Lecture 2

Channel Partitioning: Vector Coding & DMT

April 1, 2026

JOHN M. CIOFFI

Hitachi Professor Emeritus (recalled) of Engineering

Instructor EE379B – Spring 2026

Announcements & Agenda

- Announcements
  - Problem Set 1 due Wednesday, April 8 @ 18:00
  - Most relevant reading – Sections 2.5, 4.4-4.7
  - Syllabus fixed (mid 30, homework 30, final 40)

- Agenda
  - RA/MA water-fill flow charts (finish L1)
  - Vector Coding in Time-Frequency
    - $1+.9D^{-1}$ Vector-Code Example
  - DMT/OFDM partitioning
    - DMT Water-filling Software
    - Vector DMT/OFDM partitioning

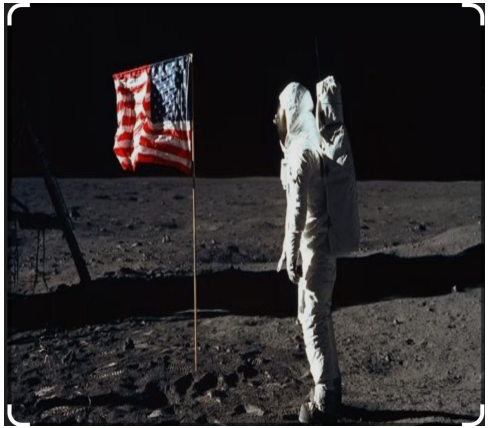
See Appendix D.1/2 and S3 video supplementary lecture

BEFORE L3

Even if you've seen Least Squares or MMSE before! (yes, even if you thought you understood it in 379A).

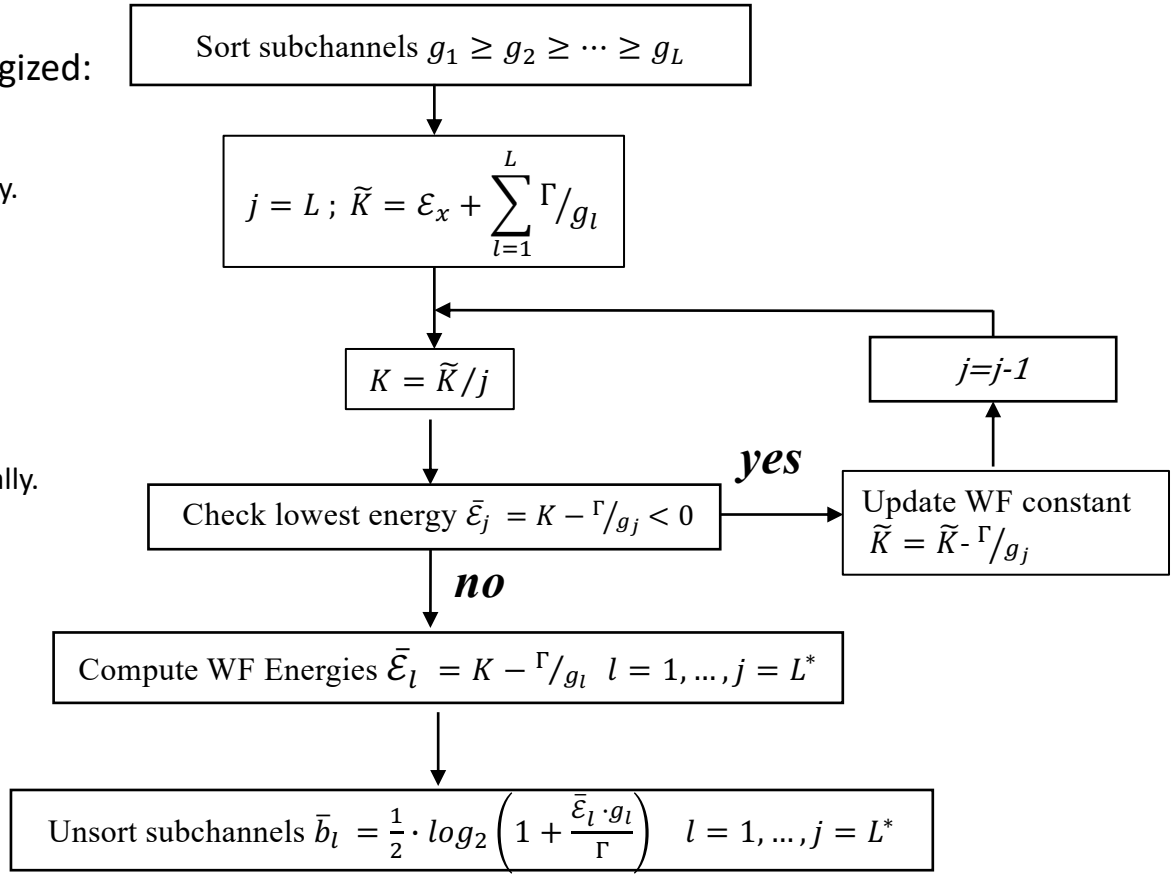
Question: Shannon conceived of water-filling, but who actually did it first? link

A. Collins: What else did he do and why important today?



RA Water-Fill Flow Chart

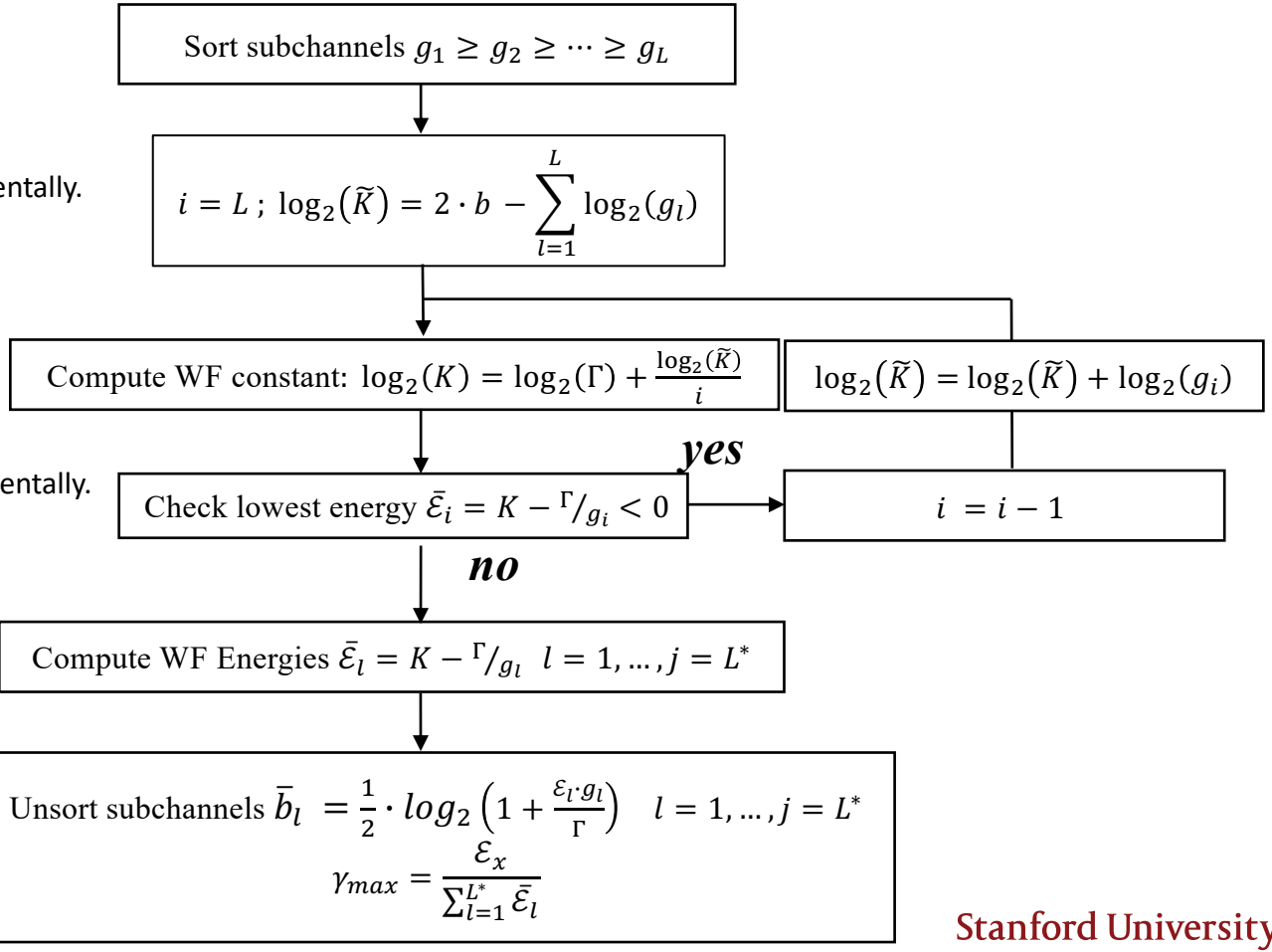
- RA WF may start with all channels energized:
  - computes K,
  - tests lowest energy, and
  - reduces number of dimensions incrementally.
- RA WF may initially energize 1 channel:
  - Computes K,
  - tests lowest energy, and
  - Increases number of dimensions incrementally.
- The sort is most complex part.
  - It can use pivots and bi-section,
  - which reduces average run time.



Margin Adaptive Flowchart

- MA WF similarly:
  - computes K,
  - tests lowest energy, and
  - reduces number of dimensions incrementally.

- MA WF avoids large products
  - Computes $\log(K)$ instead of K,
  - tests lowest energy, and
  - Increases number of dimensions incrementally.



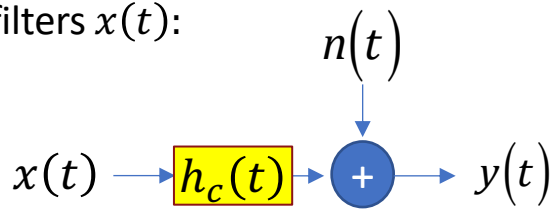
Vector Coding in Time/Frequency

Section 4.6.1

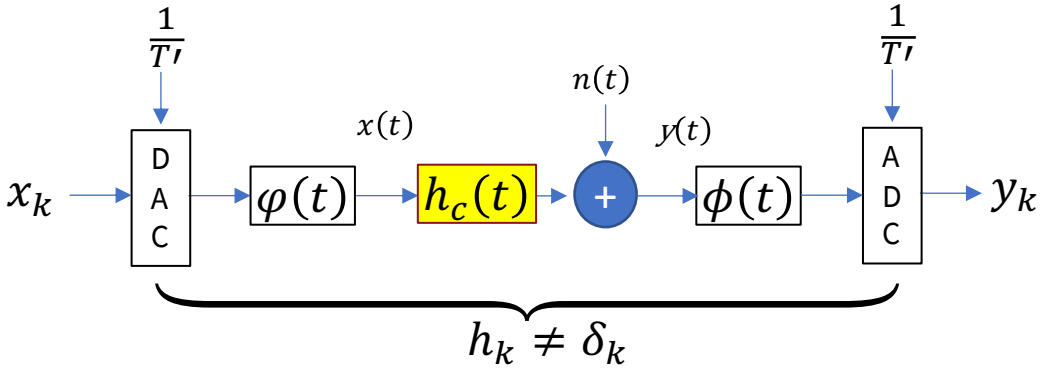
See PS1.5 (Prob 4.25) (matrix AWGN and vector coding)

Scalar Time/Frequency (filtered) AWGN Channel

- Ideal AWGN channels just add noise, but if it also filters $x(t)$:
  - attenuation,
  - band-limits, &
  - spectrally shaped noise (See Sec 1.3.7).



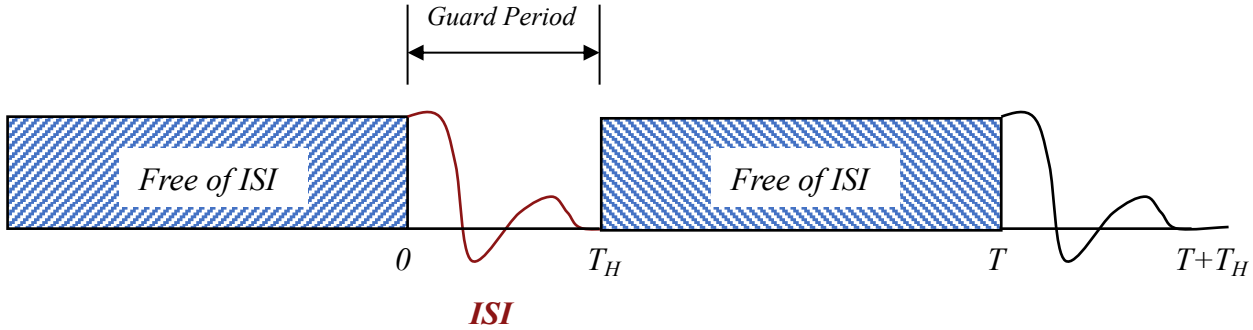
- The $h_c(t)$ causes interference between successive transmissions – complicates and changes performance;
  - see Chapter 3 in EE379A.
- Sampled equivalent has $T' < T$ as the **sample period** – generalizes 379A's "fractional spacing."



$$y_k = x_k * h_k + n_k$$

Guard Periods for frequency-time

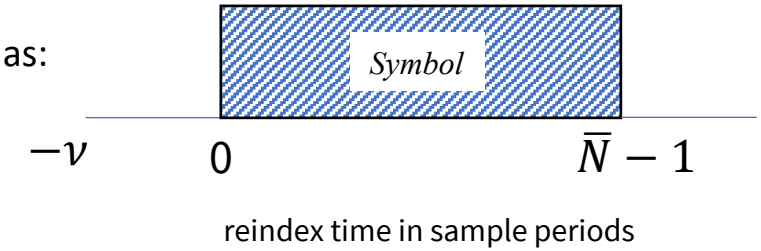
- The **guard period (GP)** T_H
  - lets ISI abate before next symbol.



- This wastes T_H/T of resources (time dimensions):
  - If $T >> T_H$, then the guard period may be worth it.
  - GP may be zeroed or anything the receiver ignores.

excess bandwidth $\alpha = \dfrac{T_H}{T - T_H}$

- A (scalar / SISO) symbol with $\bar{N}$ dimensions (samples) has:
  - $T_H = \nu \cdot T'$ so up to ν+1 non-zero samples/dimensions in h_k .
  - $\bar{N}$ works for real ($\bar{N} = 1$) or complex ($\bar{N} = 2$).



SISO (time-dimension) Case

- Simple scalar convolutional matrix channel with guard band

$$\begin{bmatrix} y_{\overline{N}-1} \\ y_{\overline{N}-2} \\ \vdots \\ y_0 \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \dots & h_\nu & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & h_0 & \ddots & h_{\nu-1} & h_\nu & \ddots & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 0 & 0 & h_0 & h_1 & \dots & h_\nu \end{bmatrix} \begin{bmatrix} x_{\overline{N}-1} \\ \vdots \\ x_0 \\ x_{-1} \\ \vdots \\ x_{-\nu} \end{bmatrix} + \begin{bmatrix} n_{\overline{N}-1} \\ n_{\overline{N}-2} \\ \vdots \\ n_0 \end{bmatrix}$$

Ignore -1:-ν

guard period — could be anything, including 0 or cyclic.

$$\boldsymbol{y} = H \boldsymbol{x} + \boldsymbol{n}$$

- Non-square shift "Toeplitz" matrix for convolution
  - More inputs than outputs when $\nu \neq 0$

SVD Again for the time-dimension case

- Singular Value Decomposition performs:

$$H = F \cdot \begin{bmatrix} \underbrace{\Lambda}_{\bar{N}\times\bar{N}} & \vdots & \underbrace{\mathbf{0}_{-1:-\nu}}_{\bar{N}\times\nu} \end{bmatrix} \cdot M^*$$

$N = \bar{N}$ if real subsymbols and $\widetilde{N} = 1$
$N = 2 \cdot \bar{N}$ if complex subsymbols and $\widetilde{N} = 2$

$$\underbrace{FF^* = F^*F = I}_{\bar{N}\times\bar{N}} \qquad \Lambda = \begin{bmatrix} \lambda_{\bar{N}-1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \lambda_1 & 0 \\ 0 & \cdots & 0 & \lambda_0 \end{bmatrix} \qquad \underbrace{MM^* = M^*M = I}_{(\bar{N}+\nu)\times(\bar{N}+\nu)}$$

unique (real) singular values ≥ 0

- The vector-coding input construction is:

$$\boldsymbol{x} = M \begin{bmatrix} \boldsymbol{X} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{m}_{\bar{N}-1} & \boldsymbol{m}_{\bar{N}-2} & \cdots & \boldsymbol{m}_1 & \boldsymbol{m}_0 & \cdots & \boldsymbol{m}_{-\nu} \end{bmatrix}}_{(\bar{N}+\nu)\times\bar{N}} \underbrace{\qquad}_{zeroed} \begin{bmatrix} X_{\bar{N}-1} \\ X_{\bar{N}-2} \\ \vdots \\ X_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \sum_{n=0}^{\bar{N}-1} X_n \cdot \boldsymbol{m}_n$$

see svd 'econ' option (matlab)

Vector-Coded Time Dimensions Only

- Channel output processed by matched vectors: $\boldsymbol{Y} = F^* \cdot \boldsymbol{y} = \begin{bmatrix} \boldsymbol{f}^*_{\bar{N}-1} \cdot \boldsymbol{y} \\ \vdots \\ \boldsymbol{f}^*_0 \cdot \boldsymbol{y} \end{bmatrix}$

$$\mathbb{E}[\boldsymbol{n} \cdot \boldsymbol{n}^*] = R_{\boldsymbol{nn}} = R_{\boldsymbol{nn}}^{1/2} \cdot R_{\boldsymbol{nn}}^{*/2}$$

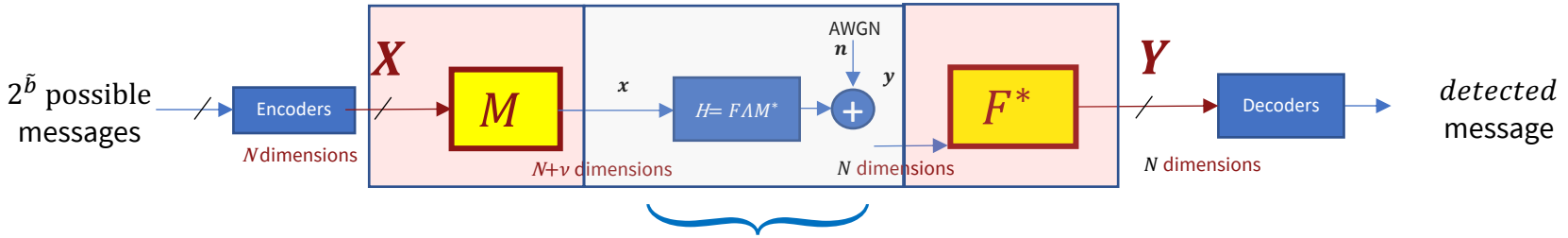
Noise-Equivalent Channel

$$\boldsymbol{y} \leftarrow R_{nn}^{-1/2}\boldsymbol{y} = \left(R_{nn}^{-1/2} \cdot H\right) \cdot \boldsymbol{x} + \tilde{\boldsymbol{n}}$$

Replaces H

Section 4.6.1.2

- Vector-coded channel partitioning is :



- Parallel Channels are then:

$$Y_n = \lambda_n \cdot X_n + N_n$$

$$SNR_n = \frac{\lambda_n^2 \cdot \bar{\mathcal{E}}_n}{\sigma^2}$$

Data Rates and Mutual Information

- For any energies and consequent SNR's
  - **If complex baseband, replace ν with 2ν** for $\bar{b}$ calculation.

$$\bar{b} = \frac{b}{N+\nu} = \frac{1}{2}\log_2\left(1+\frac{SNR_{VC}}{\Gamma}\right)$$

- When the gap = 0 dB, this is the "mutual information" , reliable $b \leq \mathbb{I}(\boldsymbol{x;y})$.
  - The **mutual information** bounds data rate, for (Gaussian) input with given autocorrelation R_{xx} , or any energy distribution (possibly not WF).
  - Good scalar-AWGN code applies "outside" the parallel channel set.

$$SNR_{VC} = 2^{2\cdot\bar{\mathbb{I}}} - 1$$

- Maximized SNR, and thus mutual information, occur when energy is water-filling → Capacity.

$$SNR_{VC,water-fill} = 2^{2\cdot\bar{c}} - 1$$

- Highest reliable data rate that can be transmitted (Shannon 1948):
  - **for the given block size $\bar{N}$ and guard period ν.**

$1+.9D^{-1}$ Vector-Code Example

Section 4.6

Matlab 1+.9D^{-1} example returns:

- Form H and do SVD:

```
>> C=[.9
zeros(7,1)];
>> R=[.9 1 zeros(1,7)];
```

```
>> H=toeplitz(C,R)

H =

  0.9000   1.0000        0        0        0        0        0        0        0
       0   0.9000   1.0000        0        0        0        0        0        0
       0        0   0.9000   1.0000        0        0        0        0        0
       0        0        0   0.9000   1.0000        0        0        0        0
       0        0        0        0   0.9000   1.0000        0        0        0
       0        0        0        0        0   0.9000   1.0000        0        0
       0        0        0        0        0        0   0.9000   1.0000        0
       0        0        0        0        0        0        0   0.9000   1.0000
```

```
>> [F,L,M]=svd(H)
F =
  -0.1612   0.3030  -0.4082   0.4642  -0.4642   0.4082   0.3030  -0.1612
  -0.3030   0.4642  -0.4082   0.1612   0.1612  -0.4082  -0.4642   0.3030
  -0.4082   0.4082   0.0000  -0.4082   0.4082   0.0000   0.4082  -0.4082
  -0.4642   0.1612   0.4082  -0.3030  -0.3030   0.4082  -0.1612   0.4642
  -0.4642  -0.1612   0.4082   0.3030  -0.3030  -0.4082  -0.1612  -0.4642
  -0.4082  -0.4082  -0.0000   0.4082   0.4082   0.0000   0.4082   0.4082
  -0.3030  -0.4642  -0.4082  -0.1612   0.1612   0.4082  -0.4642  -0.3030
  -0.1612  -0.3030  -0.4082  -0.4642  -0.4642  -0.4082   0.3030   0.1612
   f0                                                              f7

L =
   1.8712        0        0        0        0        0        0        0        0
        0   1.7857        0        0        0        0        0        0        0
        0        0   1.6462        0        0        0        0        0        0
        0        0        0   1.4569        0        0        0        0        0
        0        0        0        0   1.2237        0        0        0        0
        0        0        0        0        0   0.9539        0        0        0
        0        0        0        0        0        0   0.6566        0        0
        0        0        0        0        0        0        0   0.3443        0

M =
  -0.0775   0.1527  -0.2232   0.2868  -0.3414   0.3852   0.4153  -0.4214   0.4728
  -0.2319   0.4037  -0.4712   0.4182  -0.2608   0.0428  -0.1748   0.3238  -0.4255
  -0.3583   0.4657  -0.2480  -0.1415   0.4320  -0.4280  -0.1475  -0.1871   0.3830
  -0.4415   0.3099   0.2232  -0.4674   0.1108   0.3852   0.4008   0.0278  -0.3447
  -0.4714   0.0090   0.4712  -0.0208  -0.4705   0.0428  -0.4666   0.1348   0.3102
  -0.4445  -0.2960   0.2480   0.4602   0.0526  -0.4280   0.3140  -0.2812  -0.2792
  -0.3639  -0.4626  -0.2232   0.1806   0.4522   0.3852  -0.0146   0.3936   0.2513
  -0.2395  -0.4127  -0.4712  -0.3975  -0.2097   0.0428  -0.2917  -0.4586  -0.2261
  -0.0862  -0.1697  -0.2480  -0.3187  -0.3794  -0.4280   0.4615   0.4683   0.2035
   m0                                                              m7
```

Matlab continued

- Use singular values^2 for channel gains: SVs = [1.87 1.78 1.64 1.45 1.22 .95 .66 .34]
- The gains then are: $g_n = \frac{\lambda_n^2}{\sigma^2 (= .181)} = [19.3 \;\; 17.6 \;\; 15.0 \;\; 11.7 \;\; 8.3 \;\; 5.0 \;\; 2.4 \;\; 0.66]$
- Water-filling (RA) with 0 dB gap $K = \frac{1}{7} \cdot \left(9 + \sum_{n=0}^{6} \frac{\Gamma}{g_n} \right) = 1.43$
- Energies [1.38 1.37 1.36 1.34 1.30 1.23 1.01 0] $\mathcal{E}_n = K - \frac{\Gamma}{g_n}$
- SNRs [26.2 24.2 20.4 15.8 10.6 6.2 2.4 0] $\mathcal{E}_n \cdot g_n$

Not assigned, but might find Prob 4.17 interesting for look

Overall performance and rate

- Product SNR

$$SNR_{VC} = \left[\prod_{n=0}^{6}(SNR_n + 1)\right]^{1/9} - 1 = 6.46 = 8.1\ dB$$

- Rate = capacity

$$\bar{C}(N = 9) = \frac{1}{9} \cdot \sum_{n=0}^{6} \frac{1}{2} \cdot \log_2(1 + SNR_n) = 1.45 \text{ bits/dimension}$$

```
>> R=[.9 1 zeros(1,99)];
>> C=[.9 zeros(1,99)];
>> H=(1/sqrt(.181))*toeplitz(C,R);
>> [F,L,M]=svd(H);
>> g=diag(L).*diag(L);
```

```
>> K=(1/89)*(101+sum(ones(1,89)./g(1:89)'))
K = 1.3294
>> K-1/g(89) =  -0.0219

>> K=(1/88)*(101+sum(ones(1,88)./g(1:88)'))
K =  1.3292
>> K-1/g(88) = 0.1627

So N* = 88
```

```
>> E=K*ones(1,88)-ones(1,88)./g(1:88)';

>> snr=E.*g(1:88)';
>> b=(0.5/101)*(1/log(2))*sum(log(ones(1,88)+snr))

b =

   1.5360
```

$\bar{N} \to \infty$, then $\frac{\bar{N} + \nu}{\bar{N}} \to 1$, so no loss (max is 1.55)

DMT/OFDM Partitioning

Section 4.7.1-5

See PS1.3 (Prob 4.18)

Cyclic Extension

- Remember this convolution from vectoring coding?

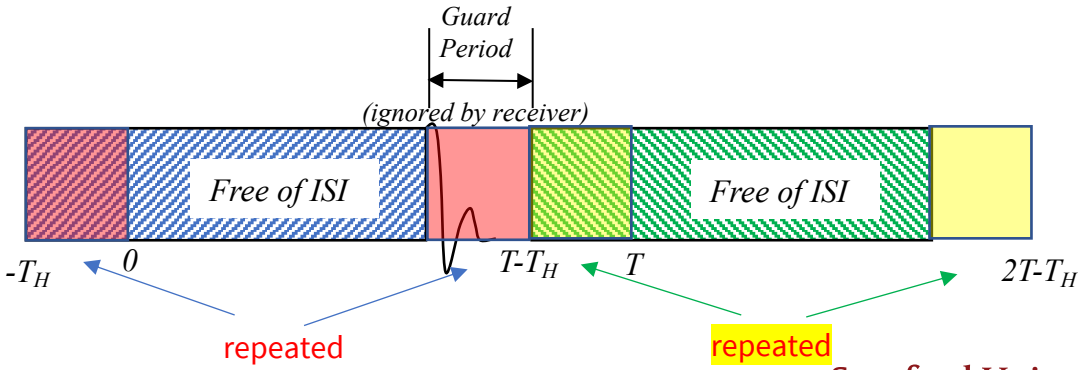
$$\begin{bmatrix} y_{\overline{N}-1} \\ y_{\overline{N}-2} \\ \vdots \\ y_0 \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \dots & h_\nu & 0 & \dots & 0 \\ 0 & h_0 & \ddots & h_{\nu-1} & h_\nu & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & h_0 & h_1 & \dots & h_\nu \end{bmatrix} \begin{bmatrix} x_{\overline{N}-1} \\ \vdots \\ x_0 \\ x_{-1} \\ \vdots \\ x_{-\nu} \end{bmatrix} + \begin{bmatrix} n_{\overline{N}-1} \\ n_{\overline{N}-2} \\ \vdots \\ n_0 \end{bmatrix}$$

guard period

Now it is cyclic

$$\boldsymbol{y} = \widetilde{H}\,\boldsymbol{x} + \boldsymbol{n}$$

- Let the guard period be such that $x_{-i} = x_{\overline{N}-i}$ for $i=1,\dots,\nu$ → **CYCLIC PREFIX** or **CYCLIC EXTENSION**.
  - **The channel now appears periodic!**



- "Toeplitz-distribution" limiting results use this cyclic-extension concept,
  $T = (\overline{N} + \nu) \cdot T'$.

Ideally $\overline{N} >> \nu$

So small loss of dimensions

Cyclic Convolution

- The matrix expression now uses an *N x N* circulant matrix:

$$\begin{bmatrix} y_{\overline{N}-1} \\ y_{\overline{N}-2} \\ \vdots \\ y_0 \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & \dots & h_\nu & 0 & \dots & 0 \\ 0 & h_0 & \ddots & h_{\nu-1} & h_\nu & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & h_0 & h_1 & \dots & h_\nu \\ h_\nu & 0 & \dots & 0 & h_0 & \dots & h_{\nu-1} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ h_1 & \dots & h_\nu & 0 & \dots & 0 & h_0 \end{bmatrix} \begin{bmatrix} x_{\overline{N}-1} \\ \vdots \\ x_0 \end{bmatrix} + \begin{bmatrix} n_{\overline{N}-1} \\ n_{\overline{N}-2} \\ \vdots \\ n_0 \end{bmatrix}$$

$\widetilde{H}$ is circulant $\overline{N} \times \overline{N}$

$$\boldsymbol{y} = \widetilde{H} \cdot \boldsymbol{x} + \boldsymbol{n}$$

- As far as output $\boldsymbol{y}$ is concerned, the input is periodic with the same period $\overline{N}$ as the output.
- The cyclic prefix is added for each and every symbol.
- ν dimensions are lost (both in terms of energy lost and no new information).

Cyclic Convolution and the DFT

- DFT = Discrete Fourier Transform
  - Normalized DFT maintains squared norm, energy from time ←→ frequency.
  - N or $\bar{N}$

$$X_n = \frac{1}{\sqrt{\bar{N}}} \cdot \sum_{k=0}^{\bar{N}-1} x_k \cdot e^{-j\frac{2\pi}{\bar{N}} \cdot kn} \quad \forall\, n \in [0 : \bar{N}-1]$$

$$\sum_{n=0}^{\bar{N}-1} |X_n|^2 = \sum_{k=0}^{\bar{N}-1} |x_k|^2$$

- IDFT = Inverse Discrete Fourier Transform
  - has symmetrical form

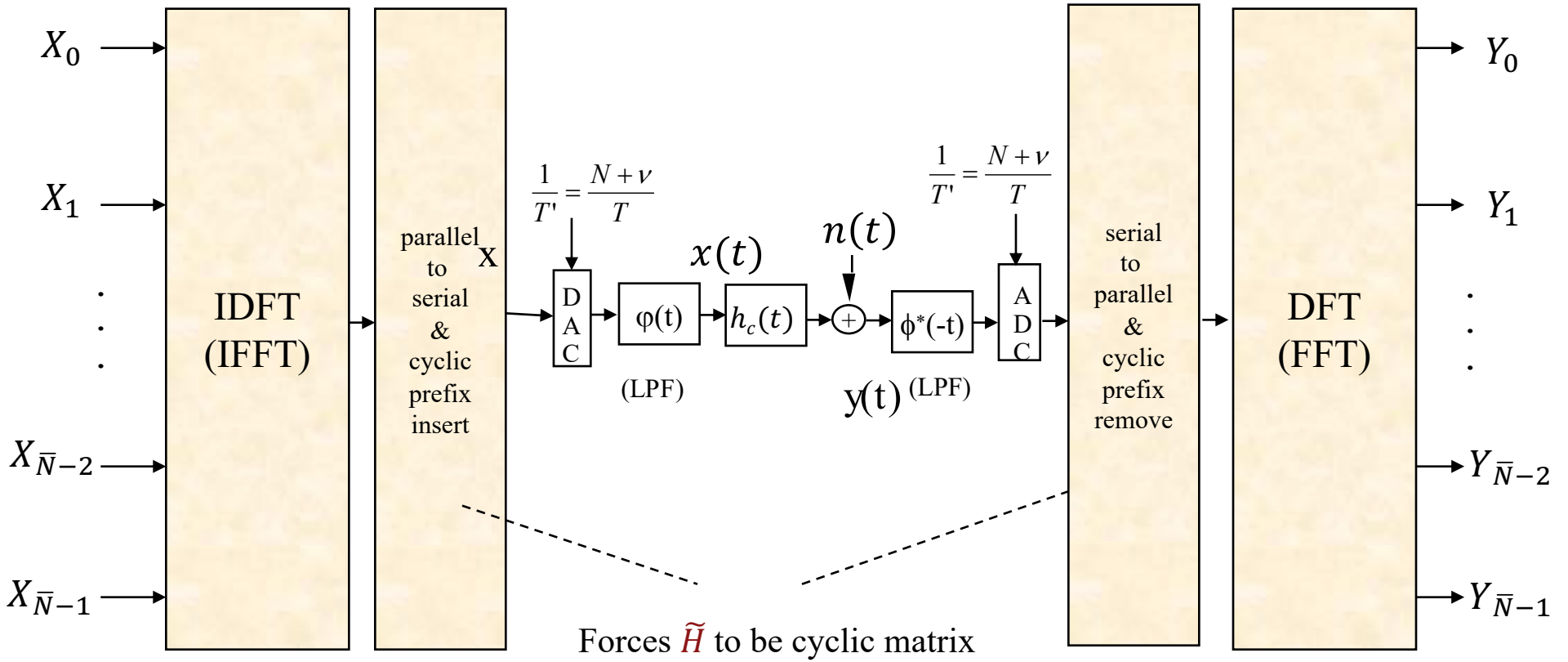
$$x_k = \frac{1}{\sqrt{\bar{N}}} \cdot \sum_{n=0}^{\bar{N}-1} X_n \cdot e^{+j\frac{2\pi}{\bar{N}} \cdot kn} \quad \forall\, k \in [0 : \bar{N}-1]$$

- Subsymbol channel $Y_n = \tilde{H}_n \cdot X_n \; (+ N_n)$
  - Vector coding with $M = F^* = Q$, but diagonal can be complex,
  - *and the guard period must be cyclic.*
  - This remains a parallel set of AWGN subchannels.

$$\tilde{H} = Q \cdot \Lambda \cdot Q^*$$

$$\Lambda = \begin{bmatrix} \tilde{H}_{N-1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \tilde{H}_0 \end{bmatrix}$$ = DFT values on diagonal

DMT / OFDM Transmission



$X_{\bar{N}-i} = X_i^*$ if $x(t)$ is real

Heavily used, wireline & wireless

Product SNR

- DMT uses 7 subchannels, DC plus 3 two-dimensional QAM subchannels, of a total of 9 dimensions.

$$\mathrm{SNR}_{DMT} = \left[\prod_{n=0}^{6}(1+\mathrm{SNR}_n)\right]^{1/9} - 1 = 7.6\ \mathrm{dB} \quad < SNR_{vc}$$

- But **no** channel-dependent partitioning, and much easier to implement ($N\log(N)\ vs\ N^2$).

- How can we exploit this??

INCREASE N !

- DMTra and DMTma will help.

$1+.9D^{-1}$ revisited

- Circulant Channel is 8 x 8 (but wastes 1 dimension in cyclic extension, and loses its energy).

$$\widetilde{H} = \begin{bmatrix} .9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .9 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .9 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .9 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .9 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .9 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & .9 \end{bmatrix} \quad 8\text{x}8$$

- Channel FFT (size 8) leads to:

Water-fill (Γ=0 dB) with $\mathcal{E}_x = 8$

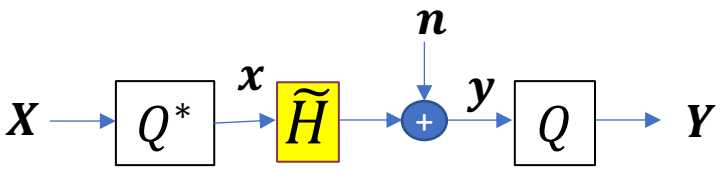
n	$\lambda_n = \lvert P_n \rvert$	$g_n = \frac{\lvert P_n \rvert^2}{.181}$	$\mathcal{E}_n$	SNR_n	b_n
0	1.90	20	1.24	24.8	2.34
1	1.76	17	1.23	20.9	2.23
6	1.76	17	1.23	20.9	2.23
2	1.35	9.8	1.19	11.7	1.85
5	1.35	9.8	1.19	11.7	1.85
3	.733	3	.96	2.9	.969
4	.733	3	.96	2.9	.969
7	.100	.05525	0	0	0

- $\bar{b} = 1.38$

See PS1.4 (Prob 4.7)

Partitioning with DFT

- DFT Partitioning is:



- DFT creates a set of parallel channels – A "Discrete MultiTone" partitioning.
  - Some call it OFDM, but there is a difference in loading (DMT optimizes loaded energy, OFDM fixes equal energy).

$$SNR_n = \frac{\mathcal{E}_n \cdot \left|\tilde{H}_n\right|^2}{\sigma^2}$$

Noise-Equivalent Channel

$$E\left[\boldsymbol{n}\boldsymbol{n}^*\right] = R_{nn}\sigma^2 = R_{nn}^{1/2} R_{nn}^{-1/2}\sigma^2$$

$$\boldsymbol{y} \leftarrow R_{nn}^{-1/2}\boldsymbol{y} = \left(R_{nn}^{-1/2}H\right)\boldsymbol{x} + \tilde{\boldsymbol{n}}$$

- Receiver is DFT (and noise remains white).
- Transmitter is IDFT (no power increase).

Neither is a function of the channel.

Can use efficient "FFT"

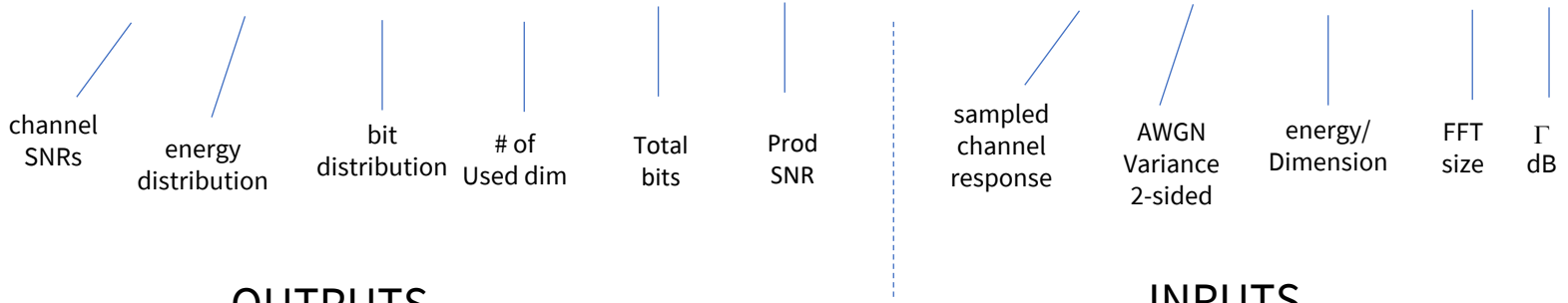
DMT Water-Filling Software

Subsection 4.7.4

See PS1.4 (Prob 4.7) and PS1.5 (Prob 4.9)

Rate Adaptive DMT

```
function [gn,en_bar,bn_bar,Nstar,b_bar,SNRdmt]=DMTra(H,NoisePSD,Ex_bar,N,gap)
```



- >> [gn,en_bar,bn_bar,Nstar,b_bar,SNRdmt]=DMTra([.9 1],.181,1,8,0)

- gn = 19.9448 17.0320 10.0000 2.9680 0.0552 2.9680 10.0000 17.0320
- en_bar = 1.2415 1.2329 1.1916 0.9547 0 0.9547 1.1916 1.2329
- bn_bar = 2.3436 2.2297 1.8456 0.9693 0 0.9693 1.8456 2.2297
- Nstar = 7
- b_bar = 1.3814
- SNRdmt = 7.6247 dB

QAMs

In using the program, know if your channel is truly baseband or complex baseband equivalent.

In real case, each dimension shown is a real dimension.

Increase N

- [gn,en_bar,bn_bar,Nstar,b_bar,SNRdmt]=DMTra([.9 1],.181,1,16,0);
  - >> SNRdmtSNRdmt = 8.1152
- [gn,en_bar,bn_bar,Nstar,b_bar,SNRdmt]=DMTra([.9 1],.181,1,1024,0);
  - >> SNRdmtSNRdmt = 8.7437
- Feel free to experiment, PS1.4 goes better if you use this.

How about non-zero gap?

- SNR can look higher, but bit rate is overall lower

```
>> [gn,En,bn_bar,Nstar,b_bar,SNRdmt]=DMTra([.9 1],.181,1,8,8.8)

gn = 19.9448   17.0320   10.0000    2.9680    0.0552    2.9680   10.0000   17.0320

En = 1.7773    1.7123    1.3991         0         0         0    1.3991    1.7123

bn_bar = 1.2521    1.1382    0.7540         0         0         0    0.7540    1.1382

Nstar = 5

b_bar = 0.5596

SNRdmt = 9.4904 dB  (remember this gets divided by the gap)

>> En.*gn = 35.4481   29.1634   13.9907         0         0         0   13.9907  29.1634
            24.7613   20.9991   11.9163    2.8336         0    2.8336   11.9163  20.9991 (Γ=0)

>> 10*log10(ans)  = 15.4959   14.6484   11.4584      -Inf      -Inf      -Inf   11.4584   14.6484
   (these subchannel SNR's also get divided by gap)
```

Data rate is roughly 1/3 of before.

Note SNRdmt increase for nonzero gap.

Suppose $1+.9D^{-1}$ were complex baseband?

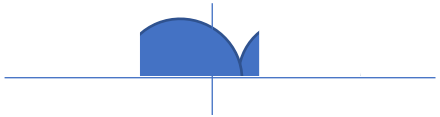
- The channel effectively has twice as many dimensions.
  - The same results would be for 8 complex dimensions.
  - So bbar, ebar are really bits/tone and energy/tone.
  - Same values, but the constellations on each tone are two dimensional.

```
>> [gn,En,bn,Nstar,b,SNRdmt]=DMTra([.9 1],.181,2,8,0)

gn =  19.9448   17.0320   10.0000    2.9680    0.0552    2.9680   10.0000   17.0320

En = 2.3843    2.3758    2.3345    2.0976         0    2.0976    2.3345    2.3758

bn = 2.8008    2.6869    2.3028    1.4266         0    1.4266    2.3028    2.6869

Nstar = 7

b = 1.7370

SNRdmt = 10.0484 dB

>> sum(En) = 16.0000      ;     >> sum(bn) =   15.6332
```

- The bits/tone though is slightly larger.
  - 1.73 > 1.38
- What happened?
- The "DC" tone is now complex and has an additional good dimension.

Suppose $1+.9jD^{-1}$ - *must be* complex baseband

- The channel definitely has twice as many real dimensions

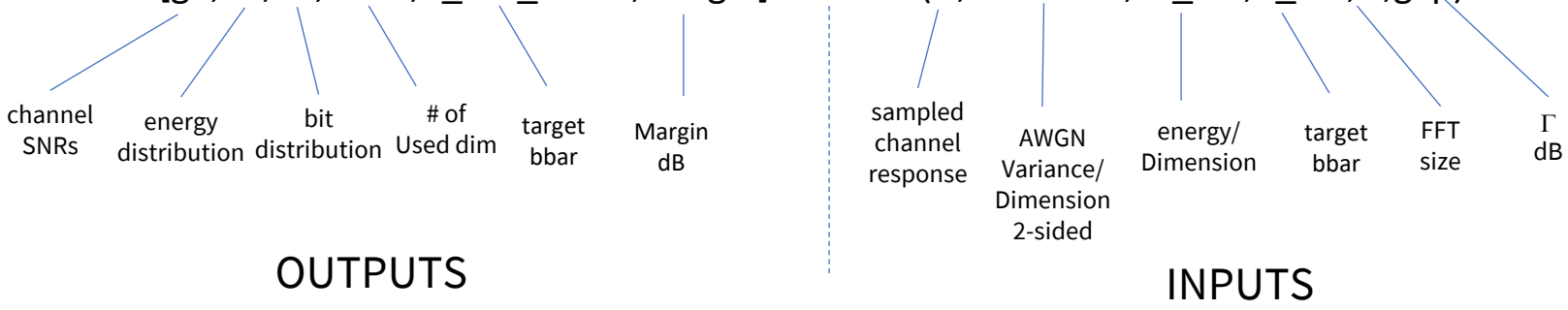


This channel rotates the earlier one in time, so circular shift in frequency

```
>> [gn,En,bn,Nstar,b,SNRdmt]=DMTra([.9*i 1],.181,2,8,0)

gn = 10.0000  17.0320  19.9448  17.0320  10.0000   2.9680   0.0552   2.9680

En = 2.3345   2.3758   2.3843   2.3758   2.3345   2.0976        0   2.0976

bn = 2.3028   2.6869   2.8008   2.6869   2.3028   1.4266        0   1.4266

Nstar = 7

b = 1.7370

SNRdmt = 10.0484

>> sum(En) = 16.0000     ;     >> sum(bn) = 15.6332
```

Margin Adaptive DMT

- function [gn,en,bn,Nstar,b_bar_check,margin]=DMTma(H,NoisePSD,Ex_bar,b_bar,N,gap)



- >> [gn,en,bn,Nstar,b_bar_check,margin]=DMTma([.9 1],.181,1,1,8,0)
- gn = 19.9448 17.0320 10.0000 2.9680 0.0552 2.9680 10.0000 17.0320
- en = 0.6043 0.5958 0.5545 0.3175 0 0.3175 0.5545 0.5958
- bn = 1.8532 1.7393 1.3552 0.4790 0 0.4790 1.3552 1.7393
- Nstar = 7
- b_bar_check = 1
- margin = 3.5410

It works real or complex, but (again) be careful.

Continuing

- >> [gn,en,bn,Nstar,b_bar_check,margin]=DMTma([.9 1],.181,1,1,8,8.8)
- gn = 19.9448 17.0320 10.0000 2.9680 0.0552 2.9680 10.0000 17.0320
- en = 4.5844 4.5193 4.2061 2.4088 0 2.4088 4.2061 4.5193
- bn = 1.8532 1.7393 1.3552 0.4790 0 0.4790 1.3552 1.7393
- Nstar = 7
- b_bar_check = 1.0000
- margin = -5.2590

Negative margin – can't do it!

[gn,en,bn,Nstar,b_bar_check,margin]=DMTma([.9 1],.181,1,1,16,0);

>> margin = 4.1445 (equivalently, code with gain >8.8-4.1=4.5 will work at Pe<1e-6)

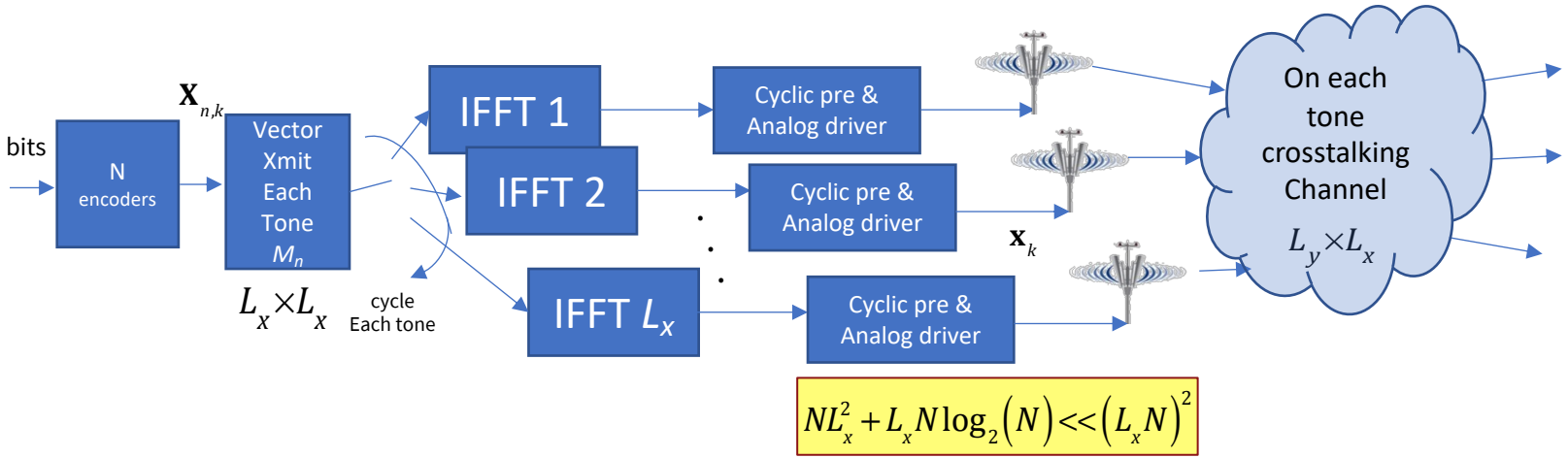
[gn,en,bn,Nstar,b_bar_check,margin]=DMTma([.9 1],.181,1,1,1024,0);

>> margin = 4.7267 (save .6 dB on code and use longer FFT)

Vector DMT/OFDM

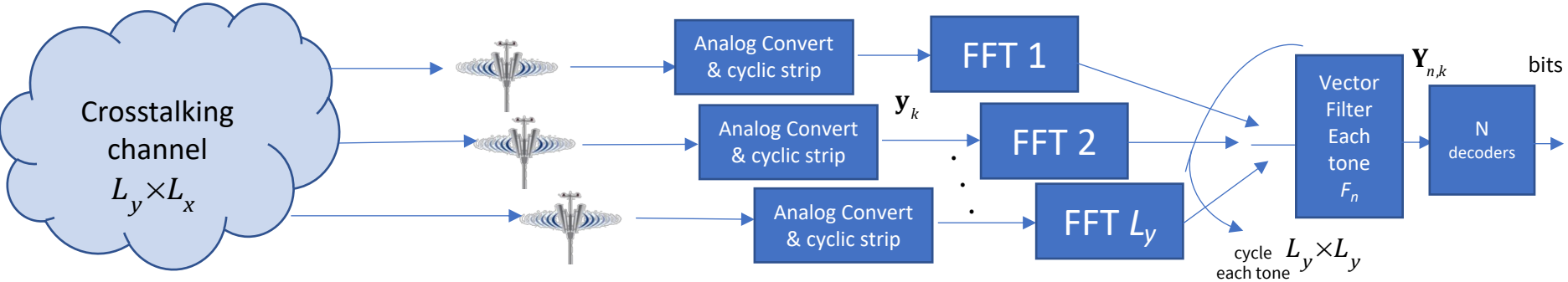
Section 4.7

Vector DMT/OFDM Transmitter



$$NL_x^2 + L_x N \log_2(N) << (L_x N)^2$$

Vector DMT/OFDM Receiver



$$\tilde{H}_n = F_n \cdot \Lambda_n \cdot M_n^*$$

$$NL_y^2 + L_y N \log_2(N) << (L_y N)^2$$

- Just much larger number of dimensions, each a scalar AWGN, $L = min(L_x, L_y)$
- $L \cdot N$ dimensions
- Can water-fill over them all (if total energy constraint, which is common)

N=1024, L=8 --> 64k+80k=144k versus 64M !!!

End Lecture 2