Lecture 14

MAC Design by Weighted Sums

May 18, 2026

JOHN M. CIOFFI

Hitachi Professor Emeritus of Engineering

Instructor EE379B – Spring 2026

Announcements & Agenda

- Announcements
  - PS7 - Final homework, due June 5.
  - Section 5.4 continued

- Agenda
  - Maximum rate sum solution (maxRMAC)
  - Minimum energy-sum solution (minPMAC)
  - MAC Design: Specify rate and energy vectors (admMAC)

Maximum rate-sum solution (maxRMAC)

Section 5.4.4

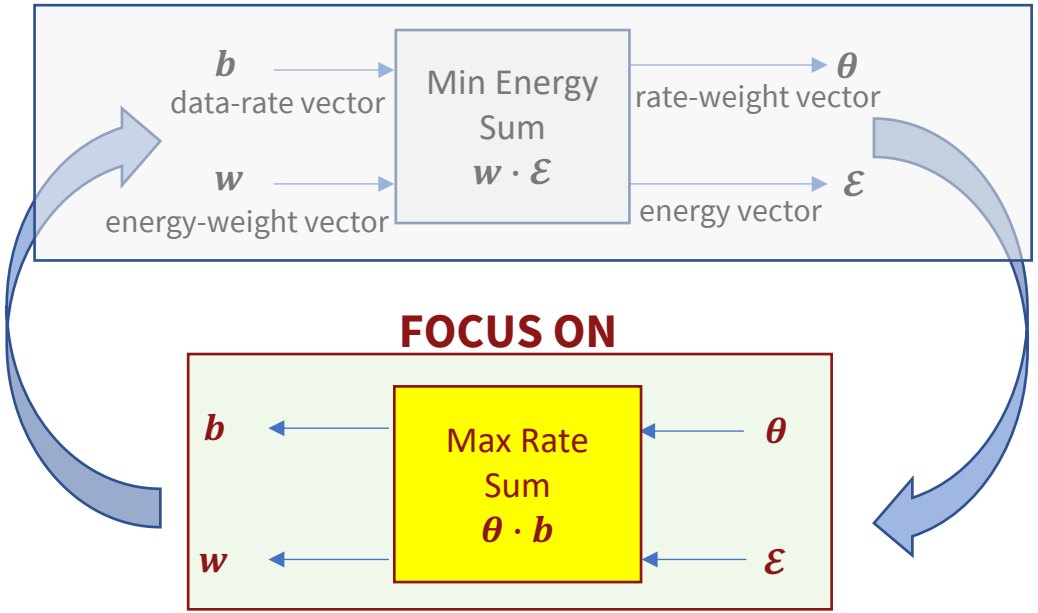
Maximize weighted rate sum

- **Maximize weighted rate** sum b, given $\boldsymbol{\mathcal{E}}$, with
  - There is no rate limit.
  - The rate weights $\boldsymbol{\theta} \succcurlyeq \mathbf{0}$, (non-negative) are given.
    - These predetermine decoding order.
    - A per user (or sometimes total) energy/power budget is also given.

$$\max_{\{R_{\boldsymbol{xx}}(u)\}} \sum_{u=1}^{U} \theta_u \cdot b_u$$

$$ST: \quad \boldsymbol{\mathcal{E}}_{\boldsymbol{x}} \preceq [\mathcal{E}_{1,max}\ \mathcal{E}_{2,max}\ \ldots \mathcal{E}_{U,max}]^* = \boldsymbol{\mathcal{E}}^*_{max} \succeq \mathbf{0}$$

$$\boldsymbol{\mathcal{E}} \succeq \mathbf{0} \quad .$$



maxRMACmimo – M. Umer 2026

- function [Eun, w, bun] = maxRMACmimo(H, Lxu, Eu, theta, cb)
  - inputs

H is the channel tensor $L_y \times U \times \overline{N}$
Lxu is number of antennas for users
Eu is the given $1 \times U$ energy vector $\boldsymbol{\mathcal{E}}$
theta is the rate weight vector $\boldsymbol{\theta}$
cb =1 cplx, =2 real

  - outputs

Eun is the $U \times \overline{N}$ matrix containing $\mathcal{E}_{u,n}$
(does not handle MAC $L_x > 1$)
w is the energy-weight vector $\boldsymbol{w}$
bun is the $U \times \overline{N}$ bit matrix containing $b_{u,n}$
(Eun is related to **bun** directly, fixed theta)

```
>> H=zeros(1,2,1);
>> H(1,1,1)=80;
H(1,2,1)=60;
>> Eu=[1 1];
>> theta=[1 1];
>> cb=2;
>> [Rxxs, Eun, w, bun] = maxRMACmimo(H, Lxu, Eu, theta, cb)
Eun =
   1.0000
   1.0000
w =
   0.6391
   0.3586
bun =
   6.3220
   0.3219
>> sum(bun,2)' =   6.3220   0.3219
>> sum(bun,'all') =   6.6439
>> [Eun, w, bun] = maxRMAC_cvx(H, Eu, theta, 1);
>> Eun' =   1.0000   1.0000
>> w' =     1.2952   0.7230
>> bun =   11.6443   0.6437
>> sum(bun) =   12.2880
```

If N=1 and complex bb, maxRMAC adjusts energy. Otherwise, use EVEN $N > 1$ if cb=2 (real bb)

Always trade dimensions for energy

Theta effect is only order: rate sums are the same.

```
>> theta=[1 2]; % reverses order
>> [Rxxs, Eun, w, bun] = maxRMACmimo(H, Lxu, Eu, theta, 2)
w =
   0.6399
   1.3597
bun =
   0.7368
   5.9071
>> sum(bun) =    6.6439
```

The theta has only changes order – sum rate is constant.

- **MIMO USE:** Runs slower, but allows variable number of transmit antennas per user: Lxu (also uses CVX),
- Also has an Rxxs output, for which trace{Rxxs(u)} is Eu.

```
[Rxxs, Eun, w, bun] = maxRMACmimo(H, [1 1], Eu, theta, 1)

Rxxs =  2x1 cell array
   {[1.0000]}
   {[1.0000]}
Eun =
   1.0000
   1.0000
w =
   1.2789
   0.7194
bun =
 1.4734
  10.8146
sum(bun) =   12.2880
```

More MIMO Examples

```
>> H2 =
   4   3   2   1
   5   6   7   8
[Rxxs, Eun, w, bun] = maxRMACmimo(H2, [2 2], [5 6], [1 1] , 1)
Rxxs = 2x1 cell array
Eun' % =   5   6
W' =   0.3880   0.3275
bun =
   8.6447
   6.8028
>> Rxxs{:,:}
=
 3.5985   2.2457
 2.2457   1.4015
=
 1.6867   2.6972
 2.6972   4.3133
```

Both users are MIMO

```
[Rxxs, Eun, w, bun] = maxRMACmimo(H2(:,1:3), [2 1], [5 6], [1 1] , 1)

Rxxs =  2x1 cell array
Eun =
   5.0000
   6.0000
W' =   0.3811   0.3144
bun =
   8.5830
   5.0661
>> Rxxs{:,:}
 =
   3.9150   2.0610
   2.0610   1.0850
 =
   6.0000
```

One user is MIMO

- The second antenna on user 1 helps both users' data rates. Why?

Vector DMT Examples

```
>> Eu = [    1    1 ];
>> theta = [    1    1];
>> H4
H4(:,:,1) =   80   60
H4(:,:,2) =   80   60
H4(:,:,3) =   80   60
H4(:,:,4) =   80   60
[Eun, w, bun] = maxRMAC_cvx(H4, 4*Eu', theta', 1)
Eun =
   1.0000   1.0000   1.0000   1.0000
   1.0000   1.0000   1.0000   1.0000
w' =   0.6391   0.3586
bun =
  12.6441  12.6441  12.6441  12.6441
   0.6438   0.6438   0.6438   0.6438
>>  [Eun, w, bun] = maxRMAC_cvx(H4, 2*Eu', theta', 2)
Eun =
   1.0000   1.0000
   1.0000   1.0000
bun =
   6.3220   6.3220
   0.3219   0.3219
>>sum(bun,'all') =  13.2879

Rxx(:,:,1) =          Rxx(:,:,2) =
   1   0              1   0
   0   1              0   1
bsum =  13.2879
bsum_lin =  2.1174
```

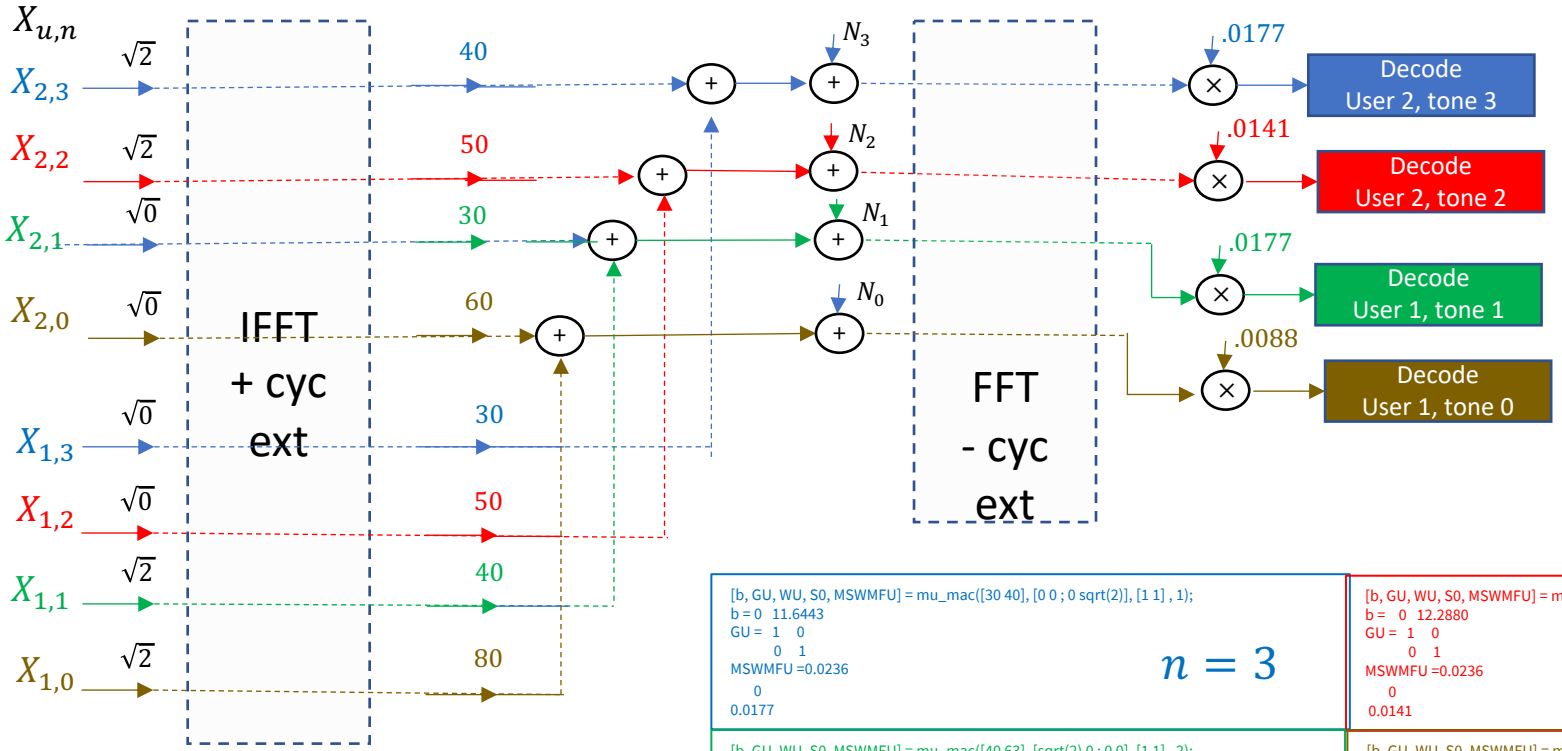
Linear substantially less– why?

```
>> H4x=zeros(1,2,4);
>> H4x(:,:,1)= [80 60];
>> H4x(:,:,2)= [40 30];
>> H4x(:,:,3)= [50 50];
>> H4x(:,:,4)= [30 40];
>> [Rxxs, Eun, w, bun] = maxRMACmimo(H4x, Lxu, 4*Eu, theta, 1)

Eun = 2.0002   1.9998   0.0000   0.0000
        0.0000   0.0000   2.0001   1.9999

w' =   0.4999   0.4999

bun =  13.6441  11.6441   0.0651         0
                0         0  12.2230  11.6442
sum(bun'*theta') =  49.2206

>> [Rxx, bsum , bsum_lin] = SWF(Eu, H4x(:,:,1:4), [1 1], ones(1,1,4), 1)
Rxx(:,:,1) =   2.0001       0
                    0       0
Rxx(:,:,2) =   1.9996       0
                    0       0
Rxx(:,:,3) =   0.0003       0
                    0  2.0000
Rxx(:,:,4) =        0       0
                    0  2.0000
bsum =  49.2206
bsum_lin =  48.4228
```

Not Hermitian symmetric

Linear is close – why?

System diagram for previous example



No feedback between users on these FDM – tonal GDFEs

```
[b, GU, WU, S0, MSWMFU] = mu_mac([30 40], [0 0 ; 0 sqrt(2)], [1 1] , 1);
b = 0  11.6443
GU =  1   0
      0   1
MSWMFU =0.0236
    0
0.0177
```

$n = 3$

```
[b, GU, WU, S0, MSWMFU] = mu_mac([50 50], [sqrt(2) 0 ; 0 0], [1 1] , 2);
b =   0  12.2880
GU =  1   0
      0   1
MSWMFU =0.0236
    0
 0.0141
```

$n = 2$

```
[b, GU, WU, S0, MSWMFU] = mu_mac([40 63], [sqrt(2) 0 ; 0 0], [1 1] , 2);
b =  11.6443      0
GU =  1   0
      0   1
MSWMFU =0.0236
  0.0177
    0
```

$n = 1$

```
[b, GU, WU, S0, MSWMFU] = mu_mac([80 60], [sqrt(2) 0 ; 0 0], [1 1] , 2);
b =  6.8220       0
GU =  1   0
      0   1
MSWMFU =
  0.0088
    0
```

$n = 0$

- FDM solution simplifies to linear receiver as optimum.

maxRESMAC – only energy-sum constraint

- function [Eun, w, bun] = maxR**ES**MACmimo(H, Lxu, Etotal, theta, cb)
  - inputs

H is the channel tensor $L_y \times U \times \overline{N}$
Lxu is number of input antennas for user u
Etotal is the total energy sum $\mathcal{E}$
theta is the rate weight vector $\boldsymbol{\theta}$
cb =1 cplx, =2 real

  - outputs

Eun is the $U \times \overline{N}$ matrix containing $\mathcal{E}_{u,n}$
(does not handle MAC $L_x > 1$)
w is the order/rate vector $\boldsymbol{w}$
bun is the $U \times \overline{N}$ bit matrix containing $b_{u,n}$

```
[Eun, w, bun] = maxRESMAC_cvx(H4x, 8, [ 1 1]',1)

Eun =
 2.0003   1.9998   1.0000   0.0000
   0.0000   0.0000   1.0000   1.9998
w =   0.4999   0.4999
bun =
 13.6442  11.6442  11.2883   0.0000
      0       0   0.9997  11.6442
>> sum(bun,'all') % =  49.2206
>> sum(Eun,'all') % =   8.0000
>> sum(bun') % = 36.5767   12.6439

>> [Rxx, bsum , bsum_lin] = SWF(Eu, H4x(:,:,1:4), [1 1], ones(1,1,4), 1);

Rxx(:,:,1) =              Rxx(:,:,3) =
   2.0001  0              0.0003       0
      0       0           0            2.0000
Rxx(:,:,2) =              Rxx(:,:,4) =
   1.9996  0              0       0
      0       0           0   2.0000
bsum =  49.2206
bsum_lin =  48.4228
```

Minimum energy-sum solution (minPMAC)

Section 5.4.4.5

minimize weighted energy sum

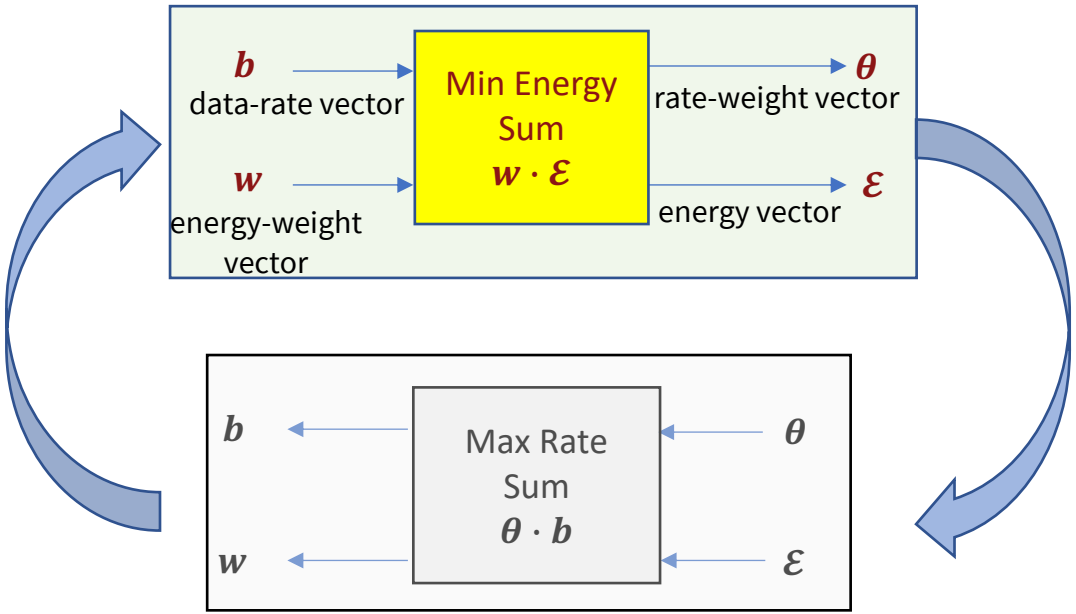
- **Minimize weighted energy** sum $\mathcal{E}$, given $\boldsymbol{b}$, with
  - There are no energy limits.
  - The energy weights $\boldsymbol{w} \succcurlyeq \mathbf{0}$, (non-negative) are given.
    - How important relatively are the users?
    - Given some fixed desired user rates; order is optimized.

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

$$ST: \quad \boldsymbol{b} \succeq [b_{1,min}\ b_{2,min}\ \ldots b_{U,min}]^* = \boldsymbol{b}^*_{min} \succeq \mathbf{0}$$

$$\boldsymbol{\mathcal{E}} \succeq \mathbf{0} \quad .$$

FOCUS ON



See S14 for theory and matrix calculus

minPMACmimo – M. Umer - 2026

- [Eun, theta, bun, FEAS_FLAG, bu_a, info] = minPMACmimo(H, Lxu, bu, w, cb)
  - inputs

H is the channel tensor $L_y \times U \times \bar{N}$
Lxu is $U \times 1$ vector of antennas/user
bu is the given min data-rate $\boldsymbol{b}$
w is the energy weight vector $\boldsymbol{w}$
cb =1 cplx; cb =2 real

Freq Domain

- outputs

Outputs:
FEAS_FLAG feasibility flag: 0 infeasible, 1 single order, 2 time-sharing.
bu_a achieved user rates in bits per channel use.
info
theta: U 1 user constraints (Lagrange multipliers)
clusters: cell containing user clusters
orderings: cell contaiing user order(s)
frac: fraction of each desing's use
Rxx: [UL x UL x N optimized autocorrelation matrices
bu_vertices: The vertices shared in any time-sharing
b_achieved: U1 average data rates achieved
bun: UL N bit distributions
Eu_avg: U 1 average user energies

```
>> H=zeros(1,2,1)  % dimensioning a tensor
%     H =  0    0
>> H(1,1,1)=80;
>> H(1,2,1)=60
%    H=  80    60
>> b' = [3 3];
>> w' = [ 1 1];
[FEAS_FLAG, bu_a, info] = minPMACmimo(H, Lu, b, w,2);
Antenna configuration: [1  1], total=2
Found 2 clusters, 1 possible decoding orders
minPMACMIMO completed in 0.016 seconds, FEAS_FLAG=1
> info.Eu_avg % =
%    0.6300    0.0175
>> info.theta' % =
%    1.2800    1.2955
>> info.b_achieved' % =
%    2.9999    3.0001
>> info.orderings{:} % =
%    1    2
>> 0.5*log2(1+(6400*info.Eu_avg(1))/(1+3600*info.Eu_avg(2))) % = 2.9999
>> 0.5*log2(1+(3600*info.Eu_avg(2))) % = 3.0001
```

Decode user 1 last (best position or top).

- Called subroutines (6)
  - See 379B web page

Increase N to 4 tones

```
>> FEAS_FLAG, bu_a, info] = minPMACmimo(H, Lu, 4*b, w, 2);
>> info.R{:,:}
ans(:,:,1) =
  0.6300      0
       0   0.0175
ans(:,:,2) =
  0.6300      0
       0   0.0175
ans(:,:,3) =
  0.6300      0
       0   0.0175
ans(:,:,4) =
  0.6300      0
       0   0.0175
>> info.Eu_avg % =    0.6300    0.0175
>> info.theta % =      1.2800    1.2955
>> info.b_achieved % =
   3.0000   3.0000   3.0000   3.0000
   3.0000   3.0000   3.0000   3.0000
>> info.bu_vertices' % =
12.0001  11.9999
```

Real bb, so ignore upper-half tones that repeat lower half.

4 real dimensions

```
>> [FEAS_FLAG, bu_a, info] = minPMACmimo(H, Lu, 5*b, w,2);
minPMACMIMO completed in 0.051 seconds, FEAS_FLAG=1
>> info.R{:,:}
5.0917      0              5.0917      0
   0      0.0500            0      0.0500
5.0917      0              5.0917      0
     0   0.0500               0   0.0500
>> info.Eu_avg % =   5.0917   0.0500
>> info.theta' % =   10.2400   10.2840
>> info.b_achieved % =
   3.7500   3.7500   3.7500   3.7500
   3.7500   3.7500   3.7500   3.7500
>> info.bu_vertices' % =   15.0000   15.0000
```

Result is correct, but energy too high, perhaps.

- Unequal gains on tones

No conjugate symmetry, Must be BB complex

```
>> H4x(:,:,1) = [ 80 60];
>> H4x(:,:,2) = [ 40 30];
>> H4x(:,:,3) = [ 50 50];
>> H4x(:,:,3) = [ 30 40];
```

```
>> [FEAS_FLAG, bu_a, info] = minPMACmimo(H4x, [1 1], 8*b, w,1);
minPMACMIMO completed in 0.043 seconds, FEAS_FLAG=1
>> info.Rxx{:,:}
   1.8098      0                 1.8094      0
      0   0.0003                    0   0.0000

   0.0000      0                 0.0000      0
      0   1.8097                    0   1.8097
>> info.theta' % =   3.6201   3.6206
>> info.bun % =
  12.5006  11.4999       0       0
   0.9993   0.0000  11.5001  11.5001
>> info.orderings % =   {[1 2]}
>> info.bu_vertices' %=   24.0005   23.9995
```

8 real dimensions FDM showing again

Still decode user 1 last (best position or top) all tones, both examples, slightly not "FDM"

Equal Thetas

```
>> [FEAS_FLAG, bu_a, info] = minPMACmimo([80 80], [1 1], [3 3], w,2);
Found 1 cluster, 2 possible decoding orders
minPMACMIMO completed in 0.192 seconds, FEAS_FLAG=2
>> info.Rxx{:,:}
  0.2656       0
       0   0.3742
>> info.orderings{:}
   2   1
   1   2

>> info.frac' % =   0.5246   0.4754
>> info.bu_vertices % =
  5.3661   0.3868
  0.6339   5.6132
```

Not the desired b, but sum is same.

Order switches

```
>> [FEAS_FLAG, bu_a, info] = minPMACmimo([80 80 80], [1 1 1], [5 4 3], [1 1 1],2);
minPMACMIMO: Solving for 3 users, 1 tones, 1 RX antennas
Antenna configuration: [1  1  1], total=3
Found 1 clusters, 6 possible decoding orders
minPMACMIMO completed in 0.417 seconds, FEAS_FLAG=2
>> info.Rxx{:,:} % =   1.0e+03 *
   0.7841       0       0
        0   1.4415       0
        0       0   0.3959
>> info.frac' % =   0.4215   0.3046   0.2739
>> info.orderings
   {[3 2 1]}
   {[3 1 2]}
   {[2 1 3]}
>> info.bu_vertices % =
  11.1294   0.3133   0.7878
   0.7525  11.5686   0.5758
   0.1181   0.1181  10.6364
>> (info.bu_vertices(:,1)*info.frac(1) + info.bu_vertices(:,2)*info.frac(2) +
info.bu_vertices(:,3)*info.frac(3) )'% =   5.0020   3.9990   2.9990
```

Split vertices 52/48 ~50/50 in decoder share
Transmitter with $\Gamma = 0$ dB can send
3 bits for all symbols, both users – Why?

Split vertices 21/50, 15/50, 14/50 in decoder share

Single-tone channel with U > L

- Forces a secondary user component (equal theta for >= 2 user components)

```
Htemp = [
   80   40   30
   30  -50  -15 ];
[FEAS_FLAG, bu_a, info] = minPMACmimo(Htemp, [1 1 1], [7 5 6] , [1 1 1],1);
minPMACMIMO: Solving for 3 users, 1 tones, 2 RX antennas
Antenna configuration: [1  1  1], total=3
Found 2 clusters, 2 possible decoding orders
minPMACMIMO completed in 0.251 seconds, FEAS_FLAG=2
>> info.Rxx{:,:} % =
   0.0967       0       0
        0  0.0891       0
        0       0  0.0560
>> info.theta' % =
   0.1968   0.1968   0.2892
>> info.bu_vertices % =
   8.5894   5.8731
   3.4105   6.1269
   6.0000   6.0000
>> info.frac % = 0.4145    0.5855
>> info.orderings % =
   {[2 1 3]}
   {[1 2 3]}
>> info.b_achieved{1}'*info.frac(1) + info.b_achieved{2}'*info.frac(2) % =
   6.9990   5.0010   6.0000
```

Example 5.4.12 continues This example to 4 tones

That example shows at lower data rates, the vertex sharing goes away with N>1, because effectively "tone-sharing"

~60/40 split for users 2 and 1, which move to top and cancel user 3 User 3 treats both users 2 and 1 as noise

Multiple Clusters

```
[FEAS_FLAG, bu_a, info] = minPMACmimo([80 60 80 60], [1 1 1 1], [3 4 5 2]  , [1 1 1 1],2);
minPMACMIMO: Solving for 4 users, 1 tone, 1 RX antennas
Found 2 clusters, 4 possible decoding orders (% only 3 need be used)
>> info.Rxx{:,:} %=   1.0e+04 *
   1.1812      0      0      0
        0 0.0001      0      0
        0      0 3.0131      0
        0      0      0 0.0000
>> info.theta' % =   1.0e+04 * [ 8.3886   8.3887   8.3886   8.3887]
>> info.frac' % =    0.0583   0.3451   0.5966  % 3 designs
>> info.clusters % =   {[1 3]}   {[2 4]}
>> info.bu_vertices %  =
   7.0859   7.0859   0.2386
   5.7000   0.7771   5.7000
   0.9141   0.9141   7.7614
   0.3001   5.2230   0.3001

>> info.orderings %=  (Users 2 and 4 have small energy – go last)
   {[3 1 4 2]}
   {[3 1 2 4]}            % orderings are first to last
   {[1 3 4 2]}

>>  info.b_achieved{1}'*info.frac(1) + info.b_achieved{2}'*info.frac(2) +
info.b_achieved{3}'*info.frac(3)%  =

   3.0010   4.0010   4.9990   1.9990
```

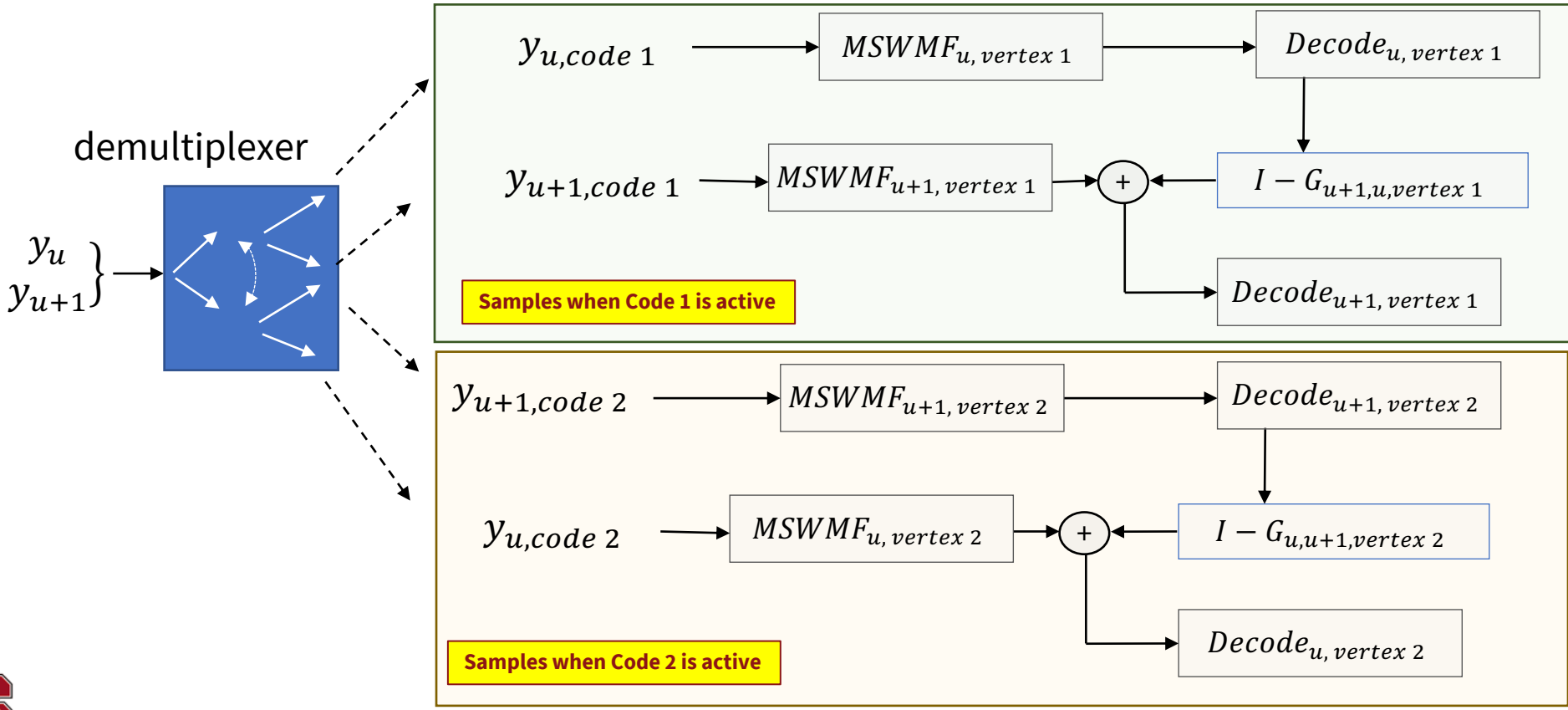
4 users in TWO clusters reduces the number of designs.

A 4-dimensional hyperspace has 3-dimensional hyperplanes, so a design on such a hyperplane needs only 3 vertices.

A design with clusters [3 1 4 2] and [3 1 2 4] needs only one More design with [1 3 x y] where [x y] can be either [2 4] or [4 2]

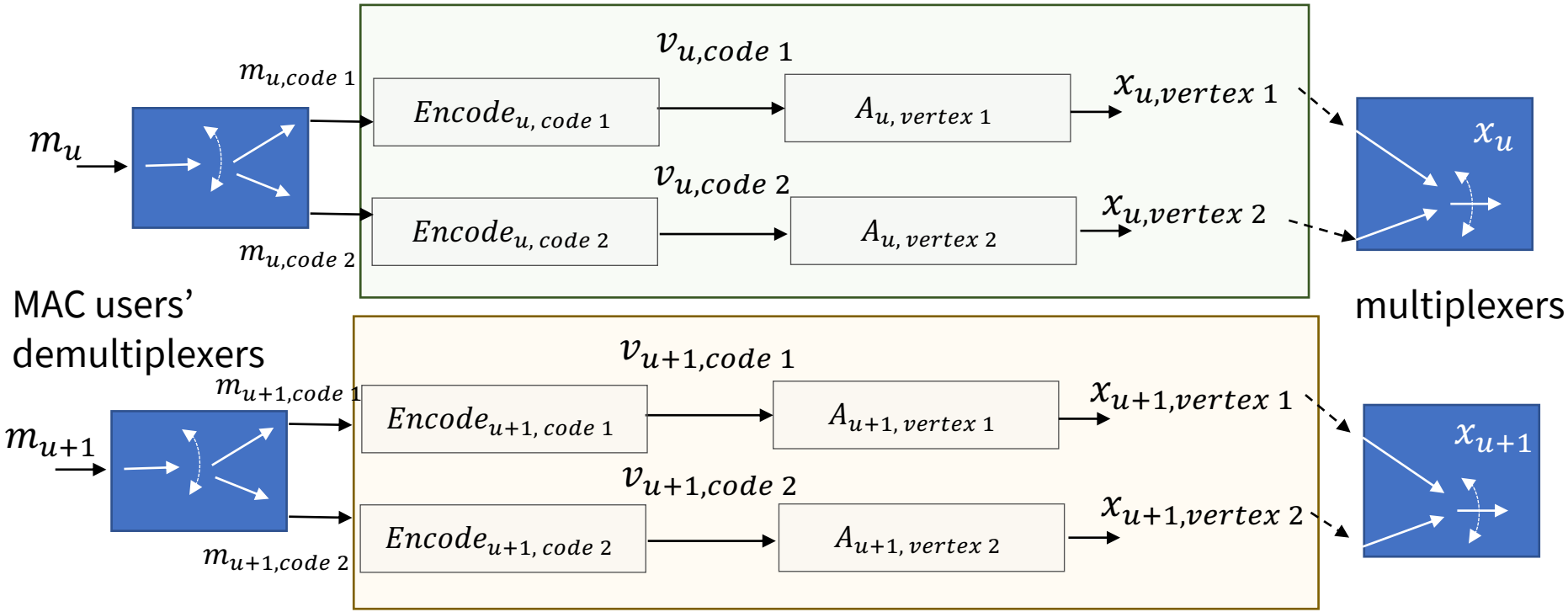
illustrate cluster receiver

- Two GDFE receivers for frac(1) …. frac(2) of the received symbols



illustrate cluster encoder modulator

- Two (U=2) encoders/modulator transceivers for frac(1) …. frac(U) of the received symbols

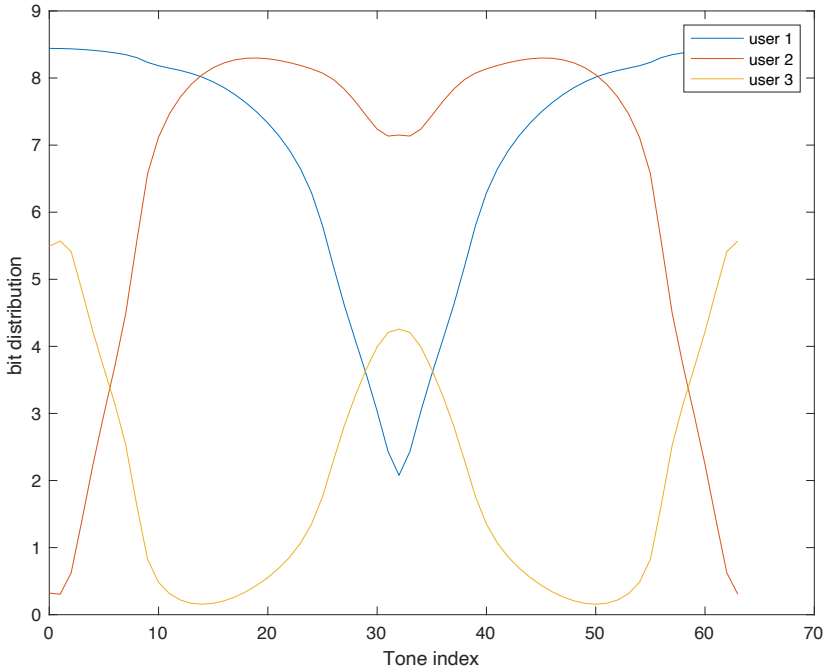


Multicluster imposes packet-level synchronization so that each user encoder at any symbol time corresponds to the proper code for that symbol.

64-tone example

```
h64=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;
>> H64 = fft(h64, 64, 3);
>> [FEAS_FLAG, bu_a, info] = minPMACmimo(H64, [1 1 1], [445 412 132]', [1 1 1]',1);
minPMACMIMO: Solving for 3 users, 64 tones, 2 RX antennas
Antenna configuration: [1  1  1], total=3
Found 2 clusters, 2 possible decoding orders
minPMACMIMO completed in 2.626 seconds, FEAS_FLAG=2

>> info.theta' % =   2.6917   2.6917   2.2404
>> info.frac % =    0.9904    0.0096
sum(info.b_achieved{1}'*info.frac(1) + info.b_achieved{2}'*info.frac(2)) % =
 445.0010  411.9990  132.0000
```

- Every tone has a cluster of 2 users (3 & 2)
  - Codes for 2 and 3 occur 1/100 and 99/100 symbols
- User 1 is removed via GDFE cancellation
  - Has only 1 code (same all symbols)



The GDFE's energize all 3 users, but some "FDM" in that only 1% time on second design

Larger finite N (I tried to 1024) did not reduce the 1%

Corresponding MAC Design (N = 8), HWH7

```
N=64;
>> info.orderings{1} % =    3    2    1
% This is already ordering of H64, which for matlab is 1 2 3
Rxx=cell2mat(info.Rxx);
cb=1;
Usize=[1 1 1];
Bu=zeros(N,3);
Aopt=zeros(3,3,N);
for n=1:N
Aopt(:,:,n)=sqrtm(Rxx(:,:,n));
end
for n=1:N
[ Bu(n,:) GU(:,:,n), WU(:,:,n), S0(:,:,n), MSWMFU(:,:,n)] = ...
mu_mac(H64(:,[1 2 3],n), Aopt([1 2 3],[1 2 3],n), Usize , cb);
end
>> sum(Bu) =  445.2750  411.7250  132.0000 %  checks

>> GU(:,:,23) % =
   1.0000 + 0.0000i  -1.4537 + 0.3749i  -0.2525 + 0.2554i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.6084 + 0.3165i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
```

One of 64 Feedback Sections

```
>> GU(:,:,23) % =
   1.0000 + 0.0000i  -1.4537 + 0.3749i  -0.2525 + 0.2554i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.6084 + 0.3165i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
```

One of 23 feedforward Sections

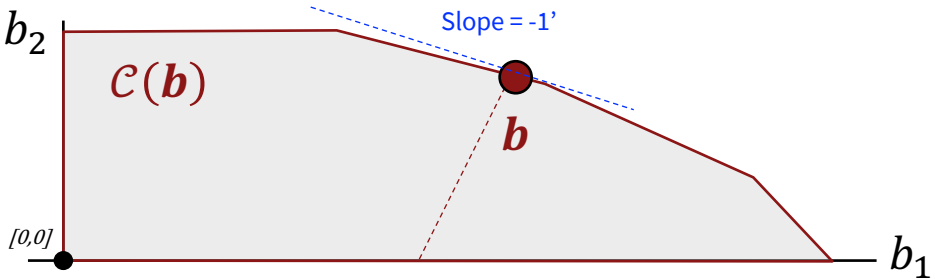
```
>> WU(:,:,23) % =
   0.0107 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0030 + 0.0008i   0.0021 + 0.0000i   0.0000 + 0.0000i
  -0.5852 + 0.3807i  -0.4797 + 0.2495i   0.7902 + 0.0000i

%%%%%%%%%%%%%%%%%%

>> MSWMFU(:,:,23) % =

   0.0517 + 0.0774i  -0.0446 + 0.0048i
   0.0111 + 0.0162i   0.0412 - 0.0047i
   0.0900 + 0.0338i   0.0200 - 0.0281i
```

- So, this is design, see problems 7.1-4

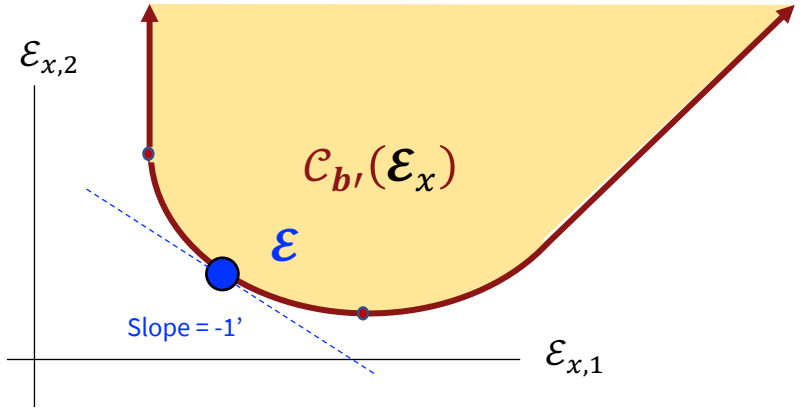
admMAC and admissibility

Section 5.4.5

Capacity region(s)



- $\mathcal{C}(\boldsymbol{b})$ contains all possible weighted **rate** sums $\sum_{u=1}^{U} \theta_u \cdot b_u$ that meet **energy-vector** constraint $\boldsymbol{\mathcal{E}} \preccurlyeq \boldsymbol{\mathcal{E}_x}$.
- The max-b-sum point is "highest" $\boldsymbol{b}$ in $\mathcal{C}(\boldsymbol{b})$.
- If $\boldsymbol{\mathcal{E}} \preccurlyeq \boldsymbol{\mathcal{E}_x}$, **then $\boldsymbol{b}$ is admissible.**
- If not, find minimum energy-sum to achieve $\boldsymbol{b}$.

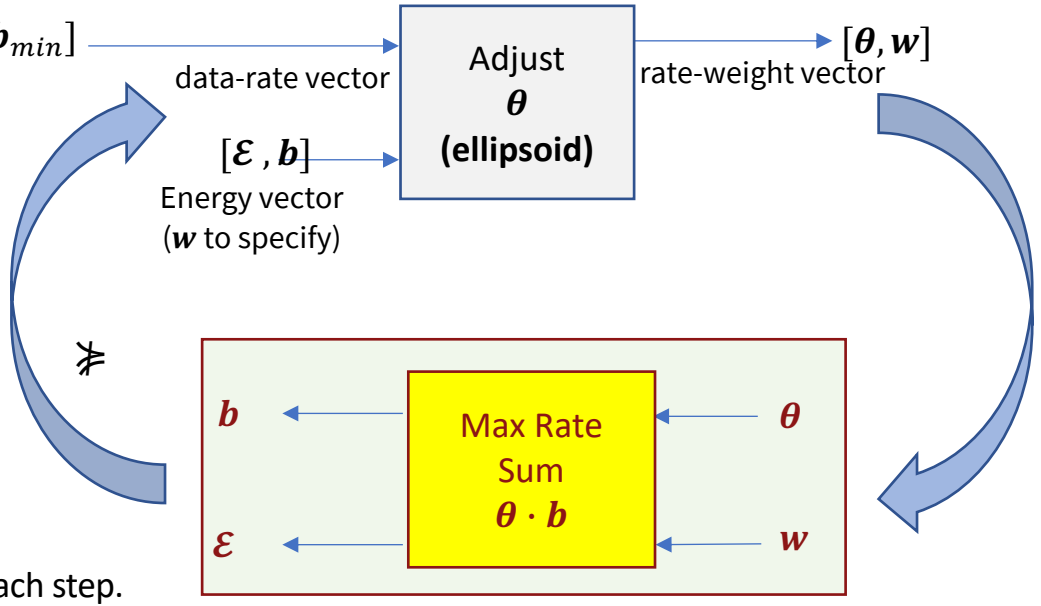


- $\mathcal{C}_{b'}(\boldsymbol{\mathcal{E}_x})$ contains all possible weighted **energy** sums $\sum_{u=1}^{U} w_u \cdot \mathcal{E}_u$ that meet **rate-vector** constraint rate vector $\succcurlyeq \boldsymbol{b'}$.
- The min-E-sum point is "lowest" $\boldsymbol{\mathcal{E}}$ in $\mathcal{C}_{b'}(\boldsymbol{\mathcal{E}})$.
- If $\boldsymbol{b'} \succcurlyeq \boldsymbol{b}$ **, then $\boldsymbol{\mathcal{E}}$ is admissible.**

Admissible: meet both $\boldsymbol{b}$ and $\boldsymbol{\mathcal{E}}$ constraints

Using admMAC to find solution

- Specify the $\mathcal{E}_{max}$ and $\boldsymbol{b}_{min}$.
- Find the weights $\boldsymbol{\theta}$ **and** $\boldsymbol{w}$.
- When $\boldsymbol{b} \not\geq \boldsymbol{b}_{min}$, change $\boldsymbol{\theta}$.
- When$\mathcal{E} \not\leq \mathcal{E}_{max}$, change $\boldsymbol{w}$.
- If there is a solution in $\mathcal{C}(\boldsymbol{b})$.
- This process finds it.
  - Both are convex methods that improve at each step.



$$\min_{\{R_{\boldsymbol{X}\boldsymbol{X}}(u,n)\}} \quad 0$$

$$ST: \quad 0 \preceq \boldsymbol{b}_{min} \preceq \sum_n [b_{1,n}\ b_{2,n} \ldots b_{U,n}]$$

$$0 \preceq \sum_n \text{trace}\{R_{\boldsymbol{X}\boldsymbol{X}}(u,n)\} \preceq [\mathcal{E}_1\ \mathcal{E}_2 \ldots \mathcal{E}_U]^* = \boldsymbol{\mathcal{E}}_{\boldsymbol{x}}$$

$$\boldsymbol{b}_n \in \mathcal{A}_n(\bar{H}_n, \{R_{\boldsymbol{X}\boldsymbol{X}}(u,n)\}_{u=1,\ldots,U} \quad .$$

admMACmimo program – thank you Umer!

admMACmimo Determine feasibility of target rates under energy constraints
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMACmimo(H, Lxu, bu, Eu, cb)
determines whether the target rate vector bu is feasible for the
(noise-whitened) multi-user MIMO MAC channel H given per-user energy
constraints Eu.

Inputs:

H channel tensor [Ly, sum(Lxu), N] with user antennas stacked.
For SISO (Lxu=1), H is Ly-by-U-by-N. H(:,:,n) is the channel
for the n-th tone.

Lxu antennas per user (scalar or 1-by-U vector). If scalar, every
user has Lxu transmit antennas; otherwise user u has Lxu(u)
transmit antennas. Use Lxu=1 for SISO case.

bu target rate of each user, length-U vector (bits per channel use).

Eu energy/symbol constraint for each user, length-U vector.

cb baseband type: 1 for complex, 2 for real.

Outputs:

FEAS_FLAG feasibility indicator:
0 = target rates infeasible
1 = feasible via single decoding order
2 = feasible via time-sharing between multiple vertices

bu_a U-by-1 vector of achieved sum rate per user.

Rxxs transmit covariance matrices:
- if FEAS_FLAG=0: returns 0
- if Lxu is scalar: Lxu-by-Lxu-by-U-by-N tensor
- if Lxu is vector: U-by-N cell array of matrices

Eun U-by-N matrix of per-user per-tone energy allocations.
If FEAS_FLAG=0, returns 0.

theta U-by-1 Lagrangian multipliers for rate constraints.

w U-by-1 Lagrangian multipliers for energy constraints.

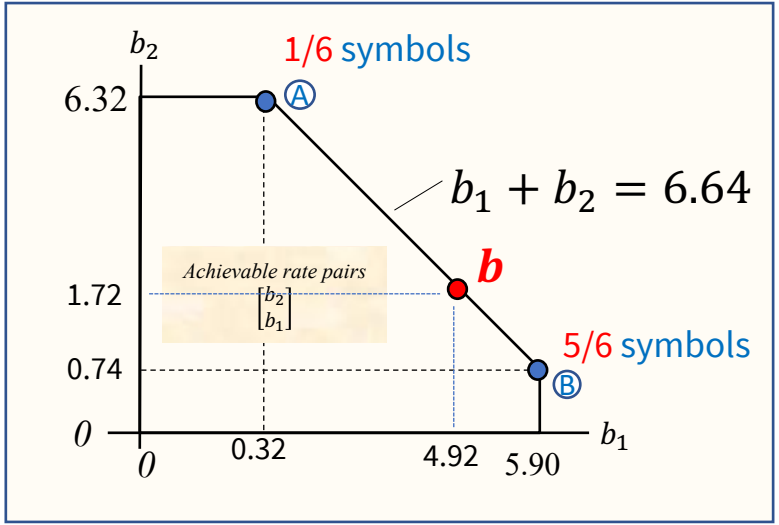
info detailed solution information (depends on FEAS_FLAG):
- if FEAS_FLAG=0: empty cell
- if FEAS_FLAG=1: 1-row table with {bu_v, bun, order}

struct with fields:
H: [2 × 3 × 64 double]
Lx: [1 1 1]
bu_min: [3 × 1 double]
w: [3 × 1 double]
cb: 1
theta: [3 × 1 double]
theta_unique: 2.9830
clusters: {[1 2 3]}
orderings: {3 × 1 cell}
weights: [3 × 1 double]
Rxx: {[3 × 3 × 64 double]}
bu_vertices: [3 × 3 double]
b_achieved: {3 × 1 cell}
bu_achieved: [3 × 1 double]
sol_status: 'Solved'
feasible: 1
idx_start: [1 2 3]
idx_end: [1 2 3]
rac: [3 × 1 double]
bun: [3 × 64 double]
Eu_avg: [0.9930 0.9508 0.9084]
elapsed_time:

Back to simple channel, pick pt on boundary

```
H = [80 60];
>> bu_min = [1.72;4.92];
>> Eu = [1;1];
Lxu=[1 1];
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w,
info] = admMACmimo(H, Lxu, bu_min, Eu, 2)
flag =  2
bu_a =
  1.7210
  4.9229
info = 2x8 table
     bu_v          bun          order     frac     clusterID
  0.73684   5.9071   {2×1 dbl}   {[1 2]}   0.82379     1
   6.322    0.32189  {2×1 dbl}   {[2 1]}   0.17621     1
>>Rxxs   % 2×1 cell array  same both vertices
   1     1
>> info.Eun{:,:}
> Eun =  % same both vertices
   1.0000
   1.0000
>> theta =
   1.0500
   1.1000
>> w =
   0.6719
   0.4279
```

```
>> [info.bun{:,:}]' % =
   0.7368   5.9071   % bottom vertex
   6.3220   0.3219   % top vertex
>> info.order{:} =
   1   2
   2   1
>> info.bu_v'*info.frac =
   1.7210
   4.9229  (vertex-sharing checks)
```



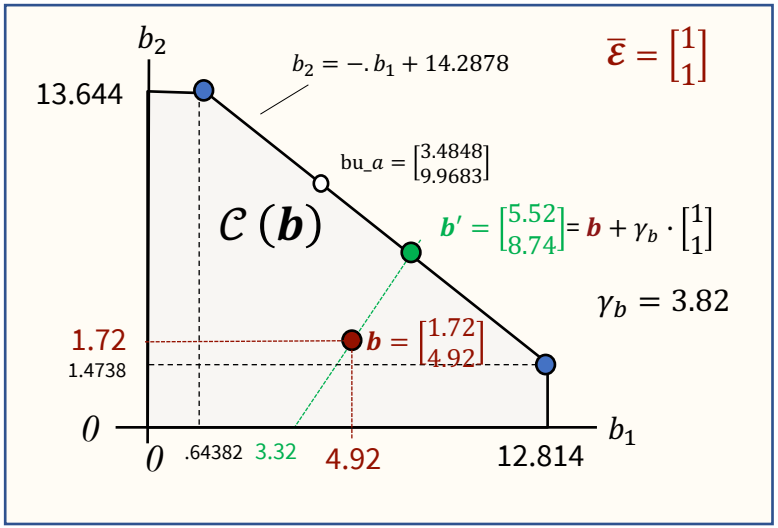

```
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] =
admMACmimo(H, Lxu, bu_min+[1 ; 1], Eu, 2);
 flag =  0
```

Confirms point is on boundary

- FEAS_FLAG = 2, vertex sharing
  - One vertex has equal thetas is enough

Complex channel, same rate pt – NOT on boundary

```
>> [FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] =
admMACmimo(H, Lxu, bu_min, 2*Eu, 1)
FEAS_FLAG =    2
bu_a =
   1.7200
   4.9200
Info.bu_v                    info.frac
    1.4738   12.8140          .3797
   13.6440    0.6438          . 0851
>> Eun' % =    2.0000    2.0000
>> theta' % =    1.0500    1.1000
>>[ info.bu_v'*info.frac]' %=  1.7200     4.9200
```




```
>> [FEAS_FLAG, bu_a, info] =
minPMACmimo(H, Lxu, bu_a, w, 1)

Info.Eu_avg:  % =  2.9574   0.1169
% lower sum – but user 2 exceeds its limit
theta % =6.0467   6.1492
FEAS_FLAG =    1
bu_a % = 5.52    8.72
```

- FEAS_FLAG = 2, vertex sharing

```
>>info.order{:} =
    1    2
    2    1
>> slope=-1;
>> int=info.bu_v(1,1)-slope*info.bu_v(1,2) =
  14.2878
>> rsum=sum(info.bu_v') % =   14.2878   14.2878
>> gammab=(rsum(1)-bu_min(2)-bu_min(1))/2 =
         3.8239
```

$\gamma_b = 3.4$ or roughly 10.2 dB (complex so x 3 dB)

FEAS_FLAG = 1, single vertex

Equal energy weight used here; others likely also will work.

```
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMACmimo(H, Lxu, bu_min+4*[1 1]',
2*Eu, 1) % flag =    0
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMACmimo(H, Lxu, bu_min+3.9*[1 1]',
2*Eu, 1) % flag =    0
[[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMACmimo(H, Lxu, bu_min+3.8*[1 1]',
2*Eu, 1) % flag =    2
bu_a' = 5.52    8.72
Margin is roughly 3.8*3 = 11.6 dB
```

admMACmimo

- There can be multi-solution vertex-sharing when in the interior of $\mathcal{C}(\boldsymbol{b})$.
  - Slope (or hyperplane normal vector) is not necessarily = -1 ($\cdot\boldsymbol{1}$).

- minPMACmimo can use any $\boldsymbol{w}$, including all equal (so energy sum), but does not guarantee a point in $\mathcal{C}(\boldsymbol{b})$.

- minPMACmimo may be preferred design method as it tends to produce larger margin.
  - Unless any user's energy is too large.
  - Then use admMAC , judiciously with the energies found – limiting any user energies that exceed allowed amounts.

Two users, high pass and low pass

- A past 2-user MAC with memory (user 2 at 1+.9D and user 1 at 1-D)

```
>> H8=zeros(1,2,8);
>> H8(1,1,:)=fft([1 .9],8);
>> H8(1,2,:)=fft([1 -1],8);
>> H8=(1/sqrt(.181))*H8;
>> b=[1 ; 1];
>> energy=[8 ; 8];
```

bu_v		bun	order	frac	clusterID	
16.692	22.594	{2×8 dbl}	{[1 2]}	0.34442	1	
21.42	17.866	{2×8 dbl}	{[2 1]}	0.57193	1	

```
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMACmimo(H8, Lxu,
18*b, (8/9)*energy, 1)
admMACMIMO: completed in 0.089 s (FEAS_FLAG=2, iters=1)
bu_a' % =18.0000   18.0000
Eun =
   1.9061   1.8975   0.7050   0.0000   0.0000   0.0000   0.7050   1.8975
   0.0000   0.0000   1.0419   1.6732   1.6809   1.6732   1.0419   0.0000
Theta' =   1.0500   1.1000
w =   0.5367   0.6372
>> sum(Eun,2)' =   7.1111   7.1111
>> info.bun{:,:} % =
   5.2860   5.0583   0.6447        0        0        0   0.6447   5.0583
        0        0   3.6453   5.0251   5.2535   5.0251   3.6453        0
   5.2860   5.0583   3.0089        0        0        0   3.0089   5.0583
        0        0   1.2811   5.0251   5.2535   5.0251   1.2811        0
>> info.frac' % =   0.3444   0.5719
>> sum(info.frac) % =   0.9164   ???
```

There is margin b=[18 18]' with energy is $\in \mathcal{S}(b)$

Find boundary point:

```
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMACmimo(H8, Lxu,
18*b+.2*[8 8]', (8/9)*energy, 1)
admMACMIMO: completed in 0.088 s (FEAS_FLAG=2, iters=1)

FEAS_FLAG % =    2   (fails, =0 when larger than .2*[8 8])
bu_a =   19.6000   19.6000
>> sum(Eun,2) % =   7.1111   7.1111
```

Margin is .2*3 = 0.6 dB

64-tone Example (3 users)

```
>> [FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMACmimo(H64, [1 1 1], [410, 390,210], 64*[1 1 1], 1)
flag =     2
bu_achieved' =   410      390       210
Theta' =        1.1168    1.0231    1.0066
w =             0.7789    0.6372    0.6853
Info.   bu_v                         bun              order       frac          clusterID

    ___________________________   _____________   ________   _________   ____
    437.95    403.35    177.26    {3 x 64 double}   {[3 2 1]}    0.84932      1
    405.93     129.3    483.32    {3 x 64 double}   {[2 1 3]}   0.045211      1
    202.84    428.36    387.36    {3 x 64 double}   {[1 3 2]}   0.097064      1
sum(info.frac) % =    0.9916   (pretty close to 1, so we know this is near region boundary)
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMACmimo(H64, [1 1 1], [410, 390,210]+64*[.1 .1 .1], 64*[1 1 1], 1)
FEAS_FLAG = 0
```

- Design might ignore one of the vertices – do ¾ share, other 2.

```
>> [FEAS_FLAG, bu_a, info] = minPMACmimo(H64, [1 1 1], [410, 390, 210], [1 1 1], 1)
minPMACMIMO completed in 3.105 seconds, FEAS_FLAG=2
FEAS_FLAG =     2
bu_a =  409.9990  390.0011  210.0000
Info.Eu_avg: [0.9308 0.8794 1.0355]  - slight violation of power but close

[FEAS_FLAG, bu_a, info] = minPMACmimo(H64, [1 1 1], [410, 390, 210], [1 1 1.1], 1)
bu_a =  409.9990  390.0021  209.9990
info.Eu_avg =    0.9930    0.9508    0.9084
>> info.frac =     0.9626    0.0207    0.0167
```

- Solution is close to admMAC solution,
- and maybe just 1 vertex.

Design the GDFE

- Now back to tone-by-tone GDFE design

```
 >> Bu=zeros(64,3);
>> GU=zeros(3,3,64);
>> WU=zeros(3,3,64);
>> S0=zeros(3,3,64);
>> Wu=zeros(3,3,64);
>> MSWMFU=zeros(3,2,64);
Aopt=zeros(3,3,64);
Aopt=sqrt(cell2mat(info.Rxx));
for n=1:64
[ Bu(n,:) GU(:,:,n), WU(:,:,n), S0(:,:,n), MSWMFU(:,:,n)] = ...
mu_mac(H64(:,:,n), Aopt(:,:,n), [1 1 1] , cb);
end
>> GU(:,:,23) =
  1.0000 + 0.0000i  -1.6470 + 0.4247i  -0.3620 + 0.3662i
  0.0000 + 0.0000i   1.0000 + 0.0000i   0.7693 + 0.3998i
  0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
```

MAC Capacity Region

- Use admMAC in U-loop that gradually increases b_u's (for all $u = 1, \ldots, U$) until admMAC returns negative value and record the last rate pair (the last before that violation happens).
- This is the exterior of $\mathcal{C}_{MAC}(\boldsymbol{b})$.
- Also, can get margin from this $\Delta\tilde{b} \cdot 3$ dB.

In the field generally, the MAC (and its dual BC as we'll see in L15, 16) is believed to be non-convex difficult problem with all kinds of AI-machine-learning approaches.

Here in 379B, you can see it is convex, readily solved, and will clearly perform better as it is the optimum (canonical GDFE sense) solution.

Hopefully, you can help educate others on this seemingly global misunderstanding … so far.

End Lecture 14