Lecture 15

MAC Design by Weighted Sums

May 24, 2023

JOHN M. CIOFFI

Hitachi Professor Emeritus of Engineering

Instructor EE392AA – Spring 2023

Announcements & Agenda

- Announcements
  - PS7 - Final homework, due June 5.
  - Section 5.4 continued
  - admMAC improved in last 24 hours, so may be some minor tweaks yet

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

- Problem Set 7 = PS7 (due June 5 or 7)
  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

Maximum rate-sum solution (maxRMAC)

Section 5.4.4

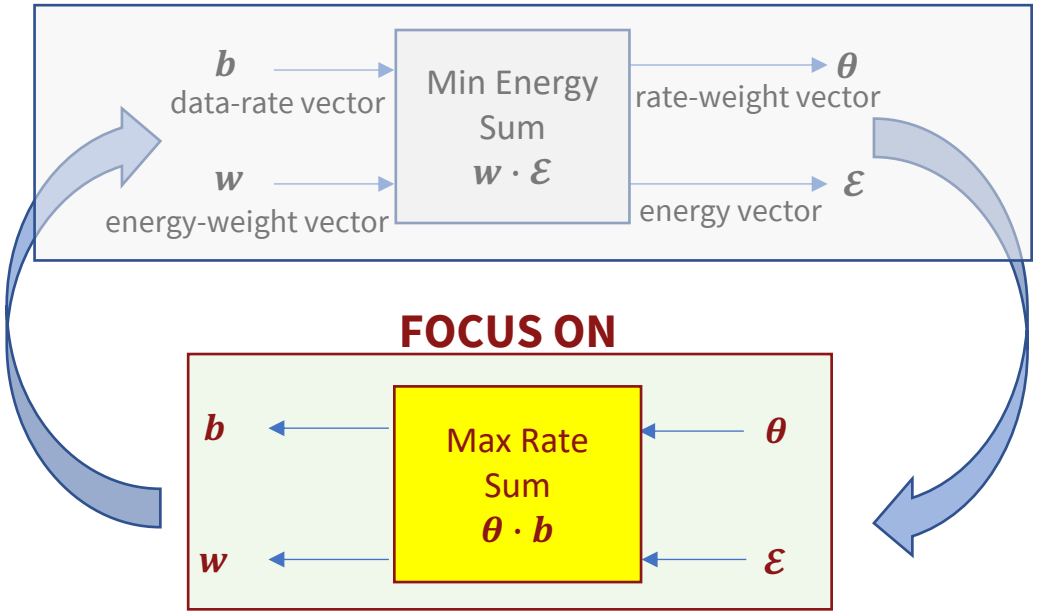
maximize weighted rate sum

- **Maximize weighted rate** sum b, given $\boldsymbol{\mathcal{E}}$
  - No rate limit
  - rate weights $\boldsymbol{\theta}$ given, non-negative

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

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

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



maxRMAC_cvx

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

H is the channel tensor $L_y \times U \times \overline{N}$
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)

There is also a no-cvx version,
No-cvx runs faster, but can deviate by 1-2% from CVX accuracy
The $\boldsymbol{w}$ weights often are scaled versions of one another

```
> H=zeros(1,2,1);
>> H(1,1,1)=80;
H(1,2,1)=60;
>> Eu=[1 1];
>> theta=[1 1];
>> cb=2;
>> [Eun, w, bun] = maxRMAC_cvx(H, Eu, theta,cb)

Eun =
   1.0000
   1.0000
w =
   0.6394
   0.3597
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' =1.2789   0.7194
>> 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

maxRMACMIMO

- Runs slower, but allows variable number of transmit antennas per user: Lxu (uses CVX)
- Also adds the 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 =
 11.6443
  0.6437
sum(bun) =  12.2880
```

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 =
   5.0000
   6.0000
W' =   0.3881   0.3275
bun =
   7.6484
   5.8337
>> Rxxs{:,:}
=   3.5983   2.2458
    2.2458   1.4017
=   1.6863   2.6971
    2.6971   4.3137
```

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 =
   7.5867
   4.1392
>> Rxxs{:,:}
 =
   3.9149   2.0611
   2.0611   1.0851
 =
   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' =   0.6396   0.3598
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
w' =   0.6396   0.3598
bun =
   6.3220   6.3220
   0.3219   0.3219
>>sum(bun,'all') =  13.2879
>>[Rxx, bsum , bsum_lin] = SWF([1 1], H4(:,:,1:2), [1 1],ones(1,1,2), 2);
Rxx(:,:,1) =              Rxx(:,:,2) =
    1    0                    1    0
    0    1                    0    1
bsum =  13.2879
bsum_lin =  2.1174
```

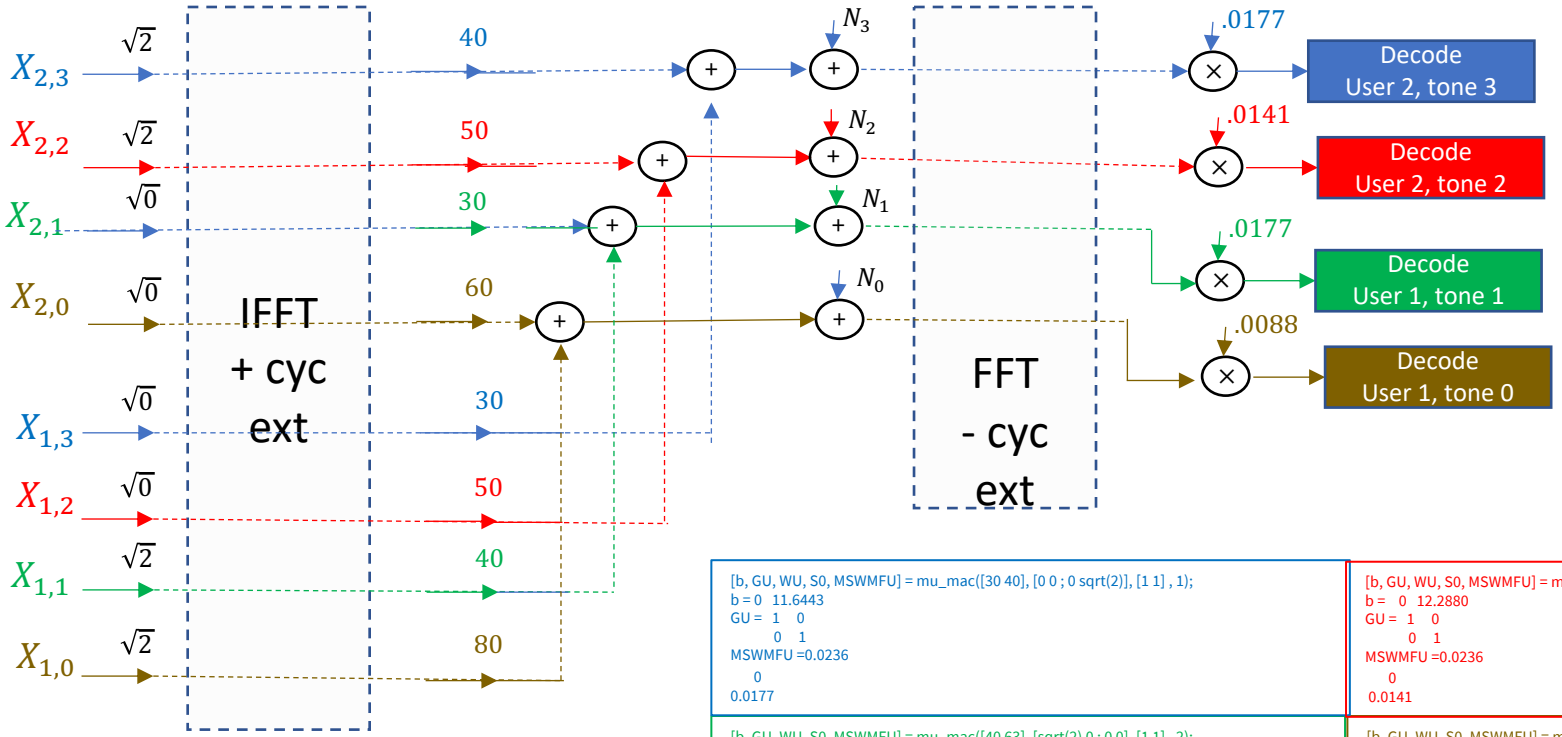
Linear substantially less– why?

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

Eun = 1.9984   1.9978   0.0037   0.0000
      0.0000   0.0000   1.9980   2.0020

w' =   0.5004   0.4995

bun =  13.6428  11.6427   3.3711   0.0023
       0.0000   0.0000   8.9181  11.6435

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

- FDM solution simplifies to linear receiver as optimum

```
[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
```

```
[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
```

```
[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
```

```
[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
```

maxRESMAC – only energy-sum constraint

- function [Eun, w, bun] = maxRESMAC(H, Etotal, theta, cb)
  - inputs

H is the channel tensor $L_y \times U \times \overline{N}$
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(H4x, 8, [ 1 1]',1)

Eun =
   2.0000   1.9996  -0.0000   0.0000
   0.0000   0.0000   2.0007   1.9997
w =   0.4998   0.4998
bun =
  13.6439  11.6440  -0.0000   0.0005
   0.0000   0.0000  12.2885  11.6436
>> sum(bun,'all') =  49.2206
>> sum(Eun,'all') =   8.0000
>> sum(bun') = 25.2884  23.9322

>> [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
```

- Also [Rxxs, Eun, w, bun] = maxRESMACMIMO(H, Lxu, Etotal, theta, cb)

Minimum energy-sum solution (minPMAC)

Section 5.4.4.5

minimize weighted energy sum

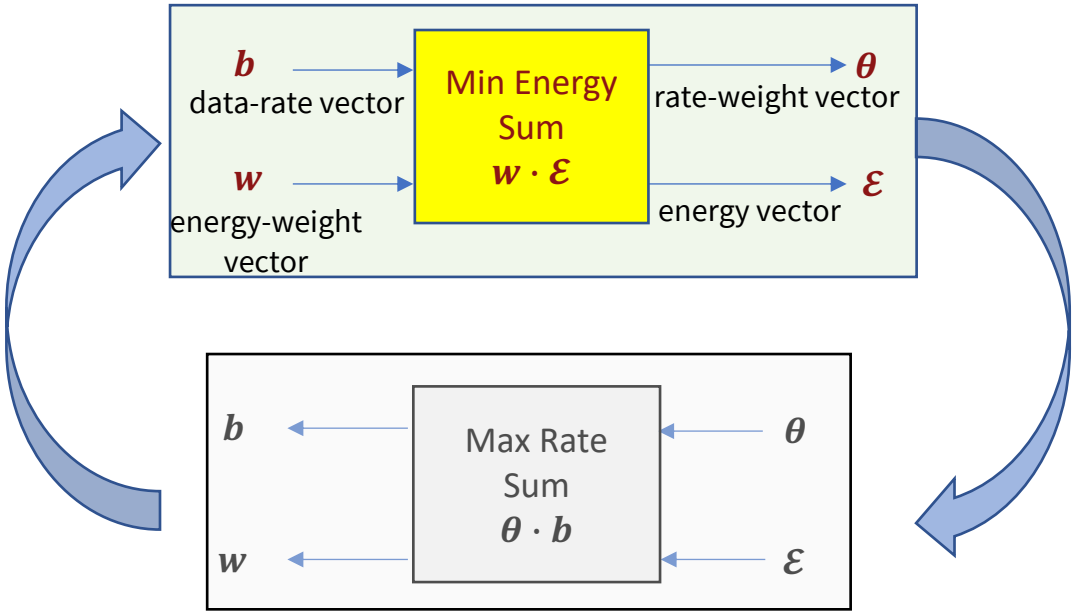
- **Minimize weighted energy** sum $\mathcal{E}$, given $\boldsymbol{b}$
  - No rate limit
  - energy weights $\boldsymbol{w}$ given, non-negative

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

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

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

FOCUS ON



minPMAC

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

H is the channel tensor $L_y \times U \times \bar{N}$ **Freq Domain**
bu is the given data-rate $\boldsymbol{b}$
w is the energy weight vector $\boldsymbol{w}$
cb =1 cplx, =2 real

- outputs

Eun is the $U \times \bar{N}$ matrix containing $\mathcal{E}_{u,n}$
(does not handle MAC $L_x > 1$)
theta is the order/rate vector $\boldsymbol{\theta}$
bun is the $U \times \bar{N}$ bit matrix containing $b_{u,n}$
FEAS_FLAG = 1 (one order), 2 vertex sharing
bu_a is user rate vector
Info is table of information (see examples)

- Called subroutines (6)
  - minPtone, Lag_dual_f, eval_f, hessian, fm_waterfill_gn , start_Ellipse
  - Now at web site

```
>> 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];
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H, b, w, 2);
Eun =
   0.6300
   0.0175
theta =
   1.2800
   1.2956
bun =
   3.0000
   3.0000
> info.order{:} =    1    2
>> 0.5*log2(1+(6400*Eun(1))/(1+3600*Eun(2))) = 3.0000
>> 0.5*log2(1+(3600*Eun(2))) = 3.0000
```

Decode user 1 last (best position or top)

Increase N to 4 tones

```
>> H=zeros(1,2,4);
>> H(1,1,:)=80 ;
>> H(1,2,:)=60 ;
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H, 4*b', w', 2)
Eun =
   0.6300   0.6300   0.6300   0.6300
   0.0175   0.0175   0.0175   0.0175
theta =
   1.2800
   1.2956
bun =
   3.0000   3.0000   3.0000   3.0000
   3.0000   3.0000   3.0000   3.0000
FEAS_FLAG =    1
bu_a' =   12.0001   12.0000
info.order{:} =    1    2
```

Real bb, so ignore Upper half tones

```
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H, 5*b', w', 2)
Eun =
   5.0917   5.0917   5.0917   5.0917
   0.0500   0.0500   0.0500   0.0500
theta =
  10.2400
  10.2840
bun =
   3.7500   3.7500   3.7500   3.7500
   3.7500   3.7500   3.7500   3.7500
FEAS_FLAG =    1
bu_a' =  15.0000  15.0000
info.order{:} =    1    2
```

result is correct, but energy too high, perhaps

- Unequal gains on tones

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

No conjugate symmetry, Must be BB complex

```
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H4x 8*b', w', 1)
Eun =
   1.8098   1.8094   0.0000   0.0000
   0.0003   0.0000   1.8097   1.8097
theta =
   3.6201
   3.6206
bun =
  12.5005  11.4993   0.0000   0.0000
   0.9995   0.0006  11.5001  11.5001
FEAS_FLAG =    1
bu_a' =  23.9998  24.0002
```

FDM showing again

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

Equal Thetas

```
Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC([80 80], b', w', 2)
Eun =
   0.3200
   0.3199
theta =
   1.2800
   1.2800
bun =
   0.4999
   5.5001
FEAS_FLAG =    2
bu_a =
   3.0000
   3.0000
info =  2x6 table
 bu_v              Eun      bun      theta    order      frac     clusterID
0.49971   5.5003  {1×2}    {1×2}    {1×2}    {1×2}    0.49998      1
 5.5001   0.49994 {1×2}    {1×2}    {1×2}    {1×2}    0.50002      1
>> info.bu_v'*info.frac  =
   3.0000
   3.0000
>> info.order{:} =
     1     2
     2     1
```

Not the desired *b*

Order switches

```
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC([80 80 80], [5 4 3], [1 1 1], 2)
Eun = [873.8222   873.8105   873.8076]
Theta' =  1.0e+03 * [5.2429   5.2429   5.2429 ]
bun =
   0.5000
   0.2925
  11.2075
FEAS_FLAG =    2
bu_a =  5.0000   4.0000   3.0000
 3×7 table

         bu_v            Eun        bun       theta      order      frac     clusterID
 0.5     11.208   0.29248  {1×3}    {1×3}    {1×3}    {1×3}    0.33511      1
11.208   0.29248  0.5      {1×3}    {1×3}    {1×3}    {1×3}    0.42492      1
0.29248  0.5      11.208   {1×3}    {1×3}    {1×3}    {1×3}    0.23998      1

>> info.bu_v'*info.frac =  5.0000   4.0000   3.0000
>> info.order{:} =
     3     1     2
     2     3     1
     1     2     3
```

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

Split vertices 17/50, 21/50, 12/50 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 ];
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(Htemp, [7 5 6], [1 1 1], 1);
Eun =    0.0560   0.0891   0.0967
theta' = 0.1968   0.1968   0.2892

>> bu_a' =          7.0000   5.0000   6.0000
MAC User ID           3        2        1          (not same as Matlab's index)
Matlab ID             1        2        3
FEAS_FLAG =    2
bu_a' =            7.0000   5.0000   6.0000
info =  2x6 table
        bu_v              Eun             bun          theta          order          frac     clusterID

   5.8732   6.1269      6   {1×3 dble}   {1×3 dble}   {1×3 dble}   {1×3 dble}   0.58514      1
   8.5895   3.4106      6   {1×3 dble}   {1×3 dble}   {1×3 dble}   {1×3 dble}   0.41485      1

>>> info.bu_v'*info.frac =   7.0000   5.0000   6.0000

>> info.order{:} =
   1   2   3
   2   1   3
```

The text in Section 5.4.4.4 continues This example to 4 tones

Also shows at lower data rates, the vertex sharing goes away with N>1, Because effectively "tone-sharing"

58/42 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

```
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC([80 60 80 60], [3 4 5 2], [1 1 1 1], 2)

Eun' =     1.0e+04 *   0.0001   0.0001   2.0973   2.0970
theta' =   1.0e+04 *   8.3886   8.3887   8.3886   8.3887
FEAS_FLAG =    2
>> bu_a' =          3.0000   4.0000   5.0000   2.0000
MAC User ID              4        3        2        1        (not same as Matlab's index)
Matlab ID                1        2        3        4
(Cluster 2 is at top – and cancels Cluster 1;  Cluster 1 treats Cluster 2 as noise)
info = 4 × 6 table
                    bu_v                          Eun          bun          theta        order   frac      clusterID
   7.5001   5.5004   0.49993   0.49964   {1x4 dble}   {1x4 dble}   {1x4 dble}   {1x4}   0.69995   1
   7.5001   0.5000   0.49993   5.5       {1x4 dble}   {1x4 dble}   {1x4 dble}   {1x4}   0.30005   1
   7.5001   5.5004   0.49993   0.49964   {1x4 dble}   {1x4 dble}   {1x4 dble}   {1x4}   0.35714   2
   0.5000   5.5004   7.5000    0.49964   {1x4 dble}   {1x4 dble}   {1x4 dble}   {1x4}   0.64286   2
>> info.frac(3:4)'*info.bu_v(3:4,[1,3]) =   3   5
>> info.frac(1:2)'*info.bu_v(1:2,[2,4]) =   4   2
>> info.order{:} =
   3   1   4   2
   3   1   2   4
   3   1   4   2
   1   3   4   2
```

Program with # clusters > 1
Reverses order within clusters
(ok for 1 cluster, no reversal)

Cluster 2 in GDFE receiver has
User 4 with $\langle b_4 \rangle =$
$7.5 \times .643 + 0.5 \times .357 = 3$
User 2 with $\langle b_2 \rangle =$
$0.5 \times .643 + 7.5 \times .357 = 5$
Users 3 and 1 both cancelled

Cluster 1 in GDFE receiver has
User 3 with $\langle b_3 \rangle =$
$5.5 \times .30 + 0.5 \times .70 = 4$
User 1 with $\langle b_1 \rangle =$
$0.5 \times .30 + 0.5 \times .70 = 2$
Treats users 1 and 2 as noise

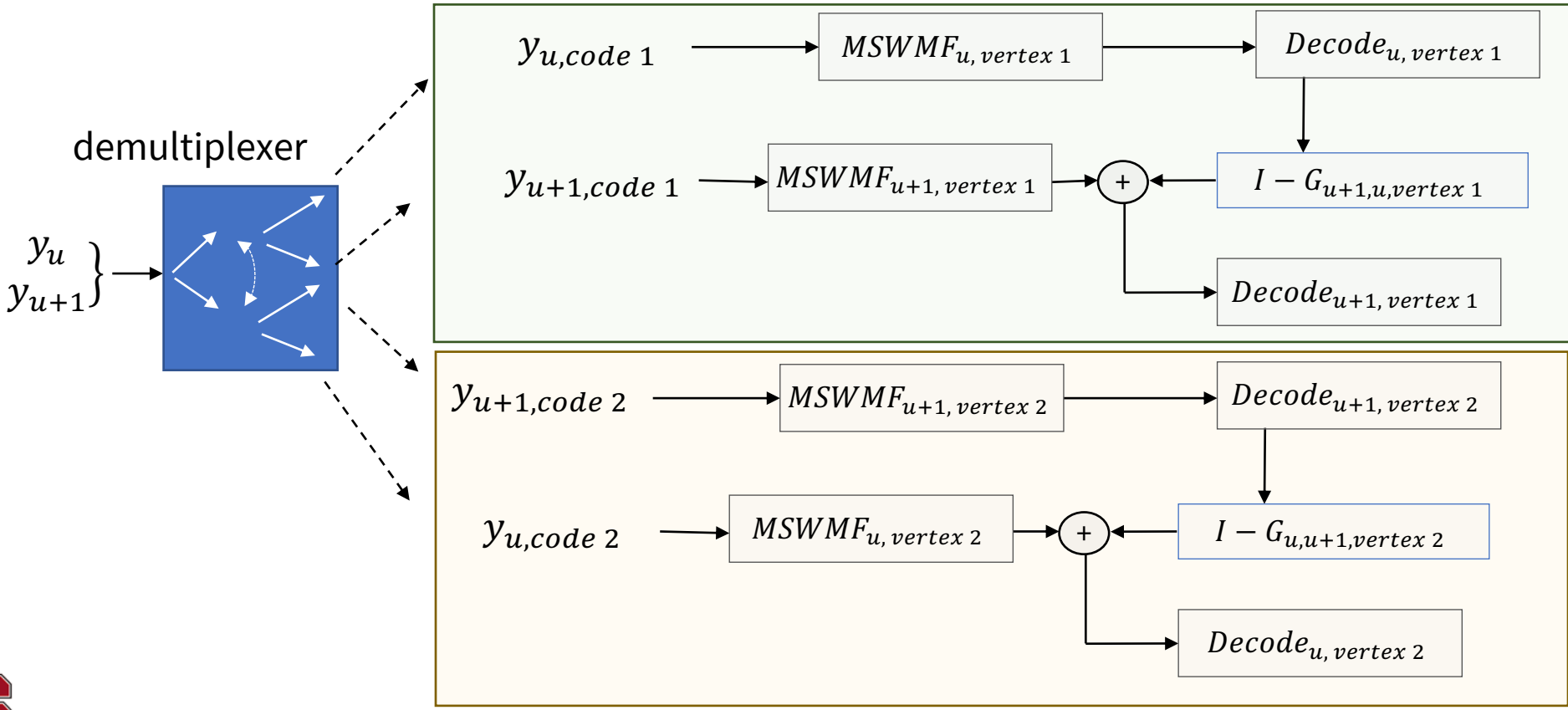
Encoder occurrence frequencies?

cluster 2 \ cluster 1	0.7	0.3
0.36	.192	.108
0.64	.448	.252

Each user has 4 encoders/decoders with roughly 4, 9, 5, and 2 turns from each 20 symbols transmitted in larger frame

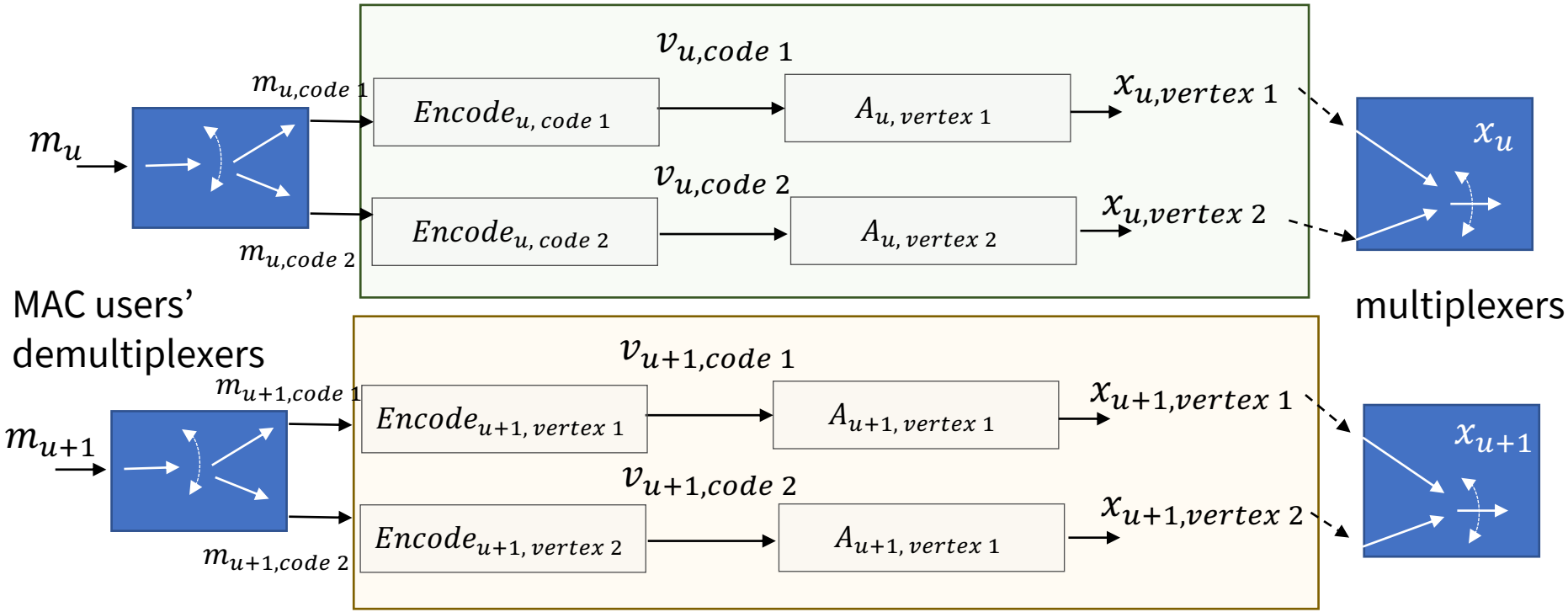
illustrate cluster receiver

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



illustrate cluster encoder modulator

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



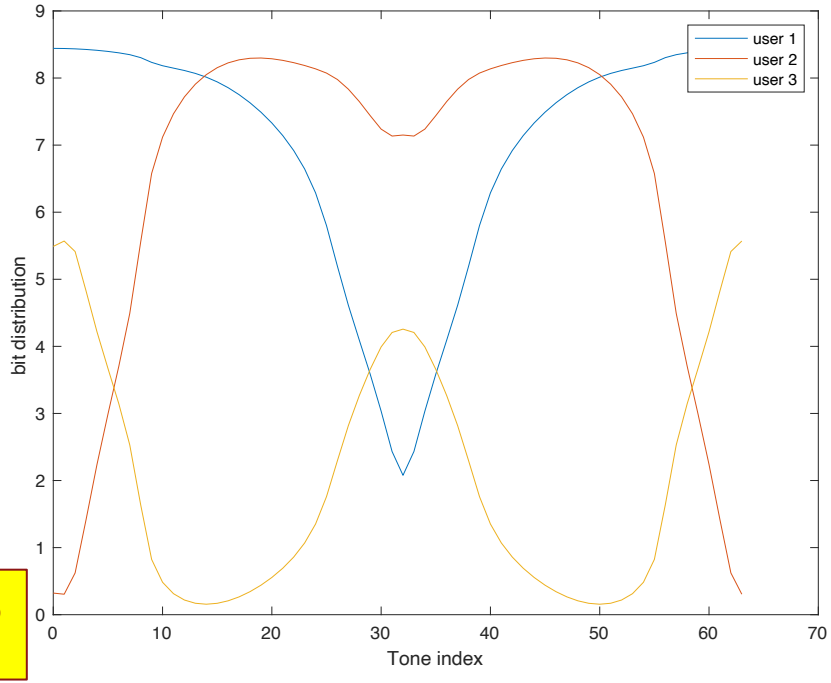
Multicluster imposes frame synchronization so that each user encoder at any symbol time corresponds to only encoder in each and every other user's set

64-tone example

```
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H64, [445 412 132]', [1 1 1]',1);
Elapsed time is 215.045054 seconds.

>> theta =
   2.6917
   2.6917
   2.2404
FEAS_FLAG =    2
bu_a =

  445.0000
  412.0000
  132.0000

      % bu_v              Eun        bun       theta    order  frac    clusterID

   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
 >>sum(Eun') = 65.9553   60.2757   40.4453
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H64, [445 412 132]', [3 1
.75]',1);
>> sum(Eun')  =  61.0398  58.6166  49.6402
>> bu_a =  445.0000  412.0000  132.0001
```

The GDFE's energize all 3 users, but some "FDM"

- 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)



Corresponding MAC Design (N = 8), HWH7

```
N=8;
U=3;
Ly=2;
cb=1;
Lxu=[1 1 1];
bsum=zeros(1,N);
H8 = fft(h, N, 3);
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H8, [54 51 16]', [1 1
1]',1)

GU=zeros(U,U,N);
WU=zeros(U,U,N);
S0=zeros(U,U,N);
Bu=zeros(U,N);
MSWMFU=zeros(U,Ly,N);
AU=zeros(3,3,N);
 for n=1:N
  AU(:,:,n)=sqrtm(diag(Eun(:,n)));
 end
 for n=1:N
  [Bu(:,n), GU(:,:,n) , WU(:,:,n),S0(:,:,n), MSWMFU(:,:,n)] = ...
   mu_mac(H(:,:,n), AU(:,:,n), Lxu, cb);
 end
bvec=sum(Bu');
Bsum(index) = sum(bvec);

>> FEAS_FLAG =    1
 >> bvec =   54.7544   51.8108   16.1050
>> bu_a =   54.0000   51.0000   16.0000
```

Design commands, Divide by 8

Did not take 8/9 of energy because that is outside the minPMAC process

But Eun x 9/8 is actual energy

8 Feedback Sections

```
>> GU
GU(:,:,1) =   1.0e+04 *
  0.0001 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0000 + 0.0000i   0.0001 + 0.0000i  -1.7799 + 0.0000i
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0001 + 0.0000i
GU(:,:,2) =
  1.0000 + 0.0000i  -0.2004 + 0.0132i   0.1823 + 0.2030i
  0.0000 + 0.0000i   1.0000 + 0.0000i   1.3125 - 0.1839i
  0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,3) =
  1.0000 + 0.0000i  -0.8690 + 0.1232i  -0.0108 + 0.0889i
  0.0000 + 0.0000i   1.0000 + 0.0000i   0.1887 + 0.0287i
  0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,4) =
  1.0000 + 0.0000i  -1.7737 + 0.5843i  -0.4677 + 0.3006i
  0.0000 + 0.0000i   1.0000 + 0.0000i   0.8075 + 0.5128i
  0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,5) =   1.0e+03 *
  0.0010 + 0.0000i  -1.7618 + 0.0000i  -9.6891 + 0.0000i
  0.0000 + 0.0000i   0.0010 + 0.0000i   0.0024 + 0.0000i
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0010 + 0.0000i
GU(:,:,6) =
  1.0000 + 0.0000i  -1.7737 - 0.5843i  -0.4677 - 0.3006i
  0.0000 + 0.0000i   1.0000 + 0.0000i   0.8075 - 0.5128i
  0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,7) =
  1.0000 + 0.0000i  -0.8690 - 0.1232i  -0.0108 - 0.0889i
  0.0000 + 0.0000i   1.0000 + 0.0000i   0.1887 - 0.0287i
  0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,8) =
  1.0000 + 0.0000i  -0.2004 - 0.0132i   0.1823 - 0.2030i
  0.0000 + 0.0000i   1.0000 + 0.0000i   1.3125 + 0.1839i
  0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
```

8 Feedforward Sections

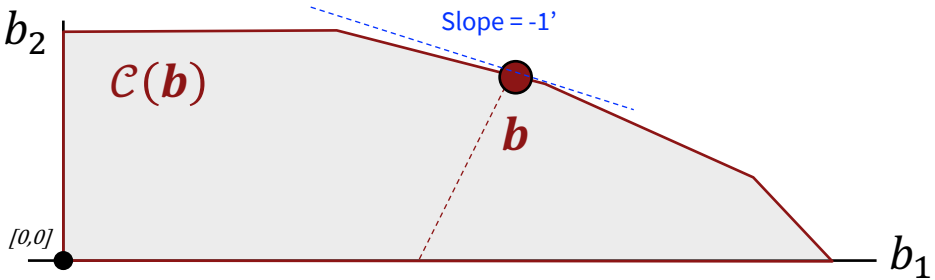
```
>> MSWMFU
MSWMFU(:,:,1) =   1.0e+03 *
  0.0000 + 0.0000i   0.0000 + 0.0000i
 -2.2006 + 0.0000i   4.6158 + 0.0000i
  0.0001 + 0.0000i  -0.0003 + 0.0000i
MSWMFU(:,:,2) =
  0.0397 + 0.0155i   0.0086 + 0.0183i
  0.0584 + 0.0483i  -0.0191 - 0.1594i
  0.0481 + 0.0390i   0.0017 - 0.1107i
MSWMFU(:,:,3) =
  0.0406 + 0.0365i  -0.0162 + 0.0203i
  0.0135 + 0.0147i   0.0294 - 0.0304i
  0.2279 + 0.0713i   0.0673 - 0.1243i
MSWMFU(:,:,4) =
  0.0666 + 0.1166i  -0.0648 - 0.0085i
  0.0123 + 0.0189i   0.0479 + 0.0042i
  0.0819 + 0.0380i   0.0158 - 0.0243i
MSWMFU(:,:,5) =   1.0e+02 *
  7.7990 + 0.0000i  -7.7990 + 0.0000i
  0.0006 + 0.0000i   0.0009 + 0.0000i
 -0.0013 + 0.0000i   0.0010 + 0.0000i
MSWMFU(:,:,6) =
  0.0666 - 0.1166i  -0.0648 + 0.0085i
  0.0123 - 0.0189i   0.0479 - 0.0042i
  0.0819 - 0.0380i   0.0158 + 0.0243i
MSWMFU(:,:,7) =
  0.0406 - 0.0365i  -0.0162 - 0.0203i
  0.0135 - 0.0147i   0.0294 + 0.0304i
  0.2279 - 0.0713i   0.0673 + 0.1243i
MSWMFU(:,:,8) =
  0.0397 - 0.0155i   0.0086 - 0.0183i
  0.0584 - 0.0483i  -0.0191 + 0.1594i
  0.0481 - 0.0390i   0.0017 + 0.1107i
```

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

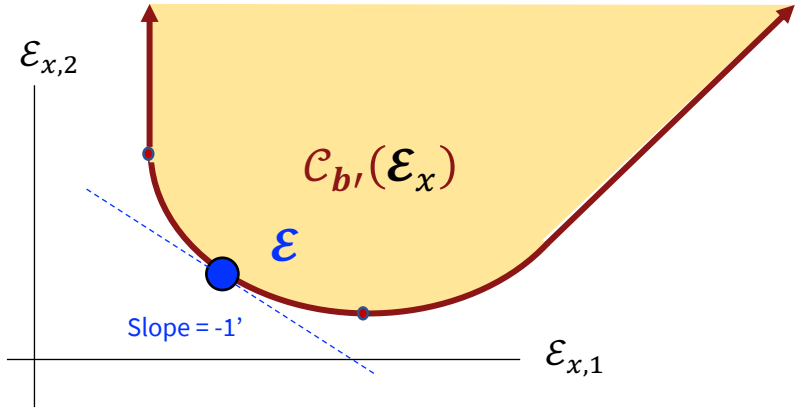
admMAC and admissibility

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 $\boldsymbol{b}$

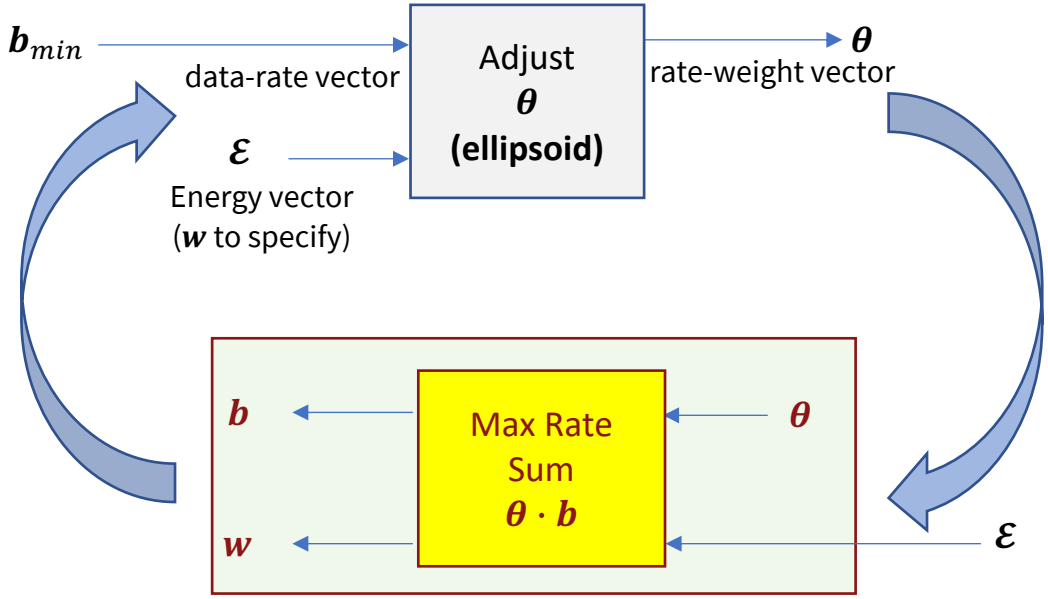


- $\mathcal{C}_{\boldsymbol{b}'}(\boldsymbol{\mathcal{E}}_x)$ contains all possible weighted **energy** sums $\sum_{u=1}^{U} w_u \cdot \mathcal{E}_u$ that meet **rate-vector** constraint rate vector $\preccurlyeq \boldsymbol{b}'$
- The min-E-sum point is "lowest" $\boldsymbol{\mathcal{E}}$ in $\mathcal{C}_{\boldsymbol{b}'}(\boldsymbol{\mathcal{E}})$
- 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 $\boldsymbol{b}_{min}$ and $\boldsymbol{\mathcal{E}}$
- Find the weights $\boldsymbol{w}$ **and** $\boldsymbol{\theta}$
- When $\boldsymbol{b}_{min} \neq \boldsymbol{b}$, change $\boldsymbol{\theta}$
- If there is a solution in $\mathcal{C}(\boldsymbol{b})$
- This process finds it
  - Both are convex methods that improve at each step



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

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

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

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

admMAC program

function [FEAS_FLAG, bu_a, info] = admMAC(H, Lxu, bu, Eu, cb)

admMAC determines whether the target rate vector bu is
feasible for (noise-whitened) channel H and energy/symbol Eu via rate region

Inputs:

- H: Ly-by-Lx-by-N channel matrix. H(:,:,n) denotes the channel for the n-th tone.
- Lxu: number of transmit antennas of each user. It can be either a scalar or a length-U vector. If it is a scalar, every user has Lxu transmit antennas; otherwise user u has Lxu(u) transmit antennas.
- bu: target rate of each user, length-U vector.
- Eu: Energy/symbol on each user, length-U vector.
- cb=1 cplex bb, cb=2 for real baseband

Outputs:

- FEAS_FLAG: indicator of achievability. FEAS_FLAG=0 if the target is not achievable; FEAS_FLAG=1 if the target is achievable by a single ordering; FEAS_FLAG=2 if the target is achievable by time-sharing
- bu_a: U-by-1 vector showing achieved sum rate of each user.
- info: various length output depending on FEAS_FLAG
  - --if FEAS_FLAG=0: empty
  - --if FEAS_FLAG=1: 1-by-5 cell array containing {Rxxs, Eun, bun, theta, w} corresponds to the single vertex
  - --if FEAS_FLAG=2: v-by-6 cell array, with each row representing a time-shared vertex {Rxxs, Eun, bun, theta, w, frac}

info's row entries in detail (one row for each vertex shared

- Rxxs: U-by-N cell array containing Rxx(u,n)'s if Lxu is a length-U vector; or Lxu-by-Lxu-by-U-by-N tensor if Lxu is a scalar. If the rate target is infeasible, output 0.
- Eun: U-by-N matrix showing users' transmit energy on each tone. If infeasible, output 0.
- bun: U-by-N matrix showing users' rate on each tone. If infeasible, output 0.
- theta: U-by-1 Lagrangian multiplier w.r.t. target rates
- w: U-by-1 Lagrangian multiplier w.r.t. energy constraints
- order: U-by-1 user order
- frac: fraction of dimensions for each vertex in time share (FF =2 ONLY)

subroutines called (directly)
maxRMAC_cvx
maxRMACMIMO

- Both energy and rate are inputs for MAC
- FEAS_FLAG = 0 if not admissible
- Info similar to minPMAC

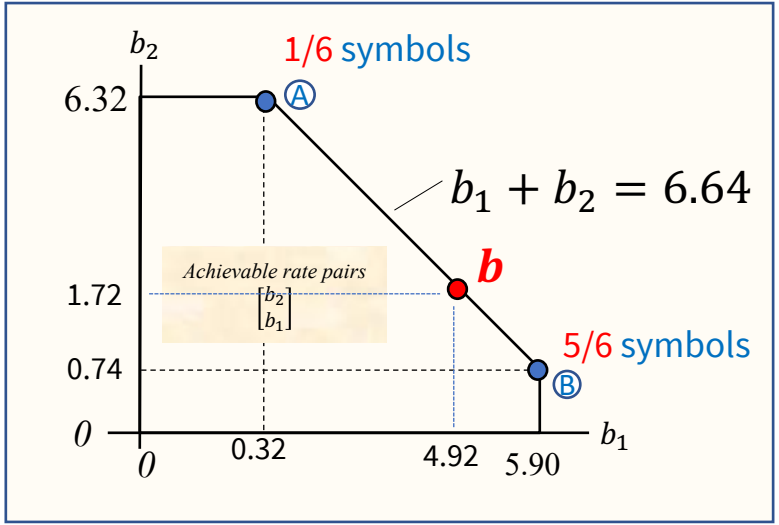
Back to simple channel, pick pt on boundary

```
H = [80 60];
>> bu_min = [1.72;4.92];
>> Eu = [1;1];
Lxu=[1 1];
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w,
info] = admMAC(H, Lxu, bu_min, Eu, cb);
flag =  2
bu_a =
   1.7210
   4.9229
info = 2x8 table
```

bu_v		Rxxs	Eun	bun	theta	order	frac	cID
6.322	.32189	{2x1}	{1x2 }	{1x2}	{1x2}	{1x2}	0.17621	1
.73684	5.9071	{2x1}	{1x2}	{1x2}	{1x2}	{1x2}	0.82379	1

```
>>Rxxs   %  2 × 1 cell array  same both vertices
   {[   1]}
  {[1.0000]}>> info.Eun{:,:}
> Eun =   % same both vertices
   1.0000
   1.0000
>> theta =
   0.9272
   0.917
>> w =
   0.5969
   0.3302
```

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




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

Confirms point is on boundary

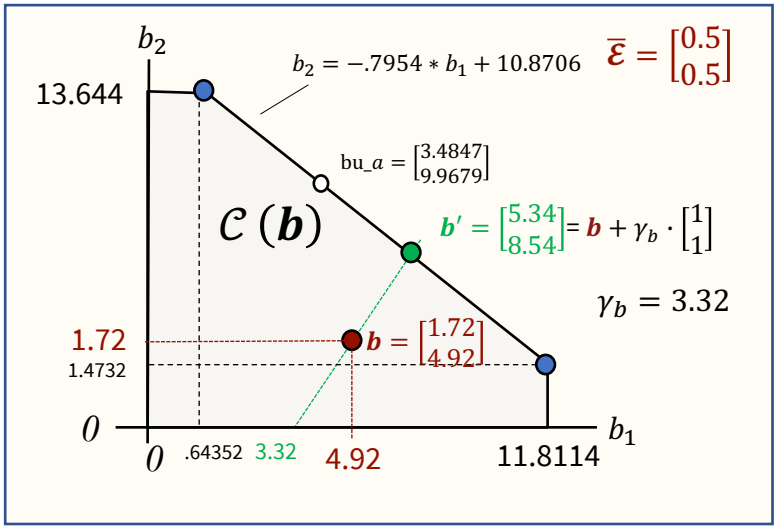
- FEAS_FLAG = 2, vertex sharing
  - One vertex has equal thetas is enough

Complex channel, same rate pt – NOT on boundary

```
>> [FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] =
admMAC(H, Lxu, bu_min, 2*Eu, 1);
flag =    2
>> bu_a =
   3.4847
   9.9679
>> info.bu_v =                 info.frac
   1.4732   11.814             . 83472
   13.644   .64352             . 16528
>> Eun =
   2.0000
   1.9989
```

- FEAS_FLAG = 2, vertex sharing

```
>>info.order{:} =
    1    2
    2    1
>> info.bu_v =
   1.4732   11.8142
  13.6440    0.6435
>> slope=(info.bu_v(1,1)-info.bu_v(1,2))/…
(info.bu_v(2,1)-info.bu_v(2,2)) =   -0.7954

>> int=info.bu_v(1,1)-slope*info.bu_v(1,2) =
   10.8706
>> rsum=sum(bu_a)=   13.4526
>> gammab=(rsum-bu_min(2)-bu_min(1))/2 =
        3.4063
```



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

```
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H, Lxu, bu_min +4*[1; 1], 2*Eu, 1)
flag =    0
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H, Lxu, bu_min +3.4*[1; 1], 2*Eu, 1)
flag =    0
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H, Lxu, bu_min +3.3*[1; 1], 2*Eu, 1)
flag =    2
bu_a' =   5.2878   8.6585
sum(bu_a) =   13.9464
```

```
>> [Eun, theta, bun, FEAS_FLAG, bu_a,
info]=minPMAC(H, bu_min, [1 1], 1)
Eun =
   0.0109   0.0081
theta =
   0.0312   0.0385
bun =
   1.7200   4.9200
FEAS_FLAG =    1
bu_a =
   1.7200
   4.9200
```

FEAS_FLAG = 1, single vertex
Lower sum energy
About 20 dB, so

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

Preferred Design

admMAC

- There can multi-solution vertex-sharing when in the interior of $\mathcal{C}(\boldsymbol{b})$
  - Slope (or hyperlane normal vector) is not necessarily = -1 $(\cdot\boldsymbol{1})$
- minPMAC can use any $\boldsymbol{w}$, including all equal (so energy sum), but does not guarantee a point in $\mathcal{C}_{\boldsymbol{b}}(\boldsymbol{\mathcal{E}})$
- minPMAC may be preferred design method as it tends to produce larger margin
  - Unless any user energy is too large
  - Then use admMAC , judiciously with the energies found – limiting any user energies that exceed allowed amounts
- If admMAC does not work, use it's $\boldsymbol{w}$ in minPMAC, try again the cycle
- If the first admMAC run does produces FEAS_FLAG = 0 , at least one user data rate is too high
- Could do similar cycle with maxRMAC, but it's hard to estimate a $\boldsymbol{\theta}$ that corresponds to $\boldsymbol{b}_{min}$

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)

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

```
[FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H,[1 1], 18*b,
(8/9)*energy,1)
FEAS_FLAG =     2
bu_a' = 19.6385          19.6385
info =  2x7 table
        bu_v          frac
    _________    _________
    21.698   17.579   0.5814
    16.778   22.499   0.4186
>> buntop=info.bun{1,:,:}; bunbot=info.bun{2,:,:};
>> reshape(buntop,2,8) =
 5.2089    4.9796    3.2601    0.0047    0.0000    0.0047    3.2601    4.9796
      0    0.0001    1.0629    5.0737    5.3058    5.0737    1.0629    0.0001
>> reshape(bunbot,2,8) =
5.2089    4.9768    0.8077    0.0001    0.0000    0.0001    0.8077    4.9768
      0    0.0028    3.5152    5.0783    5.3059    5.0783    3.5152    0.0028
>> Eun =
   1.8043    1.7937    0.8580    0.0011    0.0006    0.0011    0.8580    1.7937
   0.0005    0.0006    0.9443    1.7381    1.7447    1.7381    0.9443    0.0006
>> sum(Eun,2)' =    7.1105    7.1111
```

```
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H,18*b, [1 1]',1)
Eun =
   1.3437    1.3352    0.3090    0.0000    0.0000    0.0000    0.3090    1.3352
   0.0000    0.0000    0.9849    1.3020    1.3098    1.3020    0.9849    0.0000
theta =    2.7878
           2.7100
bun =
   4.7970    4.5693    2.0322    0.0000    0.0000    0.0000    2.0322    4.5693
        0    0.0000    1.8721    4.6758    4.9043    4.6758    1.8721    0.0000
FEAS_FLAG =     1
bu_a =   18.0000
         18.0000
>> sum(info.Eun{:,:},2)' = 4.6321    5.8835
>> 64/9 =    7.1111
>> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H,18*b+[1 ;1], [1
1]',1)
sum(info.Eun{:,:},2)' =    5.4922    7.1035
```

Margin is 3dB
For b=[18 18]'

64-tone Example (3 users)

```
>> [FEAS_FLAG, bu_a, Rxxs, Eun, theta, w, info] = admMAC(H64, [1 1 1], [410 390 210]', 64^2/67*[1 1 1], 1)
Elapsed time is 476.254513 seconds.    flag =     2
bu_achieved =  410.3922  390.3730  210.2009
info =  3×8 table
          bu_v                  Rxxs      Eun           bun          theta       order       frac      clusterID
    481.11    386.38    143.47  {1×64}  {1×3x64}  {1×3×64}  {1×3}  {1×3}  0.72759      1
    385.85    207.15    417.96  {1×64}  {1×3x64}  {1×3×64}  {1×3}  {1×3}  0.021875     1
    207.15    417.96    385.85  {1×64}  {1×3x64}  {1×3×64}  {1×3}  {1×3}  0.25054      1
>> info.bu_v'*info.frac =  410.3922  390.3730  210.2009
>> sum(cell2mat(info.Eun),3) =
   61.1343   61.1343   61.1343
   61.1343   61.1343   61.1343
   61.1343   61.1343   61.1343
```

```
[Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H,[410 390 210]', [1 1 1], 1);
theta =   2.8865   2.8865   2.8739
FEAS_FLAG = 2
bu_a =  410.0000  390.0000  210.0002
theta =   2.9830   2.9830   2.9830
info =  2x6 table
          bu_v                  Eun                      bun                      theta                frac          clusterID
    410.24    389.76    210  {1×3×64 double}  {1×3×64 double}  {1×3 double}  0.98818      1
    390.02    409.98    210  {1×3×64 double}  {1×3×64 double}  {1×3 double}  0.011823     1
>> Eun=info.Eun{1,:,:}; >> sum(Eun(1,:,:),3) =   59.5706   56.2791   66.2707
>> >> [Eun, theta, bun, FEAS_FLAG, bu_a, info]=minPMAC(H,[410 390 210]', [1.03 1 1.08], 1) bu_a' =  410
390  210
info =  2x7 table
          bu_v                  Eun          bun          theta       order       frac      clusterID
   410    394.28    205.72  {1×3×64}  {1×3×64}  {1×3}  {1×3}  0.98208      1
   410    155.42    444.58  {1×3×64}  {1×3×64}  {1×3}  {1×3}  0.017919     1

>> sum(Eun(1,:,:),3) = 60.9146   60.3152   61.0854
```

- might ignore one of the vertices – do ¾ share, other 2
- Pretty close to admMAC solution
- And maybe just 1 vertex

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 $\mathcal{C}_{MAC}(\boldsymbol{b})$
- Projects welcome in this area

End Lecture 15