STANFORD

Lecture 16

Optimal Interference Channel Design

May 29, 2026

JOHN M. CIOFFI

Hitachi Professor Emeritus of Engineering

Instructor EE379B – Spring 2026

Announcements & Agenda

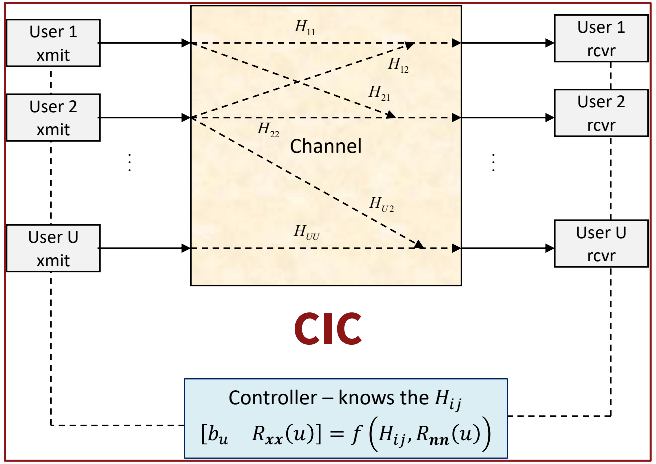
- Announcements
  - hihkjh

- Agenda
  - IC Review
  - minPIC
  - Approximate (no GDFE) methods

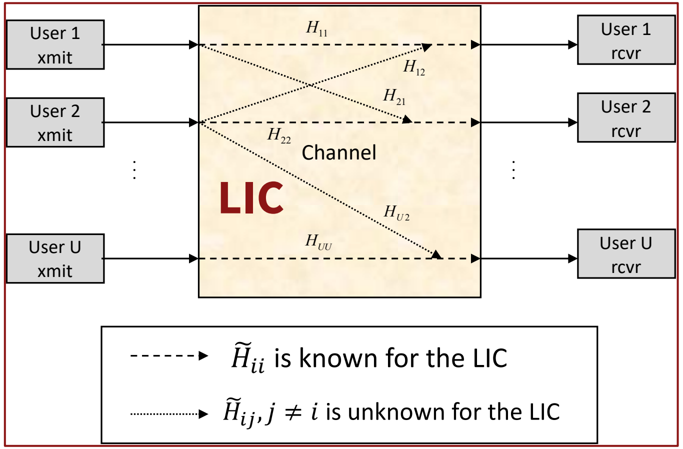
IC Review

Central or Local Control?

- **Centrally controlled IC (CIC):** controller directs users, &
  - centrally sets $b_{u,l,n}$, $\mathcal{E}_{u,l,n}$, $R_{\boldsymbol{xx}}(u,n)$, users' codes,
  - knows $H_{u\leftrightarrow u',l,n}$, $R_{\boldsymbol{nn}}(u,n)$ $\forall\ u \in \boldsymbol{u}$.
  - Examples include:
    - Distributed Antenna Systems (DAS)
    - Cell-free with mobile-edge computation &
    - CIC is often associated with licensed spectra (e.g., cellular).
  - User receivers use successive decoding (GDFE).
  - Basically, this is the IC you know already.

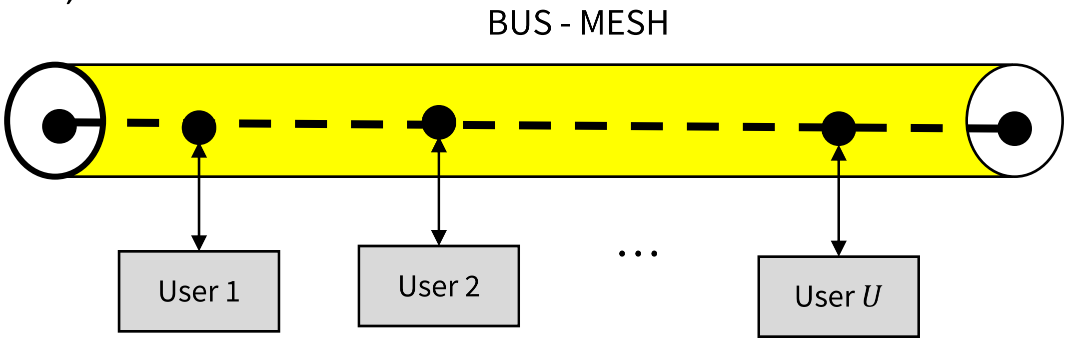


- **Locally controlled IC (LIC):** each user transmitter can only control its own transmission.
  - Locally (transceiver u) sets $b_{u,l,n}$, $\mathcal{E}_{u,l,n}$, $R_{\boldsymbol{xx}}(u,n)$, & codes.
  - Local user knows only $H_{u\leftrightarrow u,l,n}$, $\mathcal{R}_{\boldsymbol{nn}}(u,n)$.
  - Examples include:
    - Collision detection, avoidance are popular.
    - Unlicensed spectra (Wi-Fi, Bluetooth).
  - Everyone else is noise (try to detect and remove them at your own risk).



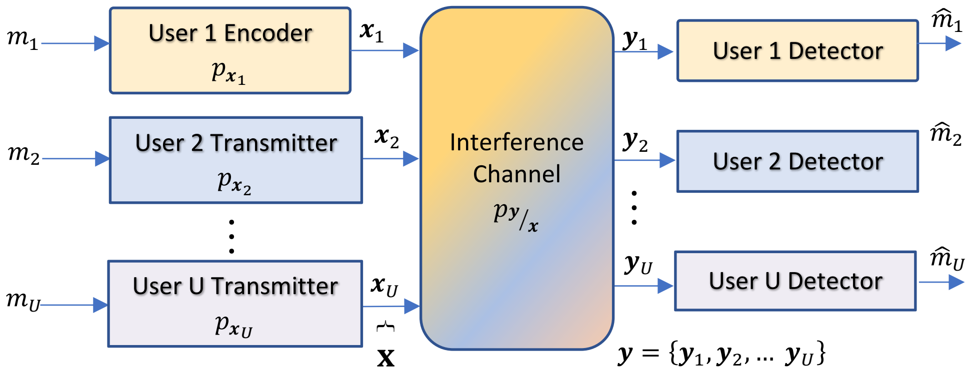
Another Interference Channel Form

- The BUS model;

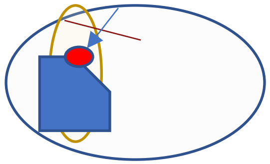


- All connections are bi-directional and the transmitter/receiver are independent; $U' = U \cdot (U-1)$.
  - This is actually $U \cdot \ (U-1)$-user MACs with $U \cdot (U-1)$-user BCs. And then, we can consider subusers ….
- Restrictions to IC might include:
  - Only U messages are permitted (e.g., a "switch"), and 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).
- Designs, so far, are "ODM" (**orthogonal division multiplex** – all users occupy mutually separate dimensions.)

The CIC



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



The achievable-region remains convex, and the Lagrangian $\boldsymbol{\theta}$ is equivalent to order.

minPIC

minPIC = more "optimum"

- minPIC receiver u cancels all subusers $i \in \mathcal{D}_u(\boldsymbol{\Pi}, p_{\boldsymbol{xy}}, \boldsymbol{b})$; the decodable set.
- Order has been restored 😀.
- The optimization is
  - $(i,u) = (RCVR, USER)$

$$\min_{\{R_{\boldsymbol{xx}}(i,u,n)\}} \sum_{n=0}^{\overline{N}-1} \sum_{u=1}^{U} w_u \cdot \underbrace{\text{trace}\left\{\sum_{i=1}^{U} R_{\boldsymbol{xx}}(i,u,n)\right\}}_{\mathcal{E}_{u,n}}$$

$$ST: \quad b_u \geq b_{min,u}$$
$$R_{\boldsymbol{xx}}(i,u,n) \succeq \mathbf{0} \quad .$$

- $\boldsymbol{\theta}$ again has U terms that determine the "**feasible**" U^2-dimensional orders $\boldsymbol{\Pi}$.
- Feasible orders retain a convex achievable-region constraint (like minPMAC).
- Implement GDFE at each receiver (no transmitter nonlinear precoders).

Review MU Capacity step: Prior-User Set

- Order vector and inverse, or
  - 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 $\mathbb{P}_u(\boldsymbol{\pi}) = \{j \mid \boldsymbol{\pi}_u^{-1}(j) < \boldsymbol{\pi}_u^{-1}(u)\}$.
  - That is "all the subusers before the desired user u" in the given order $\boldsymbol{\pi}_u$ at receiver u.
  - Receiver u best decodes these "prior" users and removes them, while "post" users are noise.
  - $\boldsymbol{\pi}_u$ can be any order in $\mathbb{P}_u(\boldsymbol{\pi}_u)$ that is chosen in design.

rcvr/ User i	$\boldsymbol{\pi}_4(i)$	$\boldsymbol{\pi}_3(i)$	$\boldsymbol{\pi}_2(i)$	$\boldsymbol{\pi}_1(i)$
$i=4$	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) depend on those user rates that are decoded/cancelled; these energies are equivalent to a good-code choice and consequent rate(s).

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

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

3-User Order example

- Given a $\boldsymbol{\theta}$, say for example with $\theta_3 > \theta_1 > \theta_2$, they determine all receivers' order:

$FOR: \quad \theta_3 \quad > \quad \theta_1 \quad > \quad \theta_2 \quad > 0$

(RCVR, USER)	Receiver 3	Receiver 1	Receiver 2
	(1,1),(2,1),(1,2),(2,2)	(2,2),(3,2),(2,3),(3,3)	(1,1),(3,1),(1,3),(3,3)
	(3,3)	(1,3)	(2,3)
	(1,3)	**(3,1)**	(2,1)
	(2,3)	**(1,1)**	**(3,2)**
	(3,1)	**(2,1)**	**(1,2)**
	(3,2)	(1,2)	**(2,2)**

THESE ARE constant XTALK, AND CAN BE 0 (first row)

- Any other order is inconsistent with the Lagrangian multipliers' interpretation and capacity/achievable region's convexity

$$
\begin{aligned}
A &\triangleq |H_{3,1}|^2 \cdot (\mathcal{E}_{1,1} + \mathcal{E}_{2,1}) + |H_{3,2}|^2 \cdot (\mathcal{E}_{1,2} + \mathcal{E}_{2,2}) + I \\
B &\triangleq |H_{3,3}|^2 \cdot \mathcal{E}_3 + A \\
C &\triangleq |H_{3,1}|^2 \cdot \mathcal{E}_{3,1} + B \\
D &\triangleq |H_{3,2}|^2 \cdot \mathcal{E}_{3,2} + C
\end{aligned}
$$

RCVR 3

$$
\begin{aligned}
b_3 &= \log_2(B) - \log_2(A) \\
b_{3,1} &= \log_2(C) - \log_2(B) \\
b_{3,2} &= \log_2(D) - \log_2(C)
\end{aligned}
$$

$\boldsymbol{\Pi}$

$$
\left\{ \sum_{u=1}^{3} \theta_u \cdot b_u \right\}_{RCVR3opt} = (\theta_3 - \theta_1) \cdot \log_2(B) + (\theta_1 - \theta_2) \cdot \log_2(C) + \theta_2 \cdot \log_2(D)
$$

Do same for other 2 receivers

- RCVR 1 optimization of rate sum

$$\begin{aligned} A &\triangleq |H_{1,3}|^2 \cdot (\mathcal{E}_{2,3} + \mathcal{E}_{3,3}) + |H_{1,2}|^2 \cdot (\mathcal{E}_{1,2} + \mathcal{E}_{2,2}) + I \\ B &\triangleq |H_{1,3}|^2 \cdot \mathcal{E}_{1,3} + A \\ C &\triangleq |H_{1,1}|^2 \cdot \mathcal{E}_1 + B \\ D &\triangleq |H_{1,2}|^2 \cdot \mathcal{E}_{1,2} + C \end{aligned}$$

RCVR 1

$$\begin{aligned} b_{1,3} &= \log_2(B) - \log_2(A) \\ b_1 &= \log_2(C) - \log_2(B) \\ b_{1,2} &= \log_2(D) - \log_2(C) \quad . \end{aligned}$$

$$\left\{\sum_{u=1}^{3} \theta_u \cdot b_u\right\}_{RCVR1opt} = (\theta_3 - \theta_1) \cdot \log_2(B) + (\theta_1 - \theta_2) \cdot \log_2(C) + \theta_2 \cdot \log_2(D)$$

- RCVR 2 optimization of rate sum

$$\begin{aligned} A &\triangleq |H_{2,3}|^2 \cdot (\mathcal{E}_{1,3} + \mathcal{E}_{3,3}) + |H_{2,1}|^2 \cdot (\mathcal{E}_{1,1} + \mathcal{E}_{3,1}) + I \\ B &\triangleq |H_{2,3}|^2 \cdot \mathcal{E}_{2,3} + A \\ C &\triangleq |H_{2,1}|^2 \cdot \mathcal{E}_{2,1} + B \\ D &\triangleq |H_{1,1}|^2 \cdot \mathcal{E}_2 + C \end{aligned}$$

RCVR 2

$$\begin{aligned} b_{2,3} &= \log_2(B) - \log_2(A) \\ b_{2,1} &= \log_2(C) - \log_2(B) \\ b_2 &= \log_2(D) - \log_2(C) \quad . \end{aligned}$$

$$\left\{\sum_{u=1}^{3} \theta_u \cdot b_u\right\}_{RCVR2opt} = (\theta_3 - \theta_1) \cdot \log_2(B) + (\theta_1 - \theta_2) \cdot \log_2(C) + \theta_2 \cdot \log_2(D)$$

- Six energies repeat – select the smallest that has corresponding lowest rate for its transmitter.
- Outer $\boldsymbol{\theta}$ loop (e.g., Ellipsoid) remains the same as minPMAC.

Generalize – first order them to simplify

- Create order of users for each of (reordered) users

θ_U	$\dots$	θ_u	$\dots$	θ_1
$\boldsymbol{U}^2 \setminus \{(1:U,U),(U,1:U-1)\}$	$\dots$	$\boldsymbol{U}^2 \setminus \{(1:U,u),(u,1:U-1)\}$	$\dots$	$\boldsymbol{U}^2 \setminus \{(U,1:U),(1,1:U-1)\}$
(U,U)	$\dots$	(u,U)	$\dots$	1,U-1
$\vdots$		$\vdots$		$\vdots$
	$\dots$	$(u,U-u+1)$	$\dots$	$(1,1)$
$(1,U)$	$\dots$	(U,u)	$\dots$	$(U,1)$
$(U,U-1)$	$\dots$	$\vdots$	$\dots$	$\vdots$
$\vdots$	$\dots$	$(1,u)$	$\dots$	$\vdots$
		$(u,U-u-1)$		
$\vdots$	$\dots$	$\vdots$	$\dots$	$\vdots$
$(U,1)$	$\dots$	$(u,1)$	$\dots$	(U,U)

Table 5.2: **Generalized of overall decoding order pairs given descending-order θ.**

Generalize A,B,C, D

$$
\begin{array}{lll}
K_{1,u} & \triangleq \sum_{i \neq u} H_{u,i} \cdot \left(\sum_{j \neq i} R_{\boldsymbol{xx}}(i,j) \right) \cdot H^*_{u,i} + I & \text{for } \; b_{u,U} \\
K_{2,u} & \triangleq H_u(2) \cdot R_{\boldsymbol{xx}}(u, 2U-3, 2) \cdot H^*_u(2^{nd}) + K_{1,u} & \text{for } \; b_{u,U-} \\
\vdots & \vdots & \\
K_{u,u} & \triangleq H_{u,2U-u+1}(2) \cdot R_{\boldsymbol{xx}}(u, 2U-u+1(2^{nd})) \cdot H^*_{u,2U-u+1}(2) + K_{u-1,u} & \text{for } \; b_u \\
\vdots & \vdots & \\
K_{2U-2,u} & \triangleq H_{u,1}(2) \cdot R_{\boldsymbol{xx}}(u, 1(2^{nd})) \cdot H^*_{u,1}(2) + K_{2U-3,u} & \text{for } \; b_{u,1}
\end{array}
$$

$$
\left\{ \sum_{u=1}^{U} \theta_u \cdot b_u \right\}_{RCV\,Buopt} = (\theta_U - \theta_{U-1}) \cdot \log_2(K_{1,u}) + \ldots + (\theta_2 - \theta_1) \cdot \log_2(K_{2U-3,u}) + \theta_2 \cdot \log_2(K_{2U-2,u})
$$

- This is convex in those quantities optimized
- Need the outer subgradient loop on theta to drive IC rate vector to bmin.

All done tacitly in minPIC

minPIC Examples

Approximate (no GDFE) Methods

CIC's "Optimum" Spectrum Balancing (no GDFEs)

- **OSB** minimizes weighted energy sum for given $\boldsymbol{b}_{min}$.
  - There is no crosstalk cancellation

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

- OSB relates $\boldsymbol{b}$ to $R_{\boldsymbol{xx}}(u,n)$.

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

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

- Tonal Lagrangian
  - Minimize each individually, and sum.
  - It's not convex (no sequential-differences transformation).

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

Still has minimum, exponential search

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

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

- Partition energy range for scalar case.

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

- **Energy step:** For each tone, search M^U possible energies to minimize tonal Lagrangian and add these tonals.
  - Typically have power-spectrum and/or bit-cap constraint on the tone to limit tonal energy
  - And then a constraint check on energy, summing over all tones
- **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 is only for admissibility search.

OSB.m

```
function [S1, S2, b1, b2] = osb(Hmag_sq, No, E, theta, mask, ...
  gap, bitcap, cb)

  osb and also finds w1 energy weight for USER 1
  A. Chowdhery ~2010 ; Updated by J. Cioffi in 2024.  It presently
handles
  only 2 users, so U=2.

  Inputs

  Hmag_sq is a N x 2 x 2 where N is FFT size. N inferred from this.
  No      is a 1 x U white-noise power spectra density matrix.
     If Hmag_sq is complex BB, then No should be the one-sided PSD.
  E       is a 1 x U energy vector.
  theta   is a 1 x U user-rate weighting vector.
  mask    is an N x U PSD maximum allowed.
  gap     is the (non-dB) linear gap (so 1 if 0 dB gap).
  bitcap  is a 1 x U maximum number of bits allowed per tone.
  cb      is 2 for real baseband and 1 for cplex bband

  Outputs

  S1      is user 1's Nx1 PSD
  S2      is user 2's Nx1 PSD
  b1      is user 1's Nx1 bit distribution
  b2      is user 2's Nx1 bit distribution

calls optimize_l2.m, which calls optimize_s.m
User order is reversed with respect to class convention.
```

Only tests integer bits/tone up to bitcap
Discards solutions that exceed user energy

```
>> H2
H2(:,:,1) =   0.6400   0.2500 % note this is squared mag each term
H2(:,:,2) =   0.4900   0.3600
>> Noise  =  1.0e-04 * [   1.0000   1.0000];
>> Ex =[    1    1];
>> mask =[    1    1];
>> gap =    1;
>> bitcap =[   15   15];
>> [S1, S2, b1, b2] = osb(H2, Noise, Ex, [0.5 .5], mask, gap, bitcap,2)

S1 =   0.6398
S2 =    0
b1 =    6 % note < 6.3 for the GDFE based IC's maximum L11:16
b2 =    0
>> [S1, S2, b1, b2] = osb(H2, Noise, Ex, [0.01 .99], mask, gap, bitcap,2)
S1 =    0
S2 =   0.1419
b1 =    0
b2 =    4.5000 <5.9 for L11:16
```

- 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.

Multitone OSB

```
h = cat(3, [1   .8 ;  -1    1], [-.9   -.7 ;  0   1]  )*10;
He = fft(h, 8, 3);
>> H3=zeros(8,2,2);
>> H3(:,1,1)=He(1,1,:);
>> H3(:,2,1)=He(2,1,:);
>> H3(:,2,2)=He(2,2,:);
>> H3(:,1,2)=He(1,2,:);
>> Noise=ones(8,2);
>> mask=ones(8,2);
>> Ex=8*Ex;
>> [S1, S2, b1, b2] = osb(H3.*conj(H3), Noise, Ex, [0.5 .5], mask, gap,
bitcap,1);
>> S1' =     0          0        0.7017    0.8272    0    0.8272    0.7017       0
>> S2' =   0.6375    0.7469       0        0          0       0          0          0.7469

>> b1' =    0    0    7    8    0    8    7    0
>> b2' =    8    8    0    0    0    0    0    8
>> sum(b1) =   30
>> sum(b2) =   24
sum(b1+b2) =   54 % < ~116 that MAC, BC, single had for this channel
```

Same 2x2 channel As in L16.

OSB solution is often FDM

- GDFE's cancellation of crosstalk makes a large difference

IW_polite.m (integer bits like osb.m)

```
% function [b, E] = iw_polite(N, df, U, Hmag, No, Ex, mask, gap, mode,
b_target, bitcap,cb)
%
% Calculates data rates of M users and corresponding bit distributions and
Energy distributions
% using iterative waterfilling
%
% Inputs
% -------
% N: number of sub-channels
% M: number of users
% Hmag: squared channel transfer and crosstalk matrix (N x U x U matrix)
% Hmag(n,i,j) is the crosstalk transfer function from loop i to j at the
nth bin.
% No: noise energy/sample
% Ex: signal energy/SYMBOL
% mask: PSD mask - largest value N x U
% gap: gap (not in dB)
% mode: (U x 1 vector) each value is one of the followings
% 0 - rate adaptive
% 1 - fixed margin (power minimization)
% 2 - margin adaptive
% b_target: target bits on 1 DMT symbol for modes 1 and 2
% bitcap: maximum possible number of bits at each frequency bin
% cb =1 for cplx BB and =2 for real BB
%
% Outputs
% --------
% b: bit distribution (N x U matrix)
% E: energy distribution (N x U matrix)
%
% Remarks
% Iterate waterfiling for each user 10 times
% Youngjae(Sean) Kim - modified J. Cioffi, April 2024
```

```
>> [b, E] = iw_polite(8, 2, H3.*conj(H3), Noise, [8 8],
mask, gap, zeros(8,1), [5 5], bitcap,1)
b =
        0    8.0000
        0    8.0000
   2.0264    1.0000
   8.0000         0
   8.0000         0
   8.0000         0
   2.0264    1.0000
        0    8.0000

E =
        0    0.6375
        0    0.7469
   0.6300    0.3609
   0.8272         0
   0.7064         0
   0.8272         0
   0.6300    0.3609
        0    0.7469
>> sum(b) =  28.0529   26.0000
>> sum([b1 b2]) =  30   24
```

```
>> [b1 b2] %(osb)
   0   8
   0   8
   7   0
   8   0
   0   0
   8   0
   7   0
   0   8
```

```
>> [S1 S2] = %(osb)
        0    0.6375
        0    0.7469
   0.7017         0
   0.8272         0
        0         0
   0.8272         0
   0.7017         0
        0    0.7469
```

sum(S1)
= 3.0577
sum(S2)
= 2.1313

sum(E)
= 3.6207
2.8531

- Sum is same, user 1 is better in osb.
- With continuous bit distribution, osb would be slightly better.

Energies < 8 because iw calls campello.m, which allows only integer bits (like osb.m).

IW.m (non-integer) – not in text, but at website

```
function function [b, E] = iw(N, U, Hmag, No, Ex, gap, mode, b_target,cb)

  Calculates data rates of M users and corresponding bit distributions and
Energy distributions
     using iterative waterfilling.

  Inputs
  ------
  N: number of sub-channels
  U: number of users
  Hmag: squared channel transfer and crosstalk matrix (N x U x U matrix)
         Hmag(n,i,j) is the crosstalk transfer function from loop i to j
at the nth bin.
  No: noise power spectrum per tone (N x U)
  Ex: signal energy/SYMBOL
  mask: PSD mask - largest value N x U
  gap: gap in dB
  mode: (U x 1 vector) each value is one of the followings
        0 - rate adaptive
        1 - fixed margin (power minimization)
        2 - margin adaptive
  b_target: target bits on 1 DMT symbol for modes 1 and 2
  bitcap: maximum possible number of bits at each frequency bin
  cb =1 for cplx BB and =2 for real BB

  Outputs
  -------
  b: bit distribution (N x U matrix)
  E: energy distribution (N x U matrix)

  Remarks
  Iterate waterfiling for each user 10 times
  Youngjae(Sean) Kim - modified J. Cioffi, April 2024
```

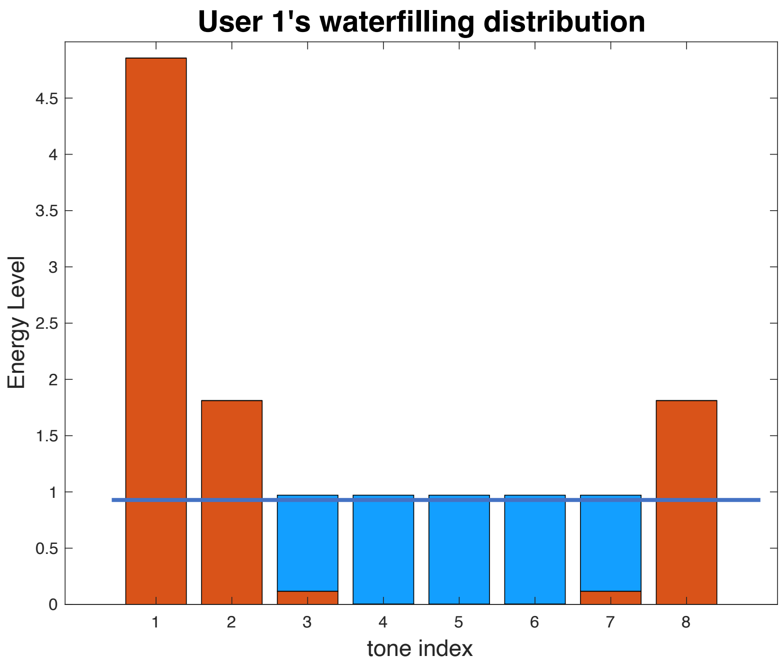
```
>> [b, E,K,NF] = iw(8, 2, H3.*conj(H3), Noise, [1 1], gap, [1 ; 1], [30 24],1)
b =   0   7.9340
      0   7.7055
 3.0668   0.3275
 7.8936        0
 8.1214        0
 7.8936        0
 3.0668   0.3275
      0   7.7055
E =   0   0.7665
      0   0.7660
 0.8535   0.1563
 0.9670        0
 0.9676        0
 0.9670        0
 0.8535   0.1563
      0   0.7660
K =   0.9711   0.7697
NF =
 97.4038   0.0031
  1.8119   0.0037
  0.1176   0.6134
  0.0041   4.0158
  0.0035      Inf
  0.0041   4.0158
  0.1176   0.6134
  1.8119   0.0037

>> sum(b) % = 30.0422   24.0000
sum(E) % =     4.6087   2.6111
>> sum([S1  S2],1) =   3.0577   2.1313
```

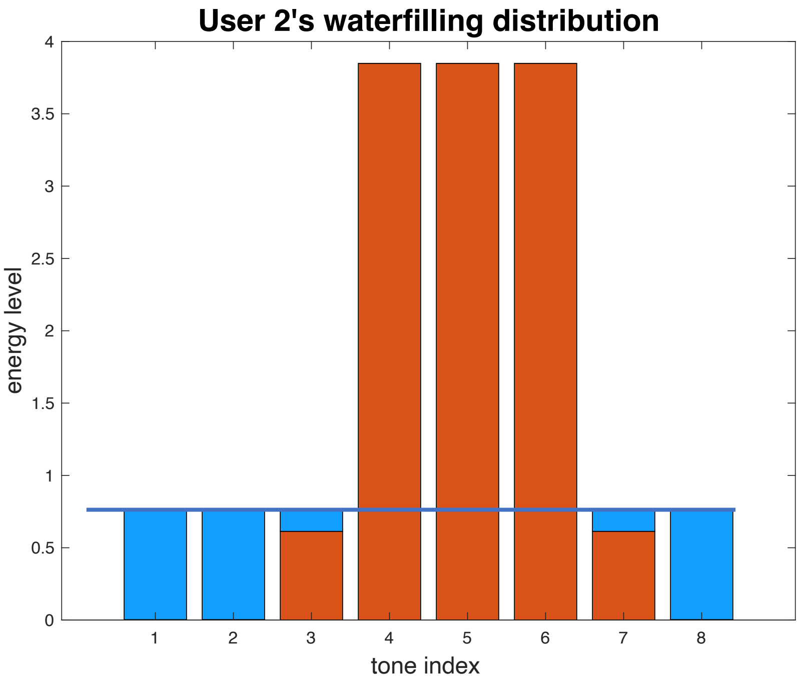
```
>> [b1 b2] %(osb)
   0   8
   0   8
   7   0
   8   0
   0   0
   8   0
   7   0
   0   8
```

```
>> [S1 S2] = %(osb)
      0     0.6375
      0     0.7469
 0.7017          0
 0.8272          0
      0          0
 0.8272          0
 0.7017          0
      0     0.7469
```

Matches bit rate now with fractional bits, but more energy

IW Water-filling outputs





- Water levels are not exactly the same (just close this time)

Command for this plot? bar([1:8],min([NF(:,1)' ; E(:,1)'],K(1)*5),'stacked')

STANFORD

End Lecture 16