdot \widetilde{H}^*_{MAC} \cdot \mathcal{J}_y = \mathcal{J}_x \cdot \begin{bmatrix} \widetilde{H}^*_U \\ \vdots \\ \widetilde{H}^*_1 \end{bmatrix} \cdot \mathcal{J}_y = \begin{bmatrix} \widetilde{H}^*_1 \\ \vdots \\ \widetilde{H}^*_U \end{bmatrix} \cdot \mathcal{J}_y$$

- The reversal allows some simplification of notation (its worse without the reversal).

See S15 for the mathematical derviation of duality

Scalar duality revisit and example

- Rewrite the scalar-duality input-deflection equations (follow from L10 scalar-energy duality):

$$\begin{aligned}
\mathcal{E}_1^{BC} &= \mathcal{E}_1^{MAC} \cdot \frac{1}{1+\mathcal{E}_2^{MAC}\cdot g_2+\ldots+\mathcal{E}_U^{MAC}\cdot g_U} \\
\mathcal{E}_2^{BC} &= \mathcal{E}_2^{MAC} \cdot \frac{1+\mathcal{E}_1^{BC}\cdot g_2}{1+\mathcal{E}_3^{MAC}\cdot g_3+\ldots+\mathcal{E}_U^{MAC}\cdot g_U} \\
\vdots &= \vdots \\
\mathcal{E}_U^{BC} &= \mathcal{E}_U^{MAC} \cdot \left(1+\left[\mathcal{E}_1^{BC}+\ldots+\mathcal{E}_{U-1}^{BC}\right]\cdot g_U\right)
\end{aligned}$$

- $g_u = \frac{|h_u|^2}{\sigma_u^2}$, above equations do not depend on (MAC's) g_1, which is BC's g_2.
- L9's duality example was for $H_{MAC} = \underset{\text{user 2}}{50} \quad \underset{\text{user 1}}{80}$

$$H_{BC} = \begin{matrix}80\\50\end{matrix} \begin{matrix}\text{user 1}\\\text{user 2}\end{matrix} \qquad \mathcal{E}^{BC} = \begin{bmatrix}3/4\\1/4\end{bmatrix} \rightarrow \begin{bmatrix}1/7504\\7503/7504\end{bmatrix} = \mathcal{E}^{MAC}$$

Simple example – treat BC as reversed MAC

$$\mathcal{E}^{BC} = \begin{bmatrix} 3/4 \\ 1/4 \end{bmatrix} \rightarrow \begin{bmatrix} 1/7504 \\ 7503/7504 \end{bmatrix} = \mathcal{E}^{MAC}$$

- Direct Design with MAC produces:

```
>> 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
```

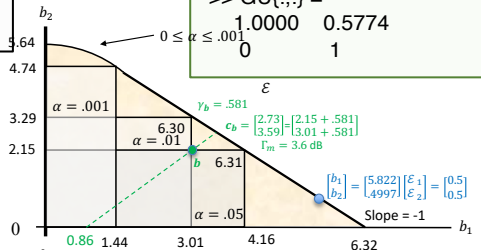
- Recall (sec 2.8) 6.322 is max sum rate (for this BC).

- Direct Design with BC produces:

```
>> Hbc=[80 ; 50];
>> AU=[sqrt(.75) sqrt(.25)][√user 1  …  √user 2]
>> 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
```

Crosstalk in BC scales @ xmit versus in MAC @ rcvr, but same cancellation.

MAX single user is $b = 6.56 > 6.322$.



BC Loss

- BC Loss - ratio of single-user capacity SNR to BC maximum-rate-sum SNR (for $[H \quad R_{nn}]$):

$$\gamma_{BC} \triangleq \frac{2^{2\cdot\bar{C}} - 1}{2^{2\cdot\bar{C}_{E-sum,dual-MAC}} - 1} = \gamma_{E-sum,dual-MAC} \geq 1$$

- This equality is assured by duality.

- For slide L15:12's example:

$$\gamma_{BC} = \frac{2^{2\cdot 6.556} - 1}{2^{2\cdot 6.322} - 1} = 1.5 \text{ dB.}$$

Try a different input on each dual

$$\mathcal{E}^{MAC} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \rightarrow \begin{bmatrix} 1/2502 \\ 2501/2502 \end{bmatrix} = \mathcal{E}^{BC}$$

special case with scalar and $U = 2$

```
>> 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

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
```

- Direct Design with BC

```
>> 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{:,:} =
    1.0000   50.0100
         0    1.0000

>> diag(cell2mat(S0)) =   1.0e+03 *
    0.0036         0
         0    1.2510

>> MSWMFunb{:,:} =
    0.6252
    0.0200

>> diag(cell2mat(MSWMFunb))*Hbc*AU =
    1.0000   50.0100
    0.0200    1.0000

>> sum(Bu) =   6.0600
```

Data rates & SNRs reverse in order.

Filters are not same.

Crosstalk cancels.

6.06 < 6.322 = previous rate sum because different input energy of [0.5 0.5] in this sum.

Vector MAC / BC Duality

Section 5.5.2

MAC2BC Full Algorithm

- **MAC2BC:** $R_{\overline{xx}}(u)=\mathcal{R}_{nn}^{-1/2}(u)\cdot F_u\cdot M_u^*\cdot \mathcal{R}_{\overline{nn}}^{-*/2}(u)\cdot R_{xx}(u)\cdot \mathcal{R}_{\overline{nn}}^{-1/2}(u)\cdot M_u\cdot F_u^*\cdot \mathcal{R}_{nn}^{-*/2}(u)$

- This finds the $R_{xx}(u)$ for the dual-BC's original MAC that has necessary data rate/energy.

$R_{xx}(u)$ from minPMACmimo

Given:

$R_{xx}(u)$ for $u=1,\dots,U$; $R_{\overline{xx}}(u)=R_{\overline{xx}}=0$

$\mathcal{R}_{nn}(U)=\mathcal{R}_{\overline{nn}}(1)=I$

BC: $\widetilde{H}_u^*, R_{nn}(u)$

MAC: $\widetilde{H}_u=R_{nn}^{-1/2}(u)\cdot H_u$

For $u=U,\dots,2$; $\mathcal{R}_{nn}(u-1)=\mathcal{R}_{nn}(u)+\widetilde{H}_u\cdot R_{xx}(u)\cdot \widetilde{H}_u^*$

For $u=1,\dots,U$

$\grave{H}_u=\mathcal{R}_{nn}^{-1/2}(u)\cdot \widetilde{H}_u\cdot \mathcal{R}_{\overline{nn}}^{-1/2}(u)$

$F_u\cdot \Lambda_u\cdot M_u^*=svd\left(\grave{H}_u\right)$

$R_{\overline{xx}}(u)=\mathcal{R}_{nn}^{-1/2}(u)\cdot F_u\cdot M_u^*\cdot \mathcal{R}_{\overline{nn}}^{*/2}(u)\cdot R_{xx}(u)\cdot \mathcal{R}_{\overline{nn}}^{1/2}(u)\cdot M_u\cdot F_u^*\cdot \mathcal{R}_{nn}^{-*/2}(u)$

$R_{\overline{xx}}=R_{\overline{xx}}+R_{\overline{xx}}(u)$

$\mathcal{R}_{\overline{nn}}(u+1)=\mathcal{R}_{\overline{nn}}(u)+\widetilde{H}_u^*\cdot R_{\overline{xx}}\cdot \widetilde{H}_u$; skip $u=U$

BC2MAC Full Algorithm

- **BC2MAC:** $R_{xx}(u)=\mathcal{R}_{\overline{n}\overline{n}}^{-*/2}(u)\cdot M_u\cdot F_u^*\cdot\mathcal{R}_{nn}^{-1/2}(u)\cdot R_{\overline{x}\overline{x}}(u)\cdot\mathcal{R}_{nn}^{-*/2}(u)\cdot F_u\cdot M_u^*\cdot\mathcal{R}_{\overline{n}\overline{n}}^{-1/2}(u).$

- Reverse is less interesting, but provided for completeness.

$R_{xx}(u)$ from a BC

Given:

$R_{\overline{x}\overline{x}}(u)$ for $u=1,\dots,U$; $\mathcal{R}_{nn}(u)=\mathcal{R}_{\overline{n}\overline{n}}(u)=0$

$\mathcal{R}_{nn}(U)=\mathcal{R}_{\overline{n}\overline{n}}(1)=I$; $R_{\overline{x}\overline{x}}=0$

BC: $\widetilde{H}_u^*, R_{nn}(u)$

MAC: $\widetilde{H}_u=R_{nn}^{-1/2}(u)\cdot H_u$

For $u=1,\dots,U-1$;

$R_{\overline{x}\overline{x}}=R_{\overline{x}\overline{x}}+R_{\overline{x}\overline{x}}(u)$

$\mathcal{R}_{\overline{n}\overline{n}}(u+1)=\mathcal{R}_{\overline{n}\overline{n}}(u)+\widetilde{H}_u^*\cdot R_{\overline{x}\overline{x}}\cdot\widetilde{H}_u$

For $u=U,\dots,1$

$\grave{H}_u=\mathcal{R}_{nn}^{-1/2}(u)\cdot\widetilde{H}_u\cdot\mathcal{R}_{\overline{n}\overline{n}}^{-1/2}(u)$

$F_u\cdot\Lambda_u\cdot M_u^*=svd\left(\grave{H}_u\right)$

$R_{xx}(u)=\mathcal{R}_{\overline{n}\overline{n}}^{-*/2}(u)\cdot M_u\,F_u^*\cdot\mathcal{R}_{nn}^{1/2}(u)\cdot R_{\overline{x}\overline{x}}(u)\cdot\mathcal{R}_{nn}^{*/2}(u)\cdot F_u\,M_u^*\cdot\mathcal{R}_{\overline{n}\overline{n}}^{-1/2}(u)$

$\mathcal{R}_{nn}(u-1)=\mathcal{R}_{nn}(u)+\widetilde{H}_u\cdot R_{xx}(u)\cdot\widetilde{H}_u^*$; skip $u=1$

Duality conversion program (Lx constant)

- Duality design uses the Rxxb = mac2bc(Rxxm , Hmac) program:
  - The input Rxxm is $L_x \times L_x \times U$ where L_x is for the MAC.
  - The input Hmac is $L_y \times L_x \times U$ where L_y AND Hmac are for the MAC
    - Hmac(:,:,1) is on the left and so equivalent to $\widetilde{H}_U$ - so then Hmac(:,:,U) is on the right
  - The output Rxxb is $L_y \times L_y \times U$ for the BC.
- With appropriate tensors, this I/O set can be repeated for each tone $n = 1, \ldots, \overline{N}$.

```
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;
0.5*log2(det([50 80]*diag([1/7504 7503/7504])*[50 80]'+1)) = 6.3220

Rxxb=mac2bc(Rxxm,Hmac)
Rxxb(:,:,1) =    0.7500
Rxxb(:,:,2) =    0.2500

Rxxm(1,1,1)=1/2;
Rxxm(1,1,2)=1/2;
Rxxb=mac2bc(Rxxm,Hmac)
Rxxb(:,:,1) =   .9998
Rxxb(:,:,2) = 1.5620e-04

>> [Rwcn , bsum]=wcnoise(1, [80 ; 50], 1)
Rwcn =
   1.0000    0.6250
   0.6250    1.0000
bsum =    6.3220
```

Matlab order is reverse of MAC (same as BC).

Note Hmac order corresponds to examples on slide 12

$b_2 = .2074 \quad b_1 = 6.116$

- What if variable $L_{x,u}$, and so then variable $L_{y.u}$, on the dual?
- Set $L_x = \max_u L_{x,u}$ and append $L_x - L_{x,u}$ zero columns to each H_u and columns/rows of each $R_{xx}(u)$.
- There will be corresponding zeroed columns/rows on $R_{\overline{xx}}(u)$ outputs.
- Duality algorithm's inverted/square-root matrices remain nonsingular.

Possible extra credit project
Variable-Lxu mac2bc / bc2mac

Reversal bc2mac program (Lx constant)

- Rxxm = bc2mac(Rxxb , Hmac) program has:
  - The input Rxxb is $L_x \times L_x \times U$ where L_x is for the dual MAC.
  - The input Hmac is $L_y \times L_x \times U$ is for the dual MAC (not the BC).
    - To reverse from mac2bc, input the mac2bc output (Rxxb) with Hbc= **conj**(permute(Hvec(:,:,end:-1:1), [order' 3])
  - The output Rxxm is $L_x \times L_x \times U$ for the dual MAC.
- With appropriate tensors, this I/O set can be repeated for each tone $n = 1, \ldots, \bar{N}$.

```
Rxxm=zeros(1,1,2);
Rxxm(1,1,1)=1/7504;   (1.3326e-04)    user 2
Rxxm(1,1,2)=7503/7504;  (0.9999)      user 1
Hmac=zeros(1,1,2);
Hmac(1,1,1)=50;
Hmac(1,1,2)=80;
Rxxb=mac2bc(Rxxm,Hmac)
Rxxb(:,:,1) =   0.7500
Rxxb(:,:,2) =   0.2500

bc2mac(Rxxb, Hmac)
ans(:,:,1) =   1.3326e-04
ans(:,:,2) =   0.9999
Checks reverses to original.

bsum = 6.322 on this channel.
```

```
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(Rxxb,Hmac)
ans(:,:,1) =    0.5000
ans(:,:,2) =    0.5000
```

```
conj( permute( Hmac(:,:,end:-1:1) , [2 1  3] ) )=
  80
  50

conj( permute( Hbc(:,:,end:-1:1) , [2 1  3] ) ) =
  50
  80
```

With constant Lx , and N=1, then the designer can find the Hbc by
Hbc=conj(**permute(** Hmac(:,:,end:-1:1) **, [2 1 3])**)
Or reverse from BC to MAC with
Hmac=conj(**permute(** Hbc(:,:,end:-1:1) **, [2 1 3])**)

- This last reverse step is usually not necessary because the designer optimizes for the dual MAC.
- This Rxxm then leads through mac2bc to Rxxb to complete the BC's needed input for design.

Example with 2 dimensions/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) =

   0.0036  -0.0050
  -0.0050   0.0074

Rxxb(:,:,2) =

   2.7951   0.9843
   0.9843   3.1939
```

```
>> bc2mac(Rxxb, Hmac)
ans(:,:,1) =
   1.0000   0.0000
   0.0000   1.0000

ans(:,:,2) =
   2.0000   1.0000
   1.0000   2.0000  (checks)
```

64-tone 3-user channel dual

```
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)); % input needs to be Ly x Lxu x U
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)+Rxxb(:,:,1,n))*Hbc(3,:,n)')));
end
bvec=sum(bbc') =  132.7477  412.8794  445.1264

>> bsum=sum(bvec) =  990.7535
```

- Equal energy used on MAC input here, so this example's dual and original MAC are not max rate sum.
  - But they are equal -- & close to max sum rate as well.
  - Note order reversal – user 1 is in best position on BC, but worst position in MAC – a feature of duality.

MAC-dual Design

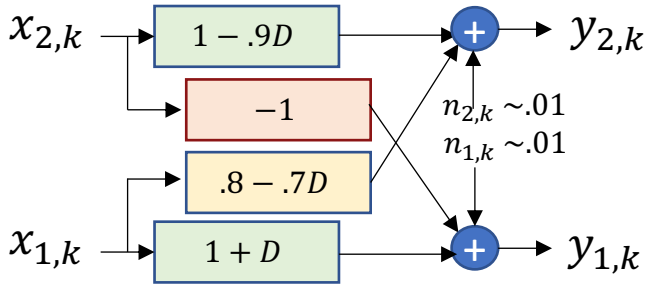
Section 5.5.4

Two-user channel with memory (Low/high pass)

$\mathcal{E}_2 = 1$

Per symbol

$\mathcal{E}_1 = 1$



$$H(D) = \begin{bmatrix} 1 - .9D & .8 - .7D \\ -1 & 1 + D \end{bmatrix}$$ complex baseband

Use 2×2 Vector DMT

```
h = cat(3, [1  .8 ; -1   1], [-.9  -.7 ; 0  1] )*10;
He = fft(h, 8, 3);        % (the matlab FFT increases energy)
```

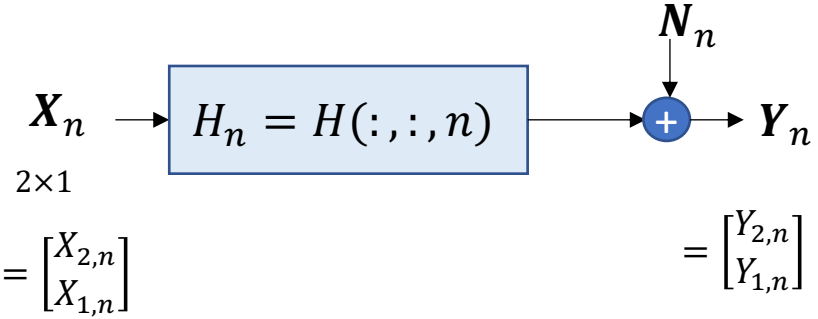
Cyclic prefix $\nu = 1$, energy loss 8/9

So far, just H →

8 tonal 2×2 channels

- Table

```
>> H
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
```



Single-User Upper Bound

- The highest data rate is the single-user capacity for the matrix AWGN – need SVD of big H.

```
Hblock=blkdiag(H(:,:,1),H(:,:,2),H(:,:,3),H(:,:,4),H(:,:,5),H(:,:,6),H(:,:,7),H(:,:,8));
g=svd(Hblock);
gains=(g.*g)' =
  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
```

- Water-fill on these singular-value-squared parallel channels ($\Gamma = 0$ dB):

```
>> 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
```

Single-user waterfill is close to equal energy on all tones and spatial dimensions – very high SNR.

```
>> 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
>>sum(bvec)/ 9 = 12.9633
```

There are now 8 MMSE-MAC GDFEs, 1 for each tone

- Repetitive process for each n, initialize/size:

```
GU=zeros(2,2,8);
WU=zeros(2,2,8);
S0=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
```

Note the sqrt(8)/3 on each of 8 dim's for each of 2 users. So 1/3 = 1/sqrt(9) for each Au a second factor of 8 matches unit-noise-whitening on 8 tones.

- Compute the MAC GDFE for each n,

```
>> for n=1:8
[Bu(:,n), GU(:,:,n) , WU(:,:,n),S0(:,:,n), MSWMFU(:,:,n)] = mu_mac(H(:,:,n), AU(:,:,n), [1 1], 1);
end
bvec=sum(Bu') = 62.3515   54.1393
Bsum =sum(bvec)   = 116.4908
```

- Bits/user/tone **10.1778**

```
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

bvec =  62.3515  54.1393
Bsum  = 116.4908
so then bsum/9 = 12.9434 bits/tone  or roughly 13 bits/Hz for both users
```

Why are single-user and MAC close?

(The flat high energy input on ALL dimensions means that SVD M might as well be I. Lots of square roots and one is I)

MAC Receiver Designs

- Unbiased total linear rcvr processing, each tone

```
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
```

- Unbiased feedback sections for each tone

```
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
```

What about the other vertex (puts user 1 at top)

- Design uses the same initialization because both users had same energy anyway

```
J=hankel([0 1]);
for n=1:8
Hflip(:,:,n)=J*H(:,:,n)*J;
end
>> for n=1:8
[Bu(:,n), GU(:,:,n) , WU(:,:,n),S0(:,:,n), MSWMF(:,:,n)] = mu_mac(Hflip(:,:,n), AU(:,:,n), [1 1], 1);
end
Bu
sum(Bu)
bvec=sum(Bu')
bsum=sum(bvec)
```

- The user bit rates are useful in duality n,

```
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 =  64.7687  51.7220
bsum =  116.4908 ,  so same (check)
```

10.1778

Rate sum is maintained on each tone, but decoding order is reversed

And the loss

- For this example, the data rate is already close to single-user WF.
- Answer for E=[1 1] using SWF:

```
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) =          Rxx(:,:,2) =          Rxx(:,:,3) =          Rxx(:,:,4) =
0.5500       0       0.9339       0       0.9401       0       0.9394       0
   0   0.8401       0       0.8991       0       0.8996       0       0.8954
Rxx(:,:,5) =          Rxx(:,:,6) =          Rxx(:,:,7) =          Rxx(:,:,8) =
0.9344       0       0.9394       0       0.9401       0       0.9339       0
   0   0.8829       0       0.8954       0       0.8996       0       0.8991
bsum =  116.5835

gamma_mac = 10*log10((2^(116.5835/9) -1)/(2^(116.4908/9)-1)) = 0.0310  dB

bsum_lin = 106.1991

Linear loss is
10*log10( (2^(116.4908/9)-1) / (2^(106.1991/9)-1) ) = 3.4430 dB
```

- Maximum Esum MAC rate sum:

```
[Rxx, bsum, bsum_lin] = macmax(16/9, h, [1 1], 8 , 1);
>> bsum =  116.5891  (so greater than Evec, but less than vector code 116.6695)

>> bsum_lin =  106.2249
```

- So, **116.5** is (also) best E-sum MAC rate sum (and also for Evec of [1 1] – pretty close already).

Order Reversal in Duality

- Semantics – alternate dual definition $\widetilde{H}_{dual}=\left(\mathcal{J}_y \cdot \widetilde{H} \cdot \mathcal{J}_x\right)^* = \mathcal{J}_x \cdot \widetilde{H}^* \cdot \mathcal{J}_y$.
  - Duality maps a single MAC output to a single input on BC.
  - $\mathcal{J}_y$ re-indexes dimensions (not users); but **no-$\mathcal{J}_y$-use** maps easily to matlab's usual indexing.
  - The $\mathcal{J}_y$ use simply makes the math look correct and symmetric, but confuses matlab programming.

$$\mathcal{J}_y \triangleq \begin{bmatrix} 0 & 0 & I_{L_{y,U}} \\ 0 & \iddots & 0 \\ I_{L_{y,1}} & 0 & 0 \end{bmatrix}$$

- All information, SVD, energy, etc are still preserved, as also true without $\mathcal{J}_y$ in $\widetilde{H}_{dual}$.

- The mac2bc and bc2mac programs basically handle $\mathcal{J}_y$ tacitly in reordering outputs, or inputs respectively.
  - L16's Hbc=conj(permute**(** Hmac(:,:,**end:-1:1)** **, [2 1 3])**) for $N = 1$; *command presumes* Hbc*'s input has dimension 1 at top.*
  - This tacitly multiplies by $\mathcal{J}_y$.

- Math Example:

$$H_{dual}(D) = \left(\mathcal{J}_y \cdot \underbrace{\begin{bmatrix} 1-.9D & .8-.7D \\ -1 & 1+D \end{bmatrix}}_{H(D)} \cdot \mathcal{J}_x \right)^* = \begin{bmatrix} 1+D^* & .8-.7D^* \\ -1 & 1-.9D^* \end{bmatrix}$$

$D^* \rightarrow e^{j\omega}$ on unit circle (FFT)

- Essentially:
  - User direct user [magnitudes, negated phase] remain the same, but priority reverses.
  - Crosstalk flow "flips-filter" from transfer of $u \rightarrow u'$ on original to $u \leftarrow u'$ (with negated phase).

Order Reversal in Duality

- Semantics – alternate dual definition $\widetilde{H}_{dual}=\left(\mathcal{J}_y \cdot \widetilde{H} \cdot \mathcal{J}_x\right)^* = \mathcal{J}_x \cdot \widetilde{H}^* \cdot \mathcal{J}_y$.
  - Duality maps a single MAC output corresponds to a single input on BC.
  - $\mathcal{J}_y$ re-indexes dimensions (not users); but **no-$\mathcal{J}_y$-use** maps easily to matlab's usual indexing.
  - The $\mathcal{J}_y$ use simply makes the math look correct and symmetric, but confuses matlab programming.

$$\mathcal{J}_y \triangleq \begin{bmatrix} 0 & 0 & I_{L_{y,U}} \\ 0 & \iddots & 0 \\ I_{L_{y,1}} & 0 & 0 \end{bmatrix}$$

- All information, SVD, energy, etc are still preserved, as also true without $\mathcal{J}_y$ in $\widetilde{H}_{dual}$.

- The mac2bc and bc2mac programs basically handle $\mathcal{J}_y$ tacitly in reordering outputs, or inputs respectively.
  - L16's Hbc=conj(permute**(** Hmac(:,:,**end:-1:1)** **, [2 1 3])**) for $N = 1$; *command presumes* Hbc*'s input has dimension 1 at top.*
  - This tacitly multiplies by $\mathcal{J}_y$.

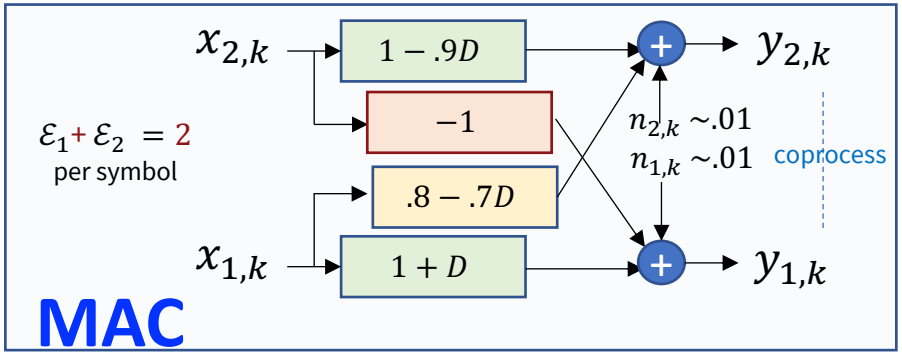
- Math Example:

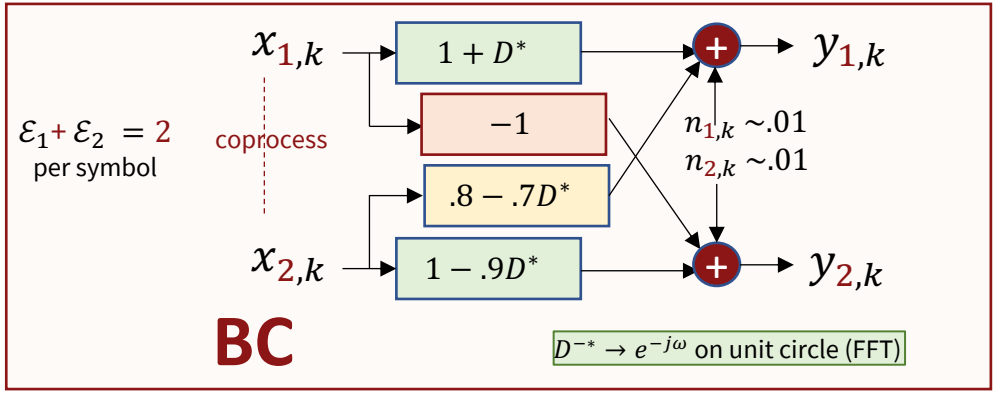
$$H_{dual}(D) = \left(\mathcal{J}_y \cdot \underbrace{\begin{bmatrix} 1-.9D & .8-.7D \\ -1 & 1+D \end{bmatrix}}_{H(D)} \cdot \mathcal{J}_x \right)^* = \begin{bmatrix} 1+D^* & .8-.7D^* \\ -1 & 1-.9D^* \end{bmatrix}$$

$D^* \rightarrow e^{j\omega}$ on unit circle (FFT)

- Essentially:
  - User direct user [magnitudes, negated phase] remain the same, but priority reverses.
  - Crosstalk flow "flips-filter" from transfer of $u \rightarrow u'$ on original to $u \leftarrow u'$ (with negated phase).

Dual Channels





- Coordination occurs on the other side.
  - **BC allows each user input to be 2D (2 sub-users/dim each).**

```
h = cat(3, [1   .8 ; -1   1], [-.9   -.7 ; 0   1] )*10;
H = fft(h, 8, 3);        % (the matlab FFT increases energy)

Hbc=zeros(2,2,8);
for n=1:8   % note N>1, so the permute command not applicable
Hbc(:,:,n)=H(:,end:-1:1,n)';  % note no Jy used here
end

>> for n=1:8
rho(n)=rank(H(:,:,n));
end
>> rho =    2    2    2    2    2    2    2    2
```

Almost same channel (D^*); but de/pre-coding order (priority) reverses

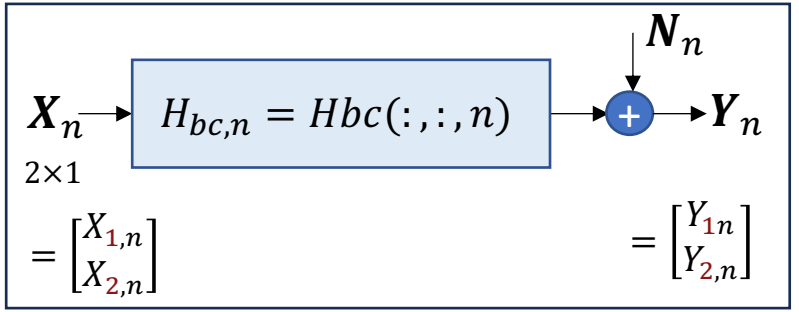
All tones have full rank.
(worst-case noise applies easily, but this design produces all Rxxb (u).)

With preliminaries now set ….

- Table

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

Now a BC, but still
there are
8 tonal 2×2 channels.



- Design uses duality and the Rxxb = mac2bc(Rxxm , H) program.
- With appropriate tensors, this I/O set can be repeated for each tone $n = 1, \ldots, 8$.

```
Rxxm=zeros(1,1,2);
Rxxm(1,1,:)=[8/9 8/9];
% same all n; thus, 3D tensor
   sufficient on this channel.

Rxxb=zeros(2,2,2,8); % 4D tensor
bbc=zeros(2,8);
```

The Rxxb's for the dual

```
for n=1:8
Rxxb(:,:,:,n)=mac2bc(Rxxm, reshape(H(:,:,n),2,1,2));
Hbc(:,:,n)=H(:,end:-1:1,n)';   % note N>1, so the permute command not applicable.
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
bvec=sum(bbc')
bsum=sum(bvec)
Rxxb;
```

```
Bu =
    6.5043
    3.6736
sum(Bu) =
```

```
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 so then 116.4908 /9 =   12.9434  bits/tone
```

Note output order reversal.

(The BC bvec is interpreted with user 2 at the bottom/right, So 62.315, whereas the MAC places it at the top/left.)

- Output
  - 16?
- 8 tones
  - 2x2 each user

```
>> 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
```

Duality works whether Rxx optimum or NOT

- Duality equates two **PER-USER** mutual-information quantities:
  - $\mathcal{I}_{MAC}(u\,/[u-1,\ldots,1]) = \mathcal{I}_{BC}(u)$.
  - $\mathcal{I}_{BC}(u) = \log_2\left(\frac{|I+\sum_i^u H_u^*\cdot R_{\overline{xx}}(i)\cdot H_u|}{|I+\sum_i^{u-1} H_u^*\cdot R_{\overline{xx}}(i)\cdot H_u|}\right)$ - this expression is only exact for H_u (not for H); it is not the chain rule.
  - It still corresponds to a MMSE estimator/decoder, implemented with lossless precoder, for user u.

- Chapter 2's worst-case noise finds an all-user BC (no receiver coordination) from MMSE-GDFE.

```
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  % alternative to chain-rule for BC
```

```
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 % some secondary user "freeloading"

Loss with respect to single-user
10*log10( (2^(116.6695 /9)-1) / (2^(116.4908 /9)-1) ) =   0.06 dB

Best BC rate sum, so with optimized input, is

[Rxx, Rwcn, bmax]=bcmax(Rxxbsum, Hbc, 1); output is bits/real-dimension
>> 2*bmax = 116.5892 > 116.4908 , but of course less than 116.6695

size(Rwcn) =   2   2   8
size(Rxx) =    2   2   8  % bcmax provides the wcn and the best Rxx (sum over users) for BC
```

Matches macmax (L16:31, as it should)

mu_bc.m – Precoder, Bloc-diag rcvr, given A

- The dual-BC design achieves the original MAC's target rates.
  - Dual-BC design does not use WCN.
  - Design U GDFEs (precoders) for each of the $\overline{N}$ tones (mu_bc program from Chapter 2, but now with $\overline{N}$ tones) & MSWMF, for $\{R_{xx}(u)\}$; any square root for each (AU)

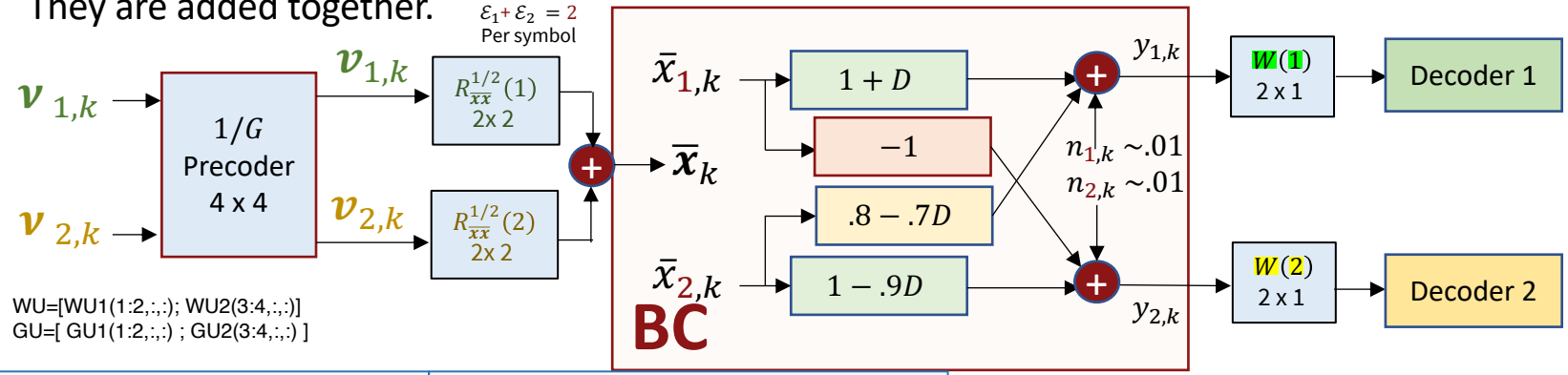
```
function [Bu, GU, S0, MSWMFunb , B] = mu_bc(H, AU, Lyu , cb)
-------------------------------------------------------------------------
 Inputs: Hu, AU , Usize, cb
 Outputs: Bu, Gunb, Wunb, S0, MSWMFunb

 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
```

Bu =
6.5043
3.6736
sum(Bu) =
from Dual-MAC

```
AU=zeros(2,4,8);
for n=1:8
AU(:,:,n)=[ sqrtm(Rxxb(:,:,1,n))  sqrtm(Rxxb(:,:,2,n)) ];
end

[Bu, Gunb, S0, MSWMFunb, B] = mu_bc(Hbc, AU, [1 1] , 1);
Bu =   54.1393   62.3515

>> B =  2×8 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}];  % convert to matrix form
end

MSWMFU=zeros(4,1,8);
for n=1:8
MSWMFU(:,:,n)=[MSWMFunb{1,n} ; MSWMFunb{2,n}]; % convert to matrix form
end
```

Precoders and Filters

- The transmit filters are the square-root matrices $R_{xx}^{1/2}(u)$, which are 2-dimensional for EACH user.
- They are added together.



GU(:,:,1) =

1.0000 + 0.0000i	0.1743 + 0.0000i	-0.6324 + 0.0000i	6.3243 + 0.0000i
0.0000 + 0.0000i	1.0000 + 0.0000i	-3.6274 + 0.0000i	36.2743 + 0.0000i
0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i	-10.0000 + 0.0000i
0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i

GU(:,:,2) =

1.0000 + 0.0000i	0.3662 + 0.6445i	-0.5488 - 0.1177i	0.2320 + 0.7298i
0.0000 + 0.0000i	1.0000 + 0.0000i	-0.5038 + 0.5653i	1.0106 + 0.2142i
0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i	-0.6768 - 1.1846i
0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i

GU(:,:,3) =

1.0000 + 0.0000i	0.9974 + 0.9035i	-0.0513 - 0.5181i	-0.2293 + 0.3117i
0.0000 + 0.0000i	1.0000 + 0.0000i	-0.2867 - 0.2598i	0.0292 + 0.2861i
0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i	-0.5525 - 0.4972i
0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i

GU(:,:,4) =

1.0000 + 0.0000i	1.6268 + 0.6419i	1.0890 - 1.3469i	-0.8561 + 0.4902i
0.0000 + 0.0000i	1.0000 + 0.0000i	0.2965 - 0.9450i	-0.3525 + 0.4404i
0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i	-0.5308 - 0.2064i
0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i

GU(:,:,5) =

1.0000 + 0.0000i	1.8789 + 0.0000i	3.5784 + 0.0000i	-1.8834 + 0.0000i
0.0000 + 0.0000i	1.0000 + 0.0000i	1.9046 + 0.0000i	-1.0024 + 0.0000i
0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i	-0.5263 + 0.0000i
0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i

GU(:,:,6) =

1.0000 + 0.0000i	1.6268 - 0.6419i	1.0890 + 1.3469i	-0.8561 - 0.4902i
0.0000 + 0.0000i	1.0000 + 0.0000i	0.2965 + 0.9450i	-0.3525 - 0.4404i
0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i	-0.5308 + 0.2064i
0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i

GU(:,:,7) =

1.0000 + 0.0000i	0.9974 - 0.9035i	-0.0513 + 0.5181i	-0.2293 - 0.3117i
0.0000 + 0.0000i	1.0000 + 0.0000i	-0.2867 + 0.2598i	0.0292 - 0.2861i
0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i	-0.5525 + 0.4972i
0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i

GU(:,:,8) =

1.0000 + 0.0000i	0.3662 - 0.6445i	-0.5488 + 0.1177i	0.2320 - 0.7298i
0.0000 + 0.0000i	1.0000 + 0.0000i	-0.5038 - 0.5653i	1.0106 - 0.2142i
0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i	-0.6768 + 1.1846i
0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	1.0000 + 0.0000i

MSWMFU(:,:,1) =
0.2960 + 0.0000i
1.6978 + 0.0000i
0.9222 + 0.0000i
-0.0922 + 0.0000i

MSWMFU(:,:,2) =
0.0616 + 0.0594i
0.1107 - 0.0327i
0.0716 + 0.1254i
-0.1058 + 0.0000i

MSWMFU(:,:,3) =
0.1051 + 0.0236i
0.0697 - 0.0395i
0.0586 + 0.0527i
-0.1060 + 0.0000i

MSWMFU(:,:,4) =
0.1855 - 0.0390i
0.0905 - 0.0597i
0.0562 + 0.0219i
-0.1059 + 0.0000i

MSWMFU(:,:,5) =
0.3217 + 0.0000i
0.1712 + 0.0000i
0.0556 + 0.0000i
-0.1056 + 0.0000i

MSWMFU(:,:,6) =
0.1855 + 0.0390i
0.0905 + 0.0597i
0.0562 - 0.0219i
-0.1059 - 0.0000i

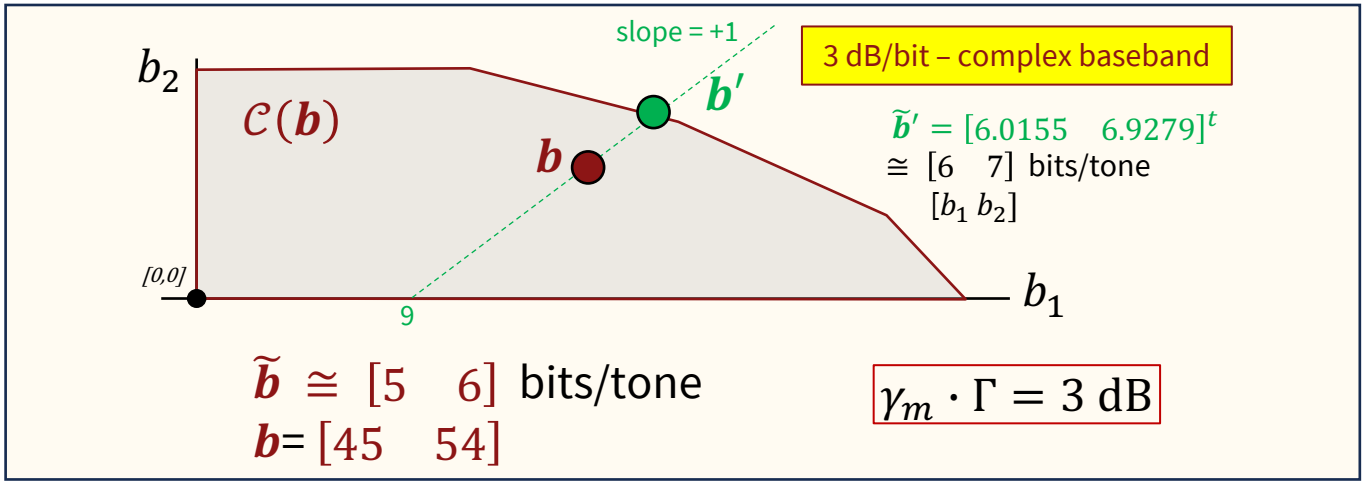
MSWMFU(:,:,7) =
0.1051 - 0.0236i
0.0697 + 0.0395i
0.0586 - 0.0527i
-0.1060 - 0.0000i

MSWMFU(:,:,8) =
0.0616 - 0.0594i
0.1107 + 0.0327i
0.0716 - 0.1254i
-0.1058 - 0.0000i

Design with Margin

- The rate vector $\boldsymbol{b}' = [54.1393 \quad 62.3515]^t \in \mathcal{C}(\boldsymbol{b})$'s boundary , and thus requires a $\Gamma = 0$ dB code.

- Use a code with $\Gamma = 1.5$ dB and leave 1.5 dB margin.



- The design (GDFE/precoder …) we just found with $\Gamma = 1.5$ dB code and 5 bits/Hz on user 2 and 6 bits/Hz on user 1 has 1.5 dB margin.
- This design works for any $\Gamma \cdot \gamma_m = 3$ dB if the energy for the point $\boldsymbol{b}'$ is used, but operated at rate $\boldsymbol{b}$.

Use of mac2bc - reminders

- It's use of svd "econ" mode is ok and implements the duality.
  - It's already in the mac2bc and bc2mac programs.
- The BC rate sum need not be the largest for the program's output Rxxb.
- If the input Rxxm is such that it corresponds to an MAC-Esum $\mathcal{C}(\boldsymbol{b})$ boundary point, then it is best.
  - But not necessarily at interior points, like those produced by admMAC and minPMACmimo for almost all admissible points (since most are not on the boundary).

Capacity Region

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 $\sum_u \mathcal{E}_u = \mathcal{E}$.

- Check any point's admissibility on the dual MAC with admMAC

End Lecture 15