
*Lecture 17*

# BC Design and the Central IC

***June 5, 2023***

JOHN M. CIOFFI

Hitachi Professor Emeritus of Engineering

Instructor EE392AA – Spring 2023

# Announcements & Agenda

- Announcements
  - PS7 – last normal homework extended to June 7
  - Section 5.6
  - PS6/7 solutions distributed individualy (PS7 on June 7) to those turned in. Please do not share.

End-term course evaluations are now open for students to provide feedback. Please direct your students to complete their feedback through any of the following options:

- in Canvas, students will see a pop-up notification on their Canvas dashboard page any time they log into Canvas during the evaluation period.
- direct link: http://course-evaluations.stanford.edu.
- in Axess > My Academics > Course and Section Evaluations > Link to the evaluation system near the top of the page.

Please refer to the table below for the deadline (Survey End Date) for students to submit their feedback as the schedule may be different for certain courses. **All times are PDT.** If you are teaching any courses where feedback opens later in the term, you will receive a separate announcement when those open.

**Course Summary**

| Course Code | Course Title | Survey Start Date | Survey End Date | Report Access Start | Response Rate |
|---|---|---|---|---|---|
| Sp23-EE-392AA-01 | DATA TRANSMISSION | 6/5/2023 8:00 AM | 6/19/2023 11:59 PM | 6/21/2023 12:00 AM | 0.00% (0/6) |

Students who complete all of their feedback will see their grades as soon as they have been submitted. Students who do not complete all of their feedback will not be able to see their grades in Axess until the day after the grade release deadline.

- Agenda
  - BC discrete design process
  - Matlab design-process review & Capacity Region construction
  - IC Review
  - CIC Optimization

# BC Discrete Design Process

# Order Reversal in Duality

- Semantics – alternate dual definition $\widetilde{H}_{dual}=\left(\mathcal{J}_y \cdot \widetilde{H} \cdot \mathcal{J}_x\right)^* = \mathcal{J}_x \cdot \widetilde{H}^* \cdot \mathcal{J}_y$
  - A single MAC output corresponds to a single input on BC, in duality.
  - $\mathcal{J}_y$ 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 actually 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 reording 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 inherently multiplies by $\mathcal{J}_y$ tacitly.

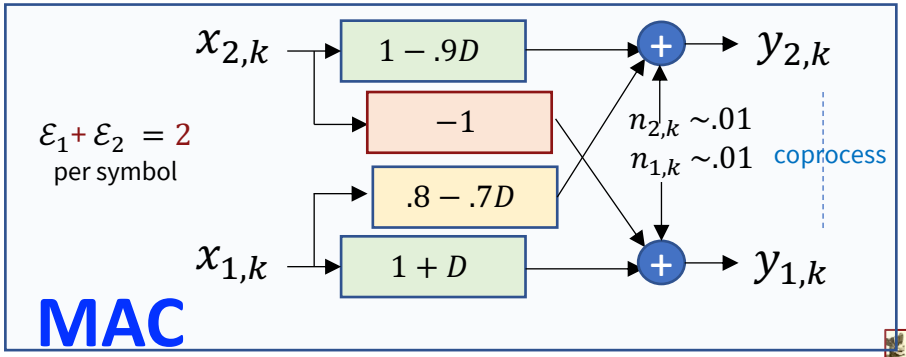
- Math Example:

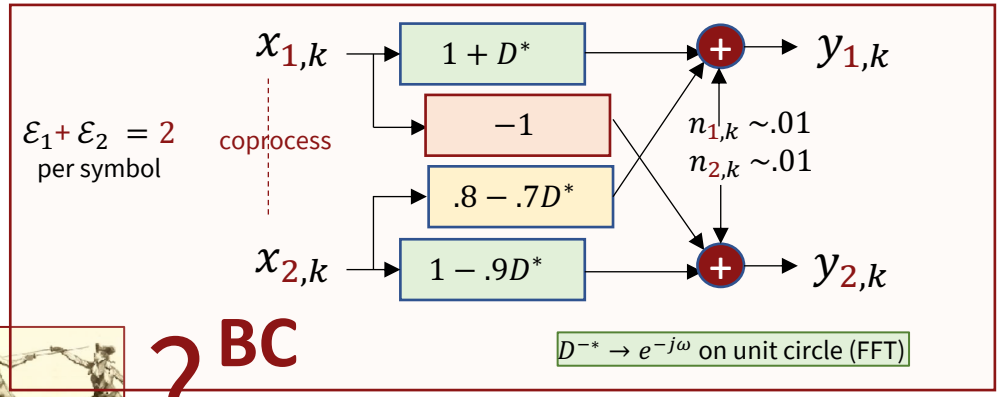
$$H_{dual}(D) = \left( \mathcal{J}_y \cdot \underbrace{\begin{bmatrix} 1-.9D & .8-.7D \\ -1 & 1+D \end{bmatrix}}_{H(D)} \cdot \mathcal{J}_x \right)^* = \begin{bmatrix} 1+D^* & .8-.7D^* \\ -1 & 1-.9D^* \end{bmatrix}$$

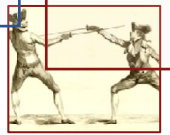
$D^* \rightarrow e^{j\omega}$ on unit circle (FFT)

- 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)

# Dualing Channels



? BC

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

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

```
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
```

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

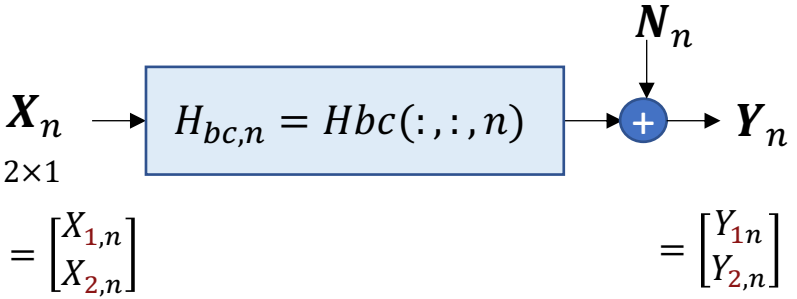
# With preliminaries now set ….

- Table

| | |
|---|---|
| Hbc(:,:,1) = | Hbc(:,:,5) = |
| 1.0000 + 0.0000i 20.0000 + 0.0000i | 15.0000 + 0.0000i 0.0000 + 0.0000i |
| 1.0000 + 0.0000i -10.0000 + 0.0000i | 19.0000 + 0.0000i -10.0000 + 0.0000i |
| Hbc(:,:,2) = | Hbc(:,:,6) = |
| 3.0503 - 4.9497i 17.0711 + 7.0711i | 12.9497 + 4.9497i 2.9289 - 7.0711i |
| 3.6360 - 6.3640i -10.0000 + 0.0000i | 16.3640 + 6.3640i -10.0000 + 0.0000i |
| Hbc(:,:,3) = | Hbc(:,:,7) = |
| 8.0000 - 7.0000i 10.0000 +10.0000i | 8.0000 + 7.0000i 10.0000 -10.0000i |
| 10.0000 - 9.0000i -10.0000 + 0.0000i | 10.0000 + 9.0000i -10.0000 + 0.0000i |
| Hbc(:,:,4) = | Hbc(:,:,8) = |
| 12.9497 - 4.9497i 2.9289 + 7.0711i | 3.0503 + 4.9497i 17.0711 - 7.0711i |
| 16.3640 - 6.3640i -10.0000 + 0.0000i | 3.6360 + 6.3640i -10.0000 + 0.0000i |

**Now a BC, but still**

**8 tonal 2×2 channels**




- Use 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 , so don't need
% 4D tensor for Rxxm

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, repeat from slide 32
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
```

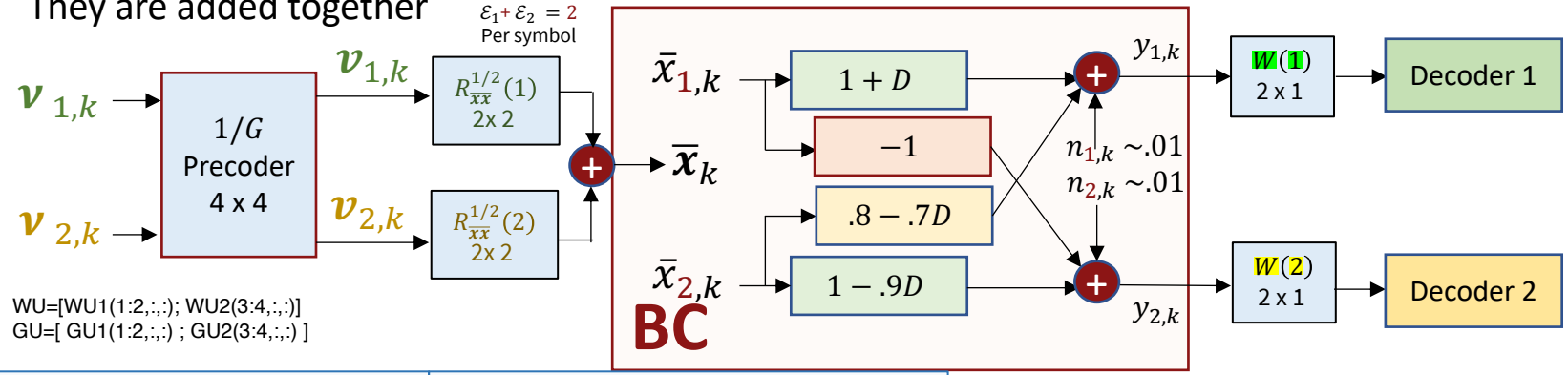
Bu =
6.5043
3.6736
sum(Bu) =

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




WU=[WU1(1:2,:,:); WU2(3:4,:,:)]
GU=[ GU1(1:2,:,:) ; GU2(3:4,:,:) ]

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

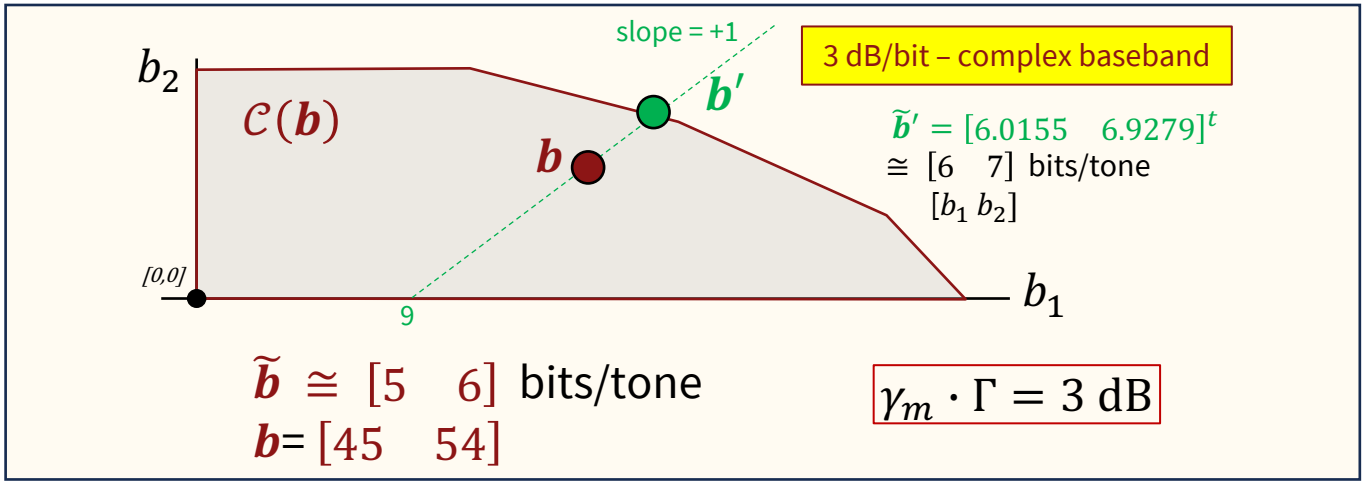
MSWMFU(:,:,1) =
MSWMFU(:,:,1) =
0.2960 + 0.0000i
1.6978 + 0.0000i
0.9222 + 0.0000i
-0.0922 + 0.0000i
MSWMFU(:,:,2) =
0.0616 + 0.0594i
0.1107 - 0.0327i
0.0716 + 0.1254i
-0.1058 + 0.0000i
MSWMFU(:,:,3) =
0.1051 + 0.0236i
0.0697 - 0.0395i
0.0586 + 0.0527i
-0.1060 + 0.0000i
MSWMFU(:,:,4) =
0.1855 - 0.0390i
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 $\Gamma = 0$ 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 minPMAC for almost all admissible points (since most are not on the boundary)

# Some Matlab Design-Process Help (PS7) & $\mathcal{C}(b)$ construction

# PS 7: $N > 1$, constant $L_x = L_{x,u} \equiv \mathfrak{L}_x/U$

- Generate Hbc that can be used with mu_bu program

```
Hmac=reshape(Hmac,Ly,Lxu,U,N);   %4D tensor from 3D tensor FFT output, which was Ly x U·Lx x N
Hmac=Hmac(:,:,order,:)
Hbc=zeros(Lxu,Ly,U,N);   % use MAC's Lxu and Ly !
for n=1:N   % note N>1
for i=1:U
Hbc(:,:,i,n)=Hmac(:,:,U+1-i,n)';
end
end
Hbc1=permute(reshape(Hbc, [Ly , 𝔏_x, N]), [ 2 1 3]); % returns to 3D tensor
```

- Generate Rxxm that can be used with mac2bc program

```
Info.Rxxs  % from slower minPMACMIMO is already (Lxu, Lxu, U, N) if Lxu is constant

E=celltomat(info.Eun) % so only need this if Lxu=1 because faster minPMAC was used. % Rxxm=cell2mat(info.Rxxs)
Rxxm=zeros(1,1,U,N); % for minPMAC
for n=1:N    Rxxm(1,1,:,n)=E(u, order,n);    end  % Since Lxu=1, conversion to mac2bc format (per tone) % any u=1,…,U works

Rxxb=zeros(Ly,Ly,U,N);
bbc=zeros(U,N);
for n=1:N   Rxxb(:,:,:,n)=mac2bc(Rxxm(:,:,:,n), Hmac(:,:,:,n));  end % per tone use of mac2bc 3D tensor to 3D tensor
```

- Variable $L_{x,u}$: set $L_x$ to $L_x = \max_u L_{x,u}$ and expand each $\widetilde{H}_u$ to have $L_x$- $L_{x,u}$ dummy zero columns

# HWH 7: BC Design Completion

- Data calculation as a double check, shown here for $U = 3$

```
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')
bsum=sum(bvec)
```

- Prepare for BC Design – stack the A matrices horizontally

```
A=zeros(Ly, Ly*U, N); % 3D tensor to match Hbc1 and mu_bc.m program
for n=1:N  A(:,:,n)=[ sqrtm(Rxxb(:,:,1,n))  …..  sqrtm(Rxxb(:,:,U,n)) ];  end

[Bu, Gunb, S0, MSWMFunb, B] = mu_bc(Hbc1, A, [Lyu] , cb);
```

- mu_bc outputs cell arrays, so allows variable (BC) $L_{y,u}$; when constant $L_{y,u} = L_y$ can translate for display

```
GU=zeros(U · 𝔏y, U · 𝔏y, N);  % 𝔏y corresponds to BC here
for n=1:N   GU(:,:,n)=[Gunb{1,n} ; …, Gunb{U,n}];  end
MSWMFU=zeros(U · 𝔏y, Ly, N);  for n=1:N  MSWMFU(:,:,n)=[MSWMFunb{1,n}; … ; MSWMFunb{U,n}];  end
```

- More generally GU cells are **each** $L_{y,u} \times L_{y,u}$ ; while MSWMFU are each $\mathfrak{L}_y \times L_{y,u}$;

# 64-tone - one vertex

```
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H64, [445 412 132]', [1 1 1]',1);
FEAS_FLAG =    2
bu_a =  445.0000    412.0000    132.0000
      %  bu_v                Eun          bun          theta    order  frac    cID
   445.81   411.19    132   {1x3x64 }    {1x3x64 }    {1x3}    {1x3}  0.99    1
   425.68   431.32    132   {1x3x64 }    {1x3x64}     {1x3}    {1x3}  0.01    1
>> info.order{:} =
    3    2    1
    3    1    2
>> info.frac'*info.bu_v =  445.0000  412.0000  131.9999
>> info.frac' =   0.9904    0.0096
>> sum(Eun') =   65.9553   60.2757   40.4453
>> sum(Eun,'all') =  166.6763 < 3 x 64
```

**Small < 1%**
**Might just go with vertex 1**
**Large/small → numerical issues**

```
H64mac =reshape(H64,2,1,3,64);
H64bc=zeros(1,2,3,64);
for n=1:64
for i=1:3   H64bc(:,:,i,n) = H64mac(:,:,4-i,n)'; end
end
H64bc1=permute(reshape(H64bc, [2 , 3, 64]), [ 2 1 3]);
3,n)) ];  end
```

```
Rxxm=zeros(1,1,3,64);
for n=1:64
E=cell2mat(info.Eun);
Rxxm(1,1,:,n)=E(1,:,n);    end
Rxxb=zeros(2,2,3,64);
bbc=zeros(3,64);
for n=1:64   Rxxb(:,:,:,n)=mac2bc(Rxxm(:,:,:,n), H64mac(:,:,:,n)); end
A=zeros(2, 6, 64);
for n=1:64  A(:,:,n)=[ sqrtm(Rxxb(:,:,1,n)) , sqrtm(Rxxb(:,:,2,n)),  sqrtm(Rxxb(:,:,3,n)) ];  end
```

```
>> [Bu, Gunb, S0, MSWMFunb, B] = mu_bc(H64bc1, A, [1 1 1], 1);
>> Bu =  131.9999  411.7251  445.2751
Buvertex1=Bu;
```

**checks with reverse order for vertex 1**

```
GU=zeros(6, 6, 64);
MSWMFU=zeros(6,1,64);
for n=1:64   GU(:,:,n)=[Gunb{1,n} ; Gunb{2,n} ; Gunb{3,n}];
MSWMFU(:,:,n)=[MSWMFunb{1,n}; MSWMFunb{2,n}  ; MSWMFunb{3,n}];  end

>> GU(:,:,23) =    % 6 x 6
  1.0000 + 0.0000i   0.0920 - 0.3464i   8.7367 + 1.1630i  10.8139 -15.1709i  -0.6748 - 4.2623i   1.9688 + 0.6639i
  0.0000 + 0.0000i   1.0000 + 0.0000i   3.1234 +24.3936i  48.6591 +18.2925i  11.0106 - 4.8735i  -0.3797 + 5.7850i
  0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   0.9891 - 1.8681i  -1.3553 - 0.3648i   0.4582 - 0.4967i
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i  -0.1475 - 0.6474i   0.3091 + 0.0816i
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i  -0.2233 + 0.4266i
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i

>> MSWMFU(:,:,23) =        % 6 x 1
  0.8840 - 0.3319i
  1.5285 + 2.1461i
  0.0553 - 0.0808i
  0.0460 + 0.0052i
  0.0535 - 0.0801i
 -0.1988 - 0.0213i
```

**Note the mu_bc match is not quite perfect on the 99% point**

```
>> A(:,:,23) =
  0.0805 - 0.0000i   0.0074 - 0.0279i   0.2114 + 0.0000i   0.2091 - 0.3950i   0.9368 - 0.0000i  -0.2092 + 0.3996i
  0.0074 + 0.0279i   0.0103 - 0.0000i   0.2091 + 0.3950i   0.9447 + 0.0000i  -0.2092 - 0.3996i   0.2172 - 0.0000i
```

- minPMAC for large N, U tends to have numerical issues
  - CVX version (minPMACMIMO) runs much longer, can tend to be more accurate

# check other vertex

- Small error in 99% on vertex share rate can require large compensation on the 1% rate

```
H64mac =reshape(H64,2,1,3,64);
H64mac=H64mac(:,:,[2 1 3],:);  % set order for other vertex
H64bc=zeros(1,2,3,64);
for n=1:64
for i=1:3   H64bc(:,:,i,n) = H64mac(:,:,4-i,n)'; end
end
H64bc1=permute(reshape(H64bc, [2 , 3, 64]), [ 2 1 3]);

Rxxm=zeros(1,1,3,64);
E=cell2mat(info.Eun);

for n=1:64
Rxxm(1,1,:,n)=E(2,[2 1 3],n);    end
Rxxb=zeros(2,2,3,64);
bbc=zeros(3,64);
for n=1:64   Rxxb(:,:,:,n)=mac2bc(Rxxm(:,:,:,n), H64mac(:,:,:,n)); end

A=zeros(2, 6, 64);
for n=1:64  A(:,:,n)=[ sqrtm(Rxxb(:,:,1,n)) , sqrtm(Rxxb(:,:,2,n)),  sqrtm(Rxxb(:,:,3,n)) ];
end

[Bu, Gunb, S0, MSWMFunb, B] = mu_bc(H64bc1, A, [1 1 1], 1);

>> Bu =  131.9999  416.7071  440.2931
>> rate = [Buvertex1 ; Bu]
  131.9999  411.7251  445.2751
  131.9999  416.7071  440.2931
>> info.frac'*rate  =
  131.9999  411.7730  445.2270   versus 412 and 445
```

```
Check with:
for n=1:64
bbc(1,n)=real(log2(1+H64bc(:,:,1,n)*Rxxb(:,:,1,n)*H64bc(:,:,1,n)'));
bbc(2,n)=real(log2((1+H64bc(:,:,2,n)*(Rxxb(:,:,2,n)+Rxxb(:,:,1,n))*H64bc(:,:,2,n)')/(1+H64bc(:
,:,2,n)*Rxxb(:,:,1,n)*H64bc(:,:,2,n)')));
bbc(3,n)=real(log2((1+H64bc(:,:,3,n)*(Rxxb(:,:,3,n)+Rxxb(:,:,2,n)+Rxxb(:,:,1,n))*H64bc(:,:,3,n)
')/(1+H64bc(:,:,3,n)*(Rxxb(:,:,2,n)+Rxxb(:,:,1,n))*H64bc(:,:,3,n)')));
end
bvec=sum(bbc') =  131.9999  411.7251  445.2751
bsum=sum(bvec) =  989.0000

>> newfrac=inv(rate(1:2,2:3)')*[412 ; 445]  =
   0.9448
   0.0552
```

**Better design is**
**19 symbols mac2bc vertex 1**
**1 symbol mac2bc vertex 2**

```
> E(:,:,21:25)
   1.1044   1.1722   0.4032
   1.1044   1.1722   0.4032

   1.0237   1.1371   0.5177
   1.0237   1.1371   0.5177

   0.9394   1.1115   0.6258
   0.9394   1.1115   0.6258

   0.8528   1.0964   0.7249
   0.8528   1.0964   0.7249

   0.7653   1.0920   0.8132
   0.7653   1.0920   0.8132
```

**Energies from minPMAC**
**But then used in mac2bc**

**Note equal energies**
**both vertices' of each tone**

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

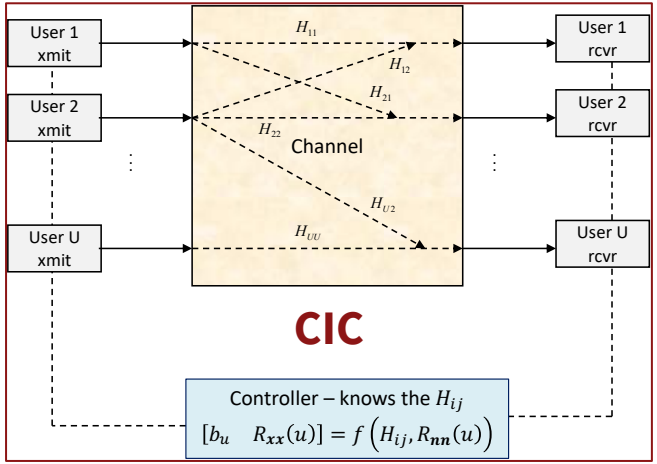
- Check any point's admissibility on the dual MAC with admMAC

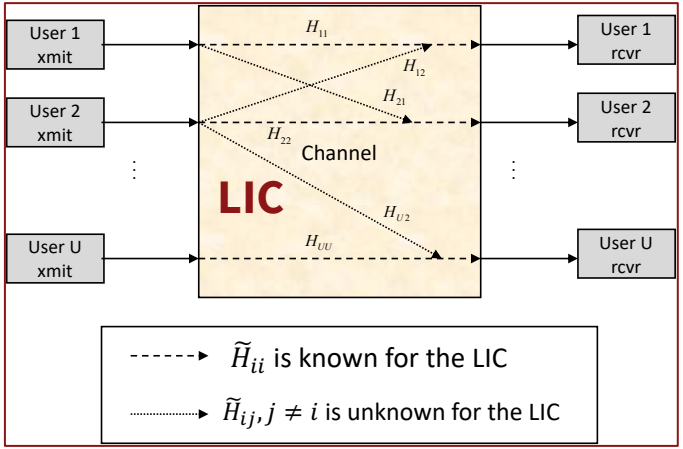
# IC Review

# Central or Local Control?

- **Centrally controlled IC (CIC):** directs users
  - Centrally sets $b_{u,l,n}$ , $\mathcal{E}_{u,l,n}$ , $R_{\boldsymbol{xx}}(u,n)$ , users' codes
  - Knows $H_{u\leftrightarrow u',l,n}$ , $R_{\boldsymbol{nn}}(u,n)\ \forall\ u \in \boldsymbol{u}$
    - Distributed Antenna Systems (DAS)
    - Cell-free with mobile-edge computation
    - Often associated with licensed spectra
  - All users may use successive decoding (GDFE) at receivers for a locally imposed order
  - Basically, the IC you know already

- **Locally controlled IC (LIC):** each user transmitter can only control its own transmission
  - Locally (transceivers $u$) sets $b_{u,l,n}$ , $\mathcal{E}_{u,l,n}$ , $R_{\boldsymbol{xx}}(u,n)$, codes
  - Knows only $H_{u\leftrightarrow u,l,n}$ , $R_{\boldsymbol{nn}}(u,n)$
    - Collision detection, avoidance
    - Unlicensed spectra
  - Everyone else is noise (try to detect and remove them at your own risk)



# Review MU Capacity step: Prior-User Set

- Order vector and inverse
  - Permutation (permutation matrix J), etc

$$\boldsymbol{\pi}_u = \begin{bmatrix} \pi(U') \\ \vdots \\ \pi(1) \end{bmatrix} \quad \boldsymbol{\pi}_u^{-1} = \begin{bmatrix} U' \\ \vdots \\ 1 \end{bmatrix} \qquad j = \pi(i) \rightarrow i = \pi^{-1}(j)$$

- Prior-User Set is $\mathbb{P}_u(\boldsymbol{\pi}) = \{j \mid \boldsymbol{\pi}^{-1}(j) < \boldsymbol{\pi}^{-1}(u)\}$
  - That is "all the users before the desired user $u$ in the given order $\boldsymbol{\pi}$.
  - Receiver $u$ best decodes these "prior" users and removes them, while "post" users are noise
  - $\boldsymbol{\pi}$ can be any order in $\mathbb{P}_u(\boldsymbol{\pi})$, but the most interesting is usually $\boldsymbol{\pi}_u$ (receiver $u$'s order)

| rcvr/ User $i$ | $\pi_4(i)$ | $\pi_3(i)$ | $\pi_2(i)$ | $\pi_1(i)$ |
|---|---|---|---|---|
| $i=4$ | 3 | 3 | 4 | 3 |
| $i=3$ | 4 | 2 | 3 | 2 |
| $i=2$ | 1 | 4 | 2 | 1 |
| $i=1$ | 2 | 1 | 1 | 4 |
| $\mathbb{P}_u(\boldsymbol{\pi}_u)$ | {1,2} | {2,4,1} | {1} | {4} |

$$\boldsymbol{\Pi} = \begin{bmatrix} 3 & 3 & 4 & 3 \\ 4 & 2 & 3 & 2 \\ 1 & 4 & 2 & 1 \\ 2 & 1 & 1 & 4 \end{bmatrix}$$

**Assumes $U' = 4$, so no subusers**

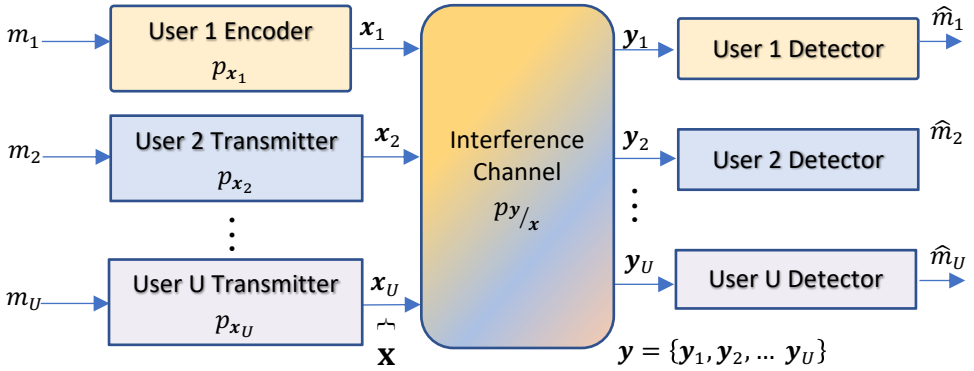
**Generally could have $(16)^4$ orders**

- Data rates (mutual information bounds really on those users who are cancelled, not noise)

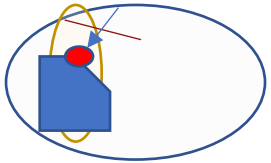
| $\mathfrak{I}$ | $\mathfrak{I}_4$ | $\mathfrak{I}_3$ | $\mathfrak{I}_2$ | $\mathfrak{I}_1$ |
|---|---|---|---|---|
| top | $\infty$ | $\mathrm{I}_3(3/1,2,4)$ 20 | $\infty$ | $\infty$ |
| | $\mathrm{I}_4(4/1,2)$ 10 | $\mathrm{I}_3(2/1,4)$ 9 | $\infty$ | $\infty$ |
| | $\mathrm{I}_4(1/2)$ 5 | $\mathrm{I}_3(4/1)$ 4 | $\mathrm{I}_2(2/1)$ 4 | $\mathrm{I}_1(1/4)$ 2 |
| bottom | $\mathrm{I}_4(2)$ 1 | $\mathrm{I}(1)$ 2 | $\mathrm{I}_2(1)$ 2 | $\mathrm{I}_1(4)$ 5 |

$$\mathrm{I}_{min}(\boldsymbol{\Pi}, p_{xy}) = \begin{bmatrix} 4 \\ 20 \\ 1 \\ 2 \end{bmatrix}$$

# The Traditional IC



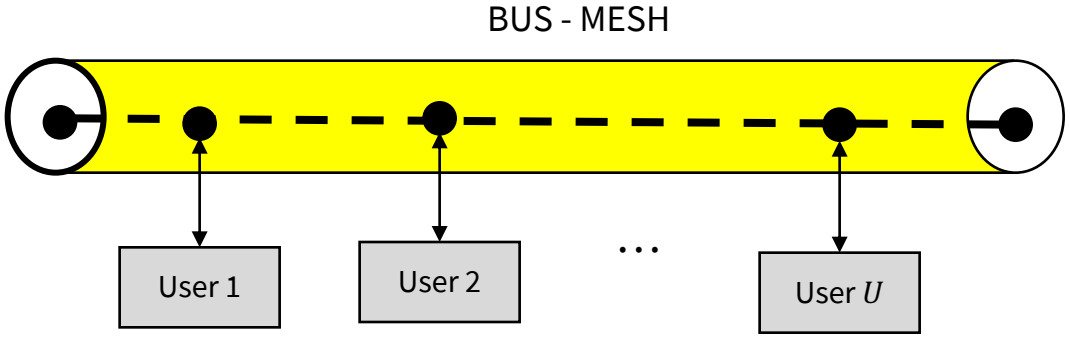

- Studied earlier and found an order-indexed set of vertices $\mathcal{I}_{min}(\boldsymbol{\Pi}, p_{\boldsymbol{xy}})$
- Then take convex combinations over the possible $(U^2!)^U$ orders of the $\mathcal{I}_{min}(\boldsymbol{\Pi}, p_{\boldsymbol{xy}})$



The achievable-region remains convex
But laGrangian $\boldsymbol{\theta}$ now has $U^2$ entries

# Another Interference Channel Form

- The BUS model




- All connections are bi-directional and the transmitter/receiver are independent; $U' = U \cdot (U-1)$
  - This is actually $U,(U-1)$-user MACs with $U,(U-1)$-user BCs
- Restrictions to IC might include
  - only $U$ messages permitted (e.g., a "switch") – there is a transmit/receive pair for each
  - In any given symbol period → time-varying IC
- The "yellow" might be a common wire(s) or air (common shared spectrum)
- Designers have headed consequently towards "ODM" (orthogonal division multiplex – all get their own)
  - As with primary-user concept

# CIC Optimization

# “Optimum” Spectrum Balancing (no GDFEs)

- Minimize weighted energy sum for given $\boldsymbol{b}_{min}$.
  - No crosstalk cancellation

$$\max_{\{R_{\boldsymbol{xx}}(u,n)\}} \quad \sum_{u=1}^{U} w_u \cdot \mathcal{E}_u$$
$$ST: \quad 0 \le \sum_{n} \text{trace}\{R_{\boldsymbol{xx}}(u,n)\} \le \mathcal{E}_{u,max} \; u = 1, \ldots, U$$

- Relate $\boldsymbol{b}$ to $R_{\boldsymbol{xx}}(u,n)$.
  - No crosstalk cancellation

$$b_u = \sum_{n} \log_2 \frac{| H_{uu,n} \cdot R_{\boldsymbol{xx}}(u,n) \cdot H^*_{uu,n} + \mathcal{R}_{noise}(u,n) |}{| \mathcal{R}_{noise}(u,n) |}$$

- $R_{noise}(u,n) = \mathrm{I} + \sum_{i \neq u} \tilde{H}_{u,i,n} \cdot R_{\boldsymbol{xx}}(i,n) \cdot \tilde{H}^*_{u,i,n}$.
  - All other users are noise additions

- Tonal Lagrangian
  - Can minimize each individually, and sum
  - It’s not convex (no sequential differences transform)

$$L_n(R_{\boldsymbol{xx}}(u,n), \boldsymbol{b}_n, \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{u=1}^{U} w_u \cdot \mathcal{E}_{u,n} - \theta_u \cdot b_{u,n}$$

# Still has minimum, exponential search

- $SNR(u,n) = \frac{|\tilde{H}_{u,u,n} \cdot R_{xx}(u,n) \cdot \tilde{H}^*_{u,u,n}|}{|R_{noise}(u,n)|}$

$$b_u = \sum_n \log_2 \left(1 + \frac{\mathrm{SNR}(u,n)}{\Gamma}\right)$$

- Partition energy range for scalar case

$$M = \frac{\max_u \mathcal{E}(u)}{\Delta\mathcal{E}}$$

- **Rxx step:** For each tone, search $M^U$ possible energies to minimize tonal Lagrangian and add them
- **Constraint:** Use sub-gradient or ellipsoid descent method to update the $\boldsymbol{\theta}$ Lagrange multiplier for rate constraints

$$\begin{aligned} \Delta \boldsymbol{b} &= \boldsymbol{b}_{min} - \sum_n \boldsymbol{b}_n \\ \Delta \boldsymbol{\mathcal{E}} &= \boldsymbol{\mathcal{E}}_{max} - \sum_n \boldsymbol{\mathcal{E}}_n \end{aligned}$$

$$\begin{aligned} \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \epsilon \cdot \Delta \boldsymbol{b} \\ \boldsymbol{w} &\leftarrow \boldsymbol{w} + \epsilon' \cdot \Delta \boldsymbol{\mathcal{E}} \end{aligned}$$

$\boldsymbol{w}$ update only for admissibility search

# OSB.m

- The OSB search can be very complex for U>3
- It also can have severe numerical issues (cause it to diverge) even in matlab double prec.
- A GDFE-based (xtalk cancelling) algorithm could also exponentially search
  - And would need to search all possible orders – exhorbitantly complex
- There are LIC (or nearly LIC) approximations to it that are very good and avoid these problems
  - Will return to them later
- And by the way, the real optimum problem is convex (unlike OSB)

Program lost – potential student project

# minPIC = more "optimum"

- minPIC concept allows for each receiver $u$ to cancel $i \in \mathcal{D}_u(\boldsymbol{\Pi}, p_{\boldsymbol{xy}}, \boldsymbol{b})$ ; the decodable set
- Order has been restored 😀
- The optimization is

$$\min_{\{R_{\boldsymbol{xx}}(u)\}} \quad \sum_{u=1}^{U} w_u \cdot \text{trace}\underbrace{\{R_{\boldsymbol{xx}}(u)\}}_{\mathcal{E}_u}$$

$$ST: \quad b_{i,u} \geq \left\{ \begin{array}{ll} b_{min,i} & \pi_u(i) \leq \pi_u(u) \\ 0 & \pi_u(i) > \pi_u(u) \end{array} \right\} \triangleq b_{min(\pi_u),u,i}$$

$$R_{\boldsymbol{xx}}(u) \succeq \mathbf{0} \quad .$$

- $\boldsymbol{\theta} \rightarrow \boldsymbol{\Theta}$ , which now has $U^2$ terms, $U$ for each receiver → determines the $\boldsymbol{\Pi}$
- This is actually convex already (like minPMAC)
- Implement GDFE at each receiver for final order ($\boldsymbol{\Theta} \rightarrow \boldsymbol{\Pi}$)

# minPIC = more "optimum"

- Lagrangian now sums IC receivers

$$L(R_{\boldsymbol{XX}}(u), \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{\Theta}) = \sum_{u=1}^{U} \left( w_u \cdot \left[ \sum_{n=0}^{\overline{N}} \text{trace}\{R_{\boldsymbol{XX}}(u,n)\} \right] - \sum_{i=1}^{U} \left[ \theta_{u,i} \cdot \left\{ \left[ \sum_{n=0}^{\overline{N}-1} b_{u,i,n} \right] - b_{min(\pi_u),u,i} \right\} \right] \right) \quad \forall\, u \in \boldsymbol{U}$$

- Tonal Lagrangian

$$L_n(R_{\boldsymbol{XX}}(u,n), b_{u,i,n}, \boldsymbol{w}, \boldsymbol{\Theta}) = \sum_{u=1}^{U} \left( w_u \cdot [\text{trace}\{R_{\boldsymbol{XX}}(u,n)\}] - \sum_{i=1}^{U} [\theta_{u,i} \cdot b_{u,i,n}] \right) \quad \forall\, u \in \boldsymbol{U}$$

- There are thus then $U$ gradient/Newton's method descent terms added in the Rxx step
- The $\boldsymbol{\Theta}$ update is like MAC, except there are now $U^2$ values to update, using $b_{min(\pi_u),u,i}$
- For any $\boldsymbol{\Theta}$, the order on Rxx step is known, and $i \notin \mathcal{D}_u(\boldsymbol{\Pi}, p_{\boldsymbol{xy}}, \boldsymbol{b})$ eventually should zero influence of Hessians/gradients – use small step size

# minPIC

- Awaits motivated student project
  - This one is an A+
- Some issues noted with convergence as zeroed theta and bits/user-rcvr going to zero puts them at bottom of derived order at each step
- The capacity region is convex, so the relationship of Rxx to b by “sharing” (vertex, time, …) ensures this desirable property


# End Lecture 17