Lecture 13

Optimized GDFE Inputs

May 19, 2023

JOHN M. CIOFFI

Hitachi Professor Emeritus of Engineering

Instructor EE392AA – Spring 2023

Announcements & Agenda

- Announcements
  - May want to revisit L8 (MAC)
  - PS6 due May 24
  - Section 5.4

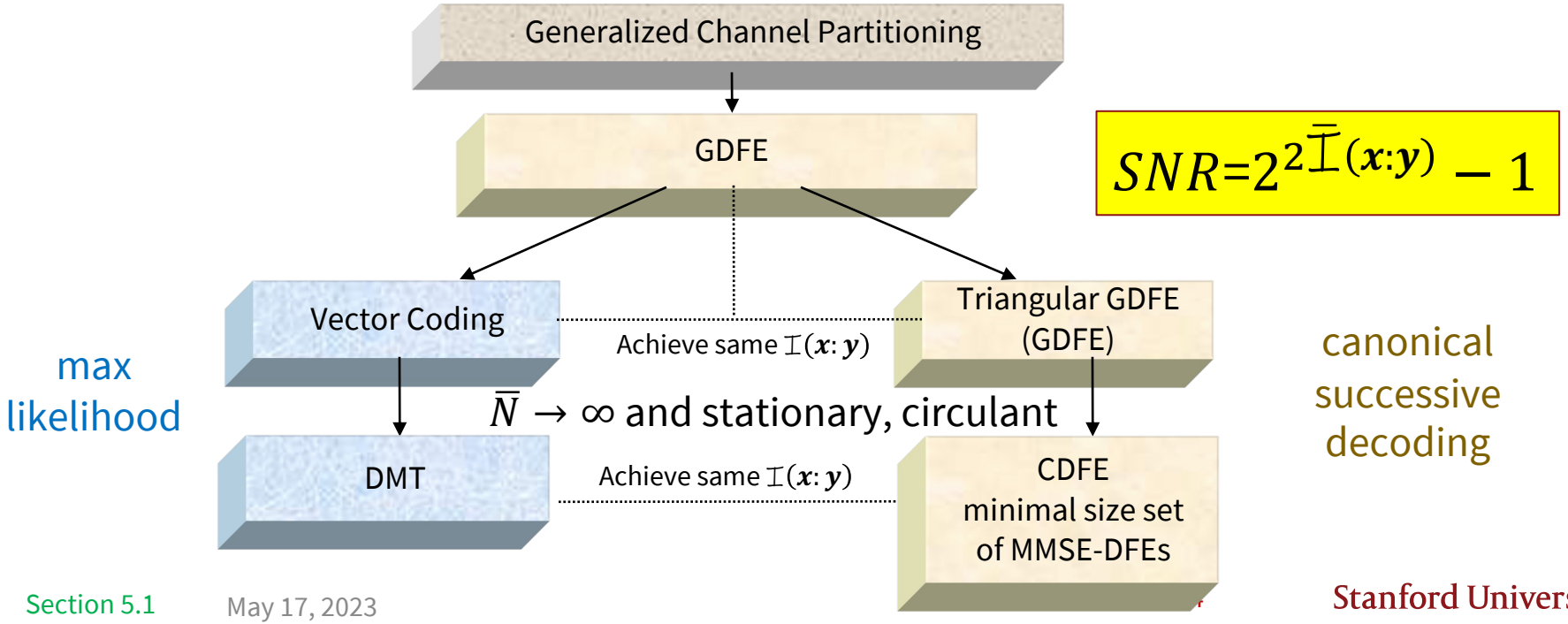
- Agenda
  - Triangular GDFE's (remainder from L12)
  - Tonal GDFE
  - Input Optimization
  - Circulant DFE (CDFE) with optimum (or other) designed inputs
  - ZF/MMSE Convergence Conditions

Special GDFE Forms

Section 5.2

Canonical Performance for any Square Root

- GDFE has **canonical** performance
  - Same good codes for AWGN ($\Gamma \rightarrow 0$ dB) work on GDFE-generated dimensions to drive $P_e \rightarrow 0$ if $\tilde{b} \leq \tilde{c}$.
  - Even though, it is not (usually) an ML detector (notice error propagation not an issue if $P_e \rightarrow 0$)



Circulant DFE

- Uses cyclic prefix, but attempts to model each energized band in time domain
- The consequent square-circulant channel matrix H has full rank, so $\mathcal{N} = \emptyset$
  - As long as H is not matrix of all constant equal values
- With circulant $\overline{N} \times \overline{N}$ input $R_{\boldsymbol{uu}} = \Phi \cdot \Phi^* = R_{\boldsymbol{xx}}$, the CDFE receiver ignores the cyclic prefix
  - The time-domain convolution appears periodic for any symbol
  - The factorization $R_{\boldsymbol{uu}}$=$\Phi \cdot \Phi^*$corresponds to "causal" filter from input $\boldsymbol{v}$
    - Special case $R_{\boldsymbol{xx}} = \mathrm{I}$, then simply direct input to channel, $\boldsymbol{x} = \boldsymbol{u} = \boldsymbol{v}$ with cyclic prefix.
  - The CDFE Imitates EE379's MMSE-DFE as $\overline{N} \to \infty$; and indeed, the CDFE is better for $\overline{N} < \infty$.
- However, good input design (almost always) introduces singularity (think water-filling)
  - This requires some care to create single-carrier OFDM bands
  - And, it's not as simple as just transmit data $\boldsymbol{x} = \boldsymbol{v}$ into the channel – interpolation of some type needed
- Also known as "Single-carrier OFDM" in standards (CDFE name and publication predates SC-OFDM by more than 10 years)

CDFE Example

```
>> H=toeplitz([.9 zeros(1,7)]',[.9 1 zeros(1,6)]);
H(8,1)=1;
>> H=1/sqrt(.181)*H;
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar,~] = computeGDFE(H, eye(8), 2, 9)
>> snrGDFEu =   7.1666 dB
>> GU =
   1.0000   0.4972        0        0        0        0        0   0.4972
        0   1.0000   0.6414        0        0        0        0  -0.2899
        0        0   1.0000   0.6930        0        0        0   0.1780
        0        0        0   1.0000   0.7128        0        0  -0.1113
        0        0        0        0   1.0000   0.7206        0   0.0702
        0        0        0        0        0   1.0000   0.7237  -0.0444
        0        0        0        0        0        0   1.0000   0.7531
        0        0        0        0        0        0        0   1.0000
>> MSWMFU =
   0.2115        0        0        0        0        0        0   0.2351
   0.1798   0.2729        0        0        0        0        0  - 0.1371
  -0.1104   0.1601   0.2948        0        0        0        0   0.0841
   0.0691  -0.1002   0.1525   0.3033        0        0        0  -0.0526
  -0.0435   0.0631  -0.0961   0.1495   0.3066        0        0   0.0332
   0.0275  -0.0399   0.0608  -0.0945   0.1483   0.3079        0  -0.0210
  -0.0174   0.0253  -0.0385   0.0598  -0.0939   0.1478   0.3084   0.0133
  -0.0933   0.0418   0.0010  -0.0439   0.0961  -0.1687   0.2772   0.1647
>> b' =
   1.7297   1.5648   1.5156   1.4978   1.4909   1.4882   1.4871   1.0792
>> bbar =   1.3170
```

- For very long symbol, → 8.4 dB
- G_u → .725 single feedback coefficient = (7.85/6.85)x.633
- W → constant-row feedforward filter
- CDFE's limit is Chapter 3's MMSE-DFE for infinite symbol length
- Same as DMT (which it should be):

```
>> [V,D]=eig(H);
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar,~] = computeGDFE(H, V, cb, 9)
snrGDFEu =   7.1666
>> GU-eye(8) =   1.0e-13 *  (It's an identity for DMT!)
>> MSWMFU(:,1:3) = (note that it is complex – why? Channel is real? V is complex)
-1.5042 + 0.0000i   1.5042 + 0.0000i  -1.5042 + 0.0000i
  0.1018 + 0.1782i   0.0540 - 0.1980i  -0.1782 + 0.1018i
  0.1018 - 0.1782i   0.0540 + 0.1980i  -0.1782 - 0.1018i
 -0.0748 + 0.0831i   0.0831 + 0.0748i   0.0748 - 0.0831i
 -0.0748 - 0.0831i   0.0831 - 0.0748i   0.0748 + 0.0831i
  0.0792 + 0.0000i   0.0792 - 0.0000i   0.0792 + 0.0000i
  0.0798 + 0.0311i   0.0784 - 0.0345i   0.0311 - 0.0798i
  0.0798 - 0.0311i   0.0784 + 0.0345i   0.0311 + 0.0798i
(the receiver FFT is included when we use computeGDFE)

>> b' =
   0.0388   0.9942   0.9942   1.7297   1.7297   2.1943   2.0862   2.0862
Note it's different, but the sum is the same as CDFE
>> bbar =   1.3170
```

Triangular (with guard period) and no energy in $\mathcal{N}$

```
>> H=(1/sqrt(.181))*toeplitz([.9 zeros(1,7)]',[.9 1 zeros(1,7)])
H=(1/sqrt(.181))*toeplitz([.9 zeros(1,7)]',[.9 1 zeros(1,7)]);
>> [Ct, Ot, Ruutt] = fixmod(H, eye(9), eye(9));
>> [A, OA, Ruupp] = fixin(Ruutt, Ct);

A =
   0.7764   0.2012  -0.1811   0.1630  -0.1467   0.1320  -0.1188   0.1069
   0.2012   0.8189   0.1630  -0.1467   0.1320  -0.1188   0.1069  -0.0962
  -0.1811   0.1630   0.8533   0.1320  -0.1188   0.1069  -0.0962   0.0866
   0.1630  -0.1467   0.1320   0.8812   0.1069  -0.0962   0.0866  -0.0779
  -0.1467   0.1320  -0.1188   0.1069   0.9038   0.0866  -0.0779   0.0702
   0.1320  -0.1188   0.1069  -0.0962   0.0866   0.9221   0.0702  -0.0631
  -0.1188   0.1069  -0.0962   0.0866  -0.0779   0.0702   0.9369   0.0568
   0.1069  -0.0962   0.0866  -0.0779   0.0702  -0.0631   0.0568   0.9489
  -0.0962   0.0866  -0.0779   0.0702  -0.0631   0.0568  -0.0511   0.0460

>> Gxbar=lohc(Ruupp);
>> Gx=Gxbar*inv(diag(diag(Gxbar))) ;
>> Xmit=A*Gxbar;

   0.8812  -0.0000   0.0000   0.0000  -0.0000   0.0000  -0.0000   0.0000
   0.2283   0.8757   0.0000   0.0000  -0.0000  -0.0000  -0.0000   0.0000
  -0.2055   0.2397   0.8681  -0.0000   0.0000   0.0000   0.0000   0.0000
   0.1850  -0.2157   0.2554   0.8574  -0.0000   0.0000   0.0000   0.0000
  -0.1665   0.1942  -0.2299   0.2779   0.8416   0.0000   0.0000  -0.0000
   0.1498  -0.1747   0.2069  -0.2501   0.3120   0.8163   0.0000   0.0000
  -0.1348   0.1573  -0.1862   0.2251  -0.2808   0.3678   0.7710  -0.0000
   0.1213  -0.1415   0.1676  -0.2026   0.2527  -0.3310   0.4733   0.6690
  -0.1092   0.1274  -0.1508   0.1823  -0.2274   0.2979  -0.4260   0.7433
```

```
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar] = computeGDFE(H, Xmit,2,9);

>> snrGDFEu =    7.4896 dB

>> GU =
   1.0000   0.8573   0.0000   0.0000  -0.0000  -0.0000  -0.0000  -0.0000
        0   1.0000   0.7628   0.0000  -0.0000   0.0000   0.0000   0.0000
        0        0   1.0000   0.7174  -0.0000   0.0000   0.0000   0.0000
        0        0        0   1.0000   0.6899   0.0000   0.0000  -0.0000
        0        0        0        0   1.0000   0.6624   0.0000   0.0000
        0        0        0        0        0   1.0000   0.6189   0.0000
        0        0        0        0        0        0   1.0000   0.5233
        0        0        0        0        0        0        0   1.0000

>> MSWMFU =
   0.4165   0.0000   0.0000  -0.0000  -0.0000  -0.0000  -0.0000  -0.0000
   0.0471   0.3738   0.0000  -0.0000   0.0000   0.0000   0.0000   0.0000
  -0.0294   0.0650   0.3560  -0.0000  -0.0000  -0.0000   0.0000   0.0000
   0.0178  -0.0394   0.0693   0.3487   0.0000   0.0000  -0.0000  -0.0000
  -0.0104   0.0231  -0.0407   0.0669   0.3452  -0.0000   0.0000   0.0000
   0.0058  -0.0129   0.0227  -0.0374   0.0599   0.3415   0.0000   0.0000
  -0.0029   0.0065  -0.0114   0.0188  -0.0302   0.0479   0.3328   0.0000
   0.0011  -0.0024   0.0042  -0.0069   0.0111  -0.0176   0.0279   0.3023

> b' =
   1.3789   1.4498   1.4859   1.5067   1.5249   1.5512   1.6042   1.7592
>> bbar =   1.3623
```

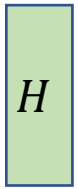
Outperforms CDFE
Same as VC w.r.t. DMT

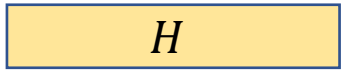
- Better!
- Convergence also somewhat evident, and easier with WF

Try running computeGDFE to implement Vector Coding – what Xmit? (hint this is easy with this white input)

Massive MIMO = diagonal dominance

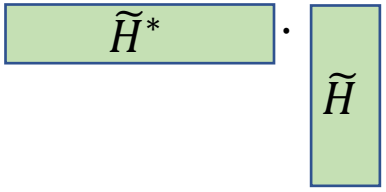
- The channel matrix H is very
  - tall (many more receiver dimensions than ϱ_H) – uplink
  - Fat (many more transmitter dimensions than ϱ_H) - downlink





- IF **tall** channel matrix $\tilde{H}$ has entries $\tilde{h}_{il}$ that are largely uncorrelated, except for same index

$$\mathbb{E}\left[\tilde{h}_{il} \cdot \tilde{h}_{kl}\right] = \left|\tilde{h}\right|^2 \cdot L_y \cdot \delta_{ik}$$



- THEN $R_f = \tilde{H}^* \cdot \tilde{H}$ off-diag's contain uncorrelated values
  - Law of large numbers – off diagonals ($\cdot\ 1/L_y$) go to zero, while diagonal grows relatively

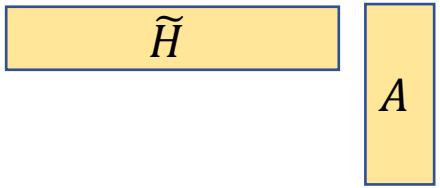
- Then R_b is also almost diagonal, like R_f

- $G = I \rightarrow$ canonical/ML with no feedback!
- Linear is sufficient and returns to ML

Massive MIMO Downlink continued

- Fat case (downlink) – small number of receivers
  - Best transmit A matrix (special square root in BC case, but even with single-user) attempts to match its long columns to the long rows of **fat** $\tilde{H}$, think water-filling (formal GDFE xmit optimization comes soon)
- IF **fat** channel matrix $\tilde{H}$ has entries $\tilde{h}_{il}$ that are largely uncorrelated, except for same index

$$\mathbb{E}[\tilde{h}_{il} \cdot a_{kl}] = \mathcal{E} \cdot L_y \cdot \delta_{ik}$$



Wireless: use many antennas at base

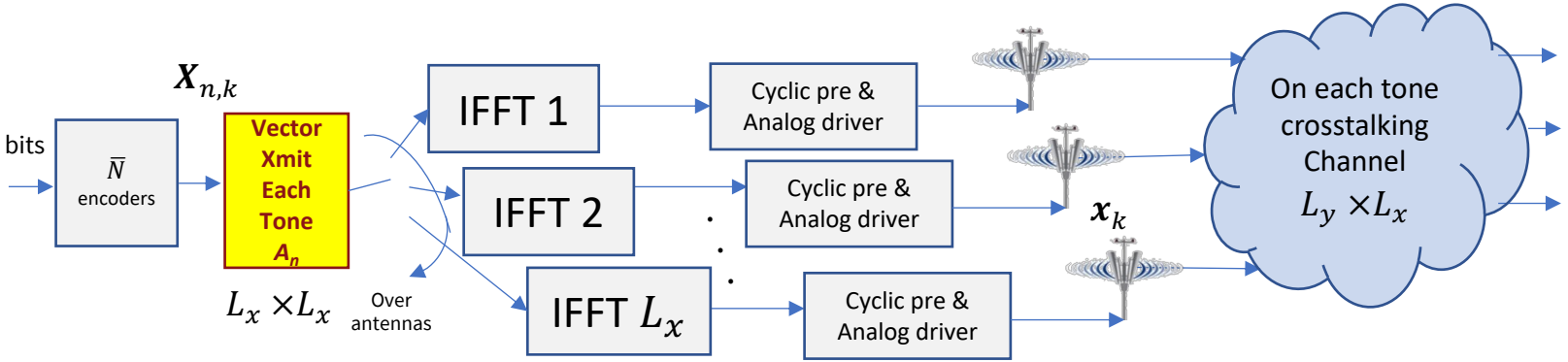
Wireline: vectored DSLs (the Gbps DSLs) often happens even when large square (not fat nor tall)

- THEN $R_b = A^* \cdot \tilde{H}^* \cdot \tilde{H} \cdot A + I$ is diagonal dominant
  - Law of large numbers again
- Here $R_{vv} = I$ by design, then and again there is no feedback
- $G = I \rightarrow$ canonical/ML with no feedback!

Again: no precoder (linear is sufficient)

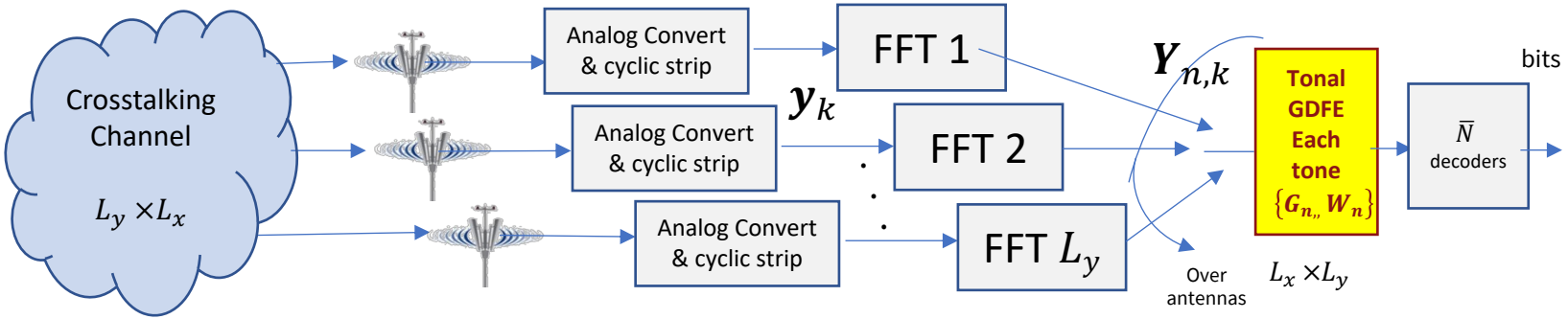
Tonal GDFE

Tonal GDFE Transmitter (see L2:31)



- Synchronize them all (like Vector DMT in Chapter 4) – the transmit filter changes (not necessarily M from SVD)

Tonal GDFE Receiver (see L2:32)



- Common symbol boundary for all antennas also at receiver.

Tonal GDFE Example

```
>> h= cat(3, h0 , h1);
>> N=8;
>> H=(10)*fft(h, N, 3)
H(:,:,1) =
  20.0000 + 0.0000i  -9.0000 + 0.0000i
   6.0000 + 0.0000i   1.0000 + 0.0000i
H(:,:,2) =
  17.0711 - 7.0711i  -7.8284 + 2.8284i
   6.8787 + 2.1213i   3.6360 + 6.3640i
H(:,:,3) =
  10.0000 -10.0000i  -5.0000 + 4.0000i
   9.0000 + 3.0000i  10.0000 + 9.0000i
H(:,:,4) =
   2.9289 - 7.0711i  -2.1716 + 2.8284i
  11.1213 + 2.1213i  16.3640 + 6.3640i
H(:,:,5) =
   0.0000 + 0.0000i  -1.0000 + 0.0000i
  12.0000 + 0.0000i  19.0000 + 0.0000i
H(:,:,6) =
   2.9289 + 7.0711i  -2.1716 - 2.8284i
  11.1213 - 2.1213i  16.3640 - 6.3640i
H(:,:,7) =
  10.0000 +10.0000i  -5.0000 - 4.0000i
   9.0000 - 3.0000i  10.0000 - 9.0000i
H(:,:,8) =
  17.0711 + 7.0711i  -7.8284 - 2.8284i
   6.8787 - 2.1213i   3.6360 - 6.3640i
```

$$H(D) = \begin{bmatrix} 1+D & -.5-.4\cdot D \\ .9-.3\cdot D & 1-.9\cdot D \end{bmatrix}$$

$$\boldsymbol{h}_0 = \begin{bmatrix} 1 & -.5 \\ .9 & 1 \end{bmatrix} \quad \boldsymbol{h}_1 = \begin{bmatrix} 1 & -.4 \\ -.3 & -.9 \end{bmatrix}$$

$$R_{nn} = .01 \cdot I$$

```
>> A=zeros(2,2,8);
for n=1:N
A(:,:,n) = sqrt(8/9)*eye(2);
end

>> cb=1;
>>Lx = 2*(9/8):
>> GU=zeros(2,2,8);
>> WU=zeros(2,2,8);
>> S0=zeros(2,2,8);
>> MSWMFU=zeros(2,2,8);

>> b=zeros(2,1,8);
>> bbar=zeros(1,8);

>> for n=1:N
[snrGDFEu(1,n), GU(:,:,n), WU(:,:,n), S0(:,:,n), MSWMFU(:,:,n),
b(:,:,n), bbar(n)] = ...
computeGDFE(H(:,:,n), A(:,:,n), cb, Lx);
end
```

Loop the design for Each tone

```
>> snrGDFEu = (in dB)
 16.2546  19.6868  20.8260  19.4629  11.9618  19.4629  20.8260  19.6868
>> GU
GU(:,:,1) =
   1.0000 + 0.0000i  -0.3991 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,2) =
   1.0000 + 0.0000i  -0.2928 + 0.0737i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,3) =
   1.0000 + 0.0000i   0.0931 + 0.1414i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,4) =
   1.0000 + 0.0000i   0.9056 + 0.1552i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,5) =
   1.0000 + 0.0000i   1.5833 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,6) =
   1.0000 + 0.0000i   0.9056 - 0.1552i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,7) =
   1.0000 + 0.0000i   0.0931 - 0.1414i
   0.0000 + 0.0000i   1.0000 + 0.0000i
GU(:,:,8) =
   1.0000 + 0.0000i  -0.2928 - 0.0737i
   0.0000 + 0.0000i   1.0000 + 0.0000i
```

8 FB Sections

Continued

```
>> MSWMFU

MSWMFU(:,:,1) =
  0.0487 + 0.0000i   0.0146 + 0.0000i
 -0.0865 + 0.0000i   0.2821 + 0.0000i
MSWMFU(:,:,2) =
  0.0460 + 0.0191i   0.0186 - 0.0057i
 -0.0409 + 0.0060i   0.0705 - 0.0787i
MSWMFU(:,:,3) =
  0.0366 + 0.0366i   0.0329 - 0.0110i
 -0.0364 - 0.0175i   0.0476 - 0.0370i
MSWMFU(:,:,4) =
  0.0166 + 0.0402i   0.0632 - 0.0120i
 -0.0381 - 0.0564i   0.0431 - 0.0177i
MSWMFU(:,:,5) =
  0.0000 + 0.0000i   0.0884 + 0.0000i
 -0.2792 + 0.0000i   0.0411 + 0.0000i
MSWMFU(:,:,6) =
  0.0166 - 0.0402i   0.0632 + 0.0120i
 -0.0381 + 0.0564i   0.0431 + 0.0177i
MSWMFU(:,:,7) =
  0.0366 - 0.0366i   0.0329 + 0.0110i
 -0.0364 + 0.0175i   0.0476 + 0.0370i
MSWMFU(:,:,8) =
  0.0460 - 0.0191i   0.0186 + 0.0057i
 -0.0409 - 0.0060i   0.0705 + 0.0787i
```

8 FF Sections

- Bit distribution
  - [space, freq]
- Lx=2.25; (2*9/8) handled cyclic-prefix dimension loss already
  - 2x because cmplx (cb=1)
  - bbar is bits/ real-dim
  - Only divide by 2*8 then for equivalent overall SNR

%Overall SNR from bbar

```
>> reshape(b,[2,8])  =
   8.6020   8.4535   8.0156   7.3838   7.0112   7.3838   8.0156   8.4535
   3.6233   6.2959   7.5772   7.1999   2.1297   7.1999   7.5772   6.2959

>> bbar =  % This is where computeGDFE used the Lx=2.25
   5.4334   6.5553   6.9301   6.4817   4.0627   6.4817   6.9301   6.5553
>> 8.6+3.6/2.25 =  5.4

>> sum(b,'all')  =  111.2179

>> sum(bbar)/16 = 3.0894
>> 10*log10(2^(2*ans)-1)  = 18.5396 dB
```

Can now design for multiple antennas, ISI, crosstalk – and its canonical!

Time to/from Freq

- **MIMO Channel:** $\boldsymbol{H}(D) = \boldsymbol{h}_0 + \boldsymbol{h}_1 \cdot D + \boldsymbol{h}_2 \cdot D^2 + \cdots + \boldsymbol{h}_\nu \cdot D^\nu$

- Matlab's conversion to Frequency-Domain with FFT is NOT unitary (it increases energy by FFT size)

```
h= cat(3, h0 , h1, …, hnu);
H=fft(h, N, 3)
```

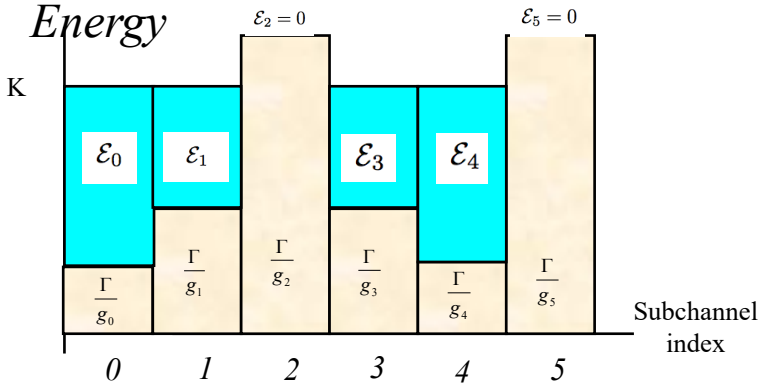
- **Why not $1/\sqrt{\bar{N}}$?** With fixed sampling period 'T'
  - $T = \bar{N} \cdot T'$; which means the energy per symbol grows by $\bar{N}$ for both input and noise
  - $\mathcal{E}_{\boldsymbol{x}} = \bar{N} \cdot \tilde{\mathcal{E}}_x = N \cdot \bar{\mathcal{E}}_x$
  - So make sure, depending on program/analysis, that the energy per symbol is $\bar{N}$ times larger

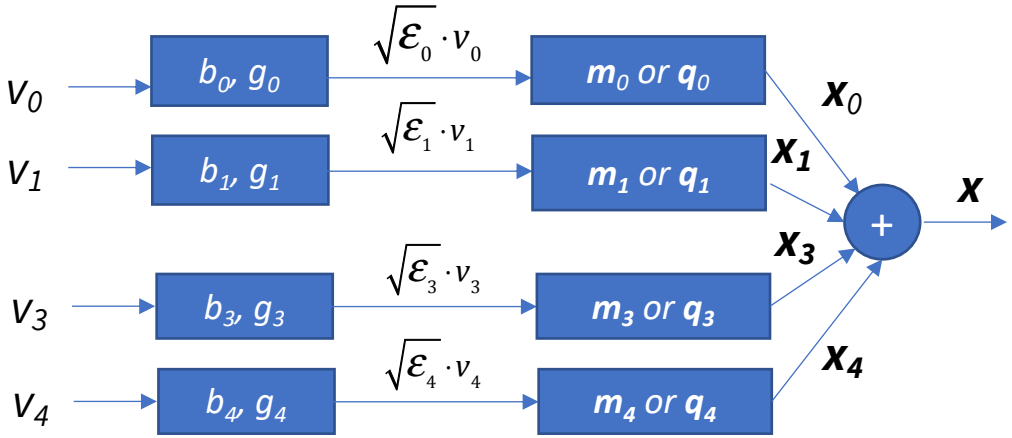
- But also: noise-whitening to $R_{\boldsymbol{nn}} = I \cdot \delta_k$ (time domain) so 1 unit/sample effectively increases the noise by $\bar{N}$ factor, so by amplitude $\sqrt{\bar{N}}$. This means the channel $\boldsymbol{h}_k$ needs to match this increase of $\sqrt{\bar{N}}$.
  - So dividing $\boldsymbol{h}_k$ by the single-sample (two-sided) square-root PSD level σ effectively increases the noise
  - Easiest accommodation is to use matlab's FFT directly with no scaling

Input Optimization

Section 5.3

Water-filling (or LC or …) occurs first



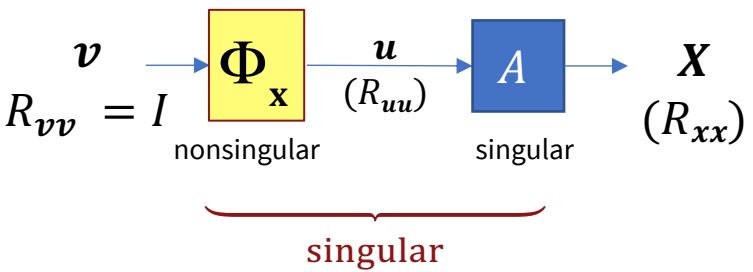


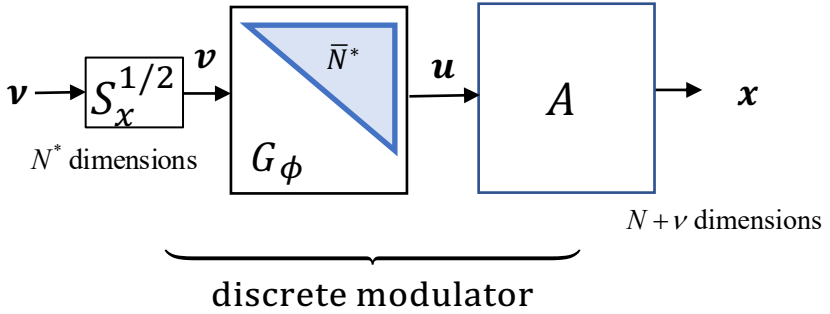
- The input optimization can be done with VC or DMT
- Optimization finds an R_{xx} that subsequently can be factored (don't really need bit dist'n)
  - Construct as modulator
  - Need M (vector coding) and/or Q (IDFT, DMT) matrices

$$R_{xx} = M \cdot diag\{\mathcal{E}_x\} \cdot M^*$$

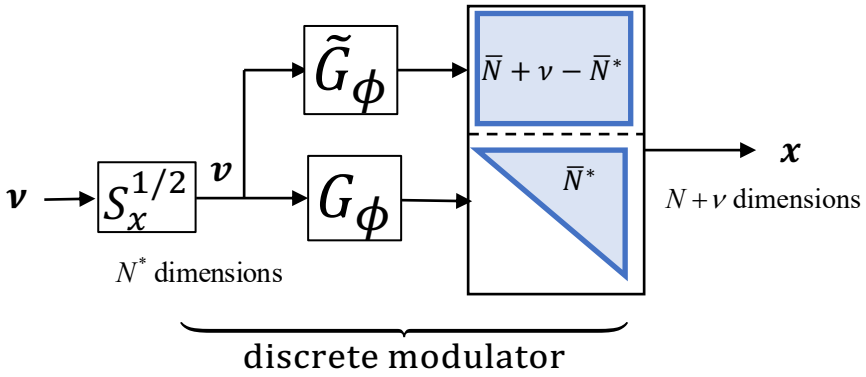
- Need not be water-filling, just some optimization or design

Optimum Transmit Structures





- Uses an optimum or good R_{xx} from VC or DMT
- Factorizations:
  - Cholesky on R_{uu} if A already known (e.g., via singularity elimination)
    - R_{uu} is nonsingular
    - Other square roots also allowed (including eigen decomposition)
  - Generalized Cholesky (if A not yet known)



- *1/T** and f_c* implemented digitally (corresponds to MMSE-DFE) – next section
- The singular part returns to nonsingular through receiver's matched-filter-matrix to form R_f

Example – 8 dimensions waterfill DMT; 7 dimensions GDFE

- From Section 4.6:
  - [gn,en_bar,bn_bar,Nstar,bbar,SNRdmt]=DMTra([.9 1],.181,1,8,0)
  - en_bar = 1.2415 1.2329 1.1916 0.9547 0 0.9547 1.1916 1.2329

Optimize INPUT

```
>> rXX=diag([en_bar(8:-1:1)]) =
  1.2329      0      0      0      0      0      0      0
       0 1.1916      0      0      0      0      0      0
       0      0 0.9547      0      0      0      0      0
       0      0      0      0      0      0      0      0
       0      0      0      0 0.9547      0      0      0
       0      0      0      0      0 1.1916      0      0
       0      0      0      0      0      0 1.2329      0
       0      0      0      0      0      0      0 1.2415
>> rXXbar=diag([en_bar(8:-1:6) en_bar(4:-1:1)]) =
  1.2329      0      0      0      0      0      0
       0 1.1916      0      0      0      0      0
       0      0 0.9547      0      0      0      0
       0      0      0 0.9547      0      0      0
       0      0      0      0 1.1916      0      0
       0      0      0      0      0 1.2329      0
       0      0      0      0      0      0 1.2415
>> J=hankel([zeros(1,7),1]);
>> Q=(1/sqrt(8))*J*fft(J);
>> J7=hankel([zeros(1,6),1]);
>> Qtilde=(1/sqrt(7))*J7*fft(J7);
>> ruu=real(Qtilde'*rXXbar*Qtilde); =
>>norm(imag(Qtilde'*rXXbar*Qtilde))=1e-16
(proves taking real part only for appearance)
>> Gubar=lohc(ruu);
```

Cholesky needs Nonsingular; Here 7x7

Interpolate INPUT with two IFFT's

```
>> Jg = [
     1     0     0     0     0     0     0
     0     1     0     0     0     0     0
     0     0     1     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     1     0     0     0
     0     0     0     0     1     0     0
     0     0     0     0     0     1     0
     0     0     0     0     0     0     1 ];
>>A=real(Q'*Jg*Qtilde*Gubar);
>> C=[.9
zeros(6,1)
1];
>> R=[.9 1 0 0 0 0 0 0];
>> H=toeplitz(C,R);
>> Ht=(1/sqrt(.181))*H;
```

8x7

FFT, then IFFT

Compute GDFE

```
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar] = computeGDFE(Ht, A, 2, 9)

snrGDFEu =    7.6247 dB
>> GU =
   1.0000    0.4654   -0.0309   -0.0024    0.0245   -0.0574    0.4464
        0    1.0000    0.5340   -0.0310   -0.0052    0.0327   -0.2178
        0         0    1.0000    0.5510   -0.0307   -0.0091    0.1120
        0         0         0    1.0000    0.5554   -0.0289   -0.0499
        0         0         0         0    1.0000    0.5549   -0.0102
        0         0         0         0         0    1.0000    0.5555
        0         0         0         0         0         0    1.0000

>> MSWMFU =
   0.2144    0.0140   -0.0036    0.0019   -0.0022    0.0042   -0.0120    0.1767
   0.0788    0.2646    0.0434   -0.0124    0.0080   -0.0090    0.0157   -0.0972
  -0.0572    0.0191    0.2737    0.0812   -0.0222    0.0152   -0.0179    0.0609
   0.0413   -0.0281   -0.0237    0.2623    0.1233   -0.0296    0.0215   -0.0421
  -0.0318    0.0250   -0.0033   -0.0541    0.2364    0.1663   -0.0321    0.0320
   0.0279   -0.0225    0.0093    0.0179   -0.0731    0.1999    0.2058   -0.0254
  -0.1112    0.0652   -0.0290   -0.0061    0.0501   -0.1141    0.2104    0.1753


>> b' =   1.8981   1.8022   1.7816   1.7762   1.7752   1.7762   1.6233

>> bbar =   1.3814
```

SNR Improves (xmit opt)

Circulant DFE

3GPP calls this "Single-Carrier OFDM"

Repeated for each of $L_y \cdot L_x$ spatial paths

Start with OFDM-like signal

- IFFT from tone inputs $\boldsymbol{X}$ $\quad \boldsymbol{x} = Q^* \cdot \boldsymbol{X}$

$$\boldsymbol{X} = \begin{bmatrix} \widetilde{\boldsymbol{X}}_M \\ \vdots \\ \widetilde{\boldsymbol{X}}_2 \\ 0 \\ \widetilde{\boldsymbol{X}}_1 \\ 0 \end{bmatrix} = J_g \cdot \begin{bmatrix} \widetilde{\boldsymbol{X}}_M \\ \vdots \\ \widetilde{\boldsymbol{X}}_1 \end{bmatrix} \qquad R_{\boldsymbol{xx}} = Q^* \cdot J_g \cdot R_{\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}} \cdot J_g \cdot Q$$

- Dimensions $\quad \bar{N}^* = \sum_{i=1}^{M} \bar{N}_i$

- Nonsingular, but circulant? $\quad |R_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}}| > 0$

complex baseband

$$J_g = \begin{bmatrix} 0_{\bar{N}_{z,M+1} \times \bar{N}_M} & 0_{\bar{N}_{z,M+1} \times \bar{N}_{M-1}} & \cdots & 0_{\bar{N}_{z,M+1} \times \bar{N}_1} \\ I_{\bar{N}_M \times \bar{N}_M} & 0_{\bar{N}_M \times \bar{N}_{M-1}} & \cdots & 0_{\bar{N}_M \times \bar{N}_1} \\ 0_{\bar{N}_{z,M} \times \bar{N}_M} & 0_{\bar{N}_{z,M} \times \bar{N}_{M-1}} & \cdots & 0_{\bar{N}_{z,M} \times \bar{N}_1} \\ 0_{\bar{N}_{z,M-1} \times \bar{N}_M} & I_{\bar{N}_{M-1} \times \bar{N}_{M-1}} & \cdots & 0_{\bar{N}_{z,M-1} \times \bar{N}_1} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{\bar{N}_{z,1} \times \bar{N}_M} & 0_{\bar{N}_{z,1} \times \bar{N}_{M-1}} & \cdots & I_{\bar{N}_1 \times \bar{N}_1} \\ 0_{\bar{N}_{z,1} \times \bar{N}_M} & 0_{\bar{N}_{z,1} \times \bar{N}_{M-1}} & \cdots & 0_{\bar{N}_{z,1} \times \bar{N}_1} \end{bmatrix}$$

real baseband

$$J_g = \begin{bmatrix} J_g^- & 0_{\frac{N}{2} \times \left(N_1^- + N_M^- + \sum_{m=2}^{M-1} N_m\right)} \\ 0_{\frac{N}{2} \times \left(N_1^+ + N_M^+ + \sum_{m=2}^{M-1} N_m\right)} & J_g^+ \end{bmatrix},$$

where J_g^- and J_g^+ are respectively defined by

$$J_g^- = \begin{bmatrix} 0_{N_{z,1}^- \times N_1^-} & 0_{N_{z,1}^- \times N_2} & \cdots & 0_{N_{z,1}^- \times N_M^-} \\ I_{N_1^- \times N_1^-} & 0_{N_1^- \times N_2} & \cdots & 0_{N_1^- \times N_M^-} \\ 0_{N_{z,2} \times N_1^-} & 0_{N_{z,2}^- \times N_2} & \cdots & 0_{N_{z,2} \times N_M^-} \\ 0_{N_2 \times N_1^-} & I_{N_2 \times N_2} & \cdots & 0_{N_2 \times N_M^-} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{N_M^- \times N_1^-} & 0_{N_M^- \times N_2} & \cdots & I_{N_M^- \times N_M^-} \\ 0_{N_{z,M}^- \times N_1^-} & 0_{N_{z,1}^- \times N_2} & \cdots & 0_{N_{z,M}^- \times N_M^-} \end{bmatrix}$$

and

$$J_g^+ = \begin{bmatrix} 0_{N_{z,M}^+ \times N_M^+} & 0_{N_{z,M}^+ \times N_{M-1}} & \cdots & 0_{N_{z,M}^+ \times N_1^+} \\ I_{N_M^+ \times N_M^+} & 0_{N_M^+ \times N_{M-1}} & \cdots & 0_{N_M^+ \times N_1^+} \\ 0_{N_{z,M-1} \times N_M^+} & 0_{N_{z,M-1} \times N_{M-1}} & \cdots & 0_{N_{z,M-1} \times N_1^+} \\ 0_{N_{M-1} \times N_M^+} & I_{N_{M-1} \times N_{M-1}} & \cdots & 0_{N_{M-1} \times N_1^+} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{N_1^+ \times N_M^+} & 0_{N_1^+ \times N_{M-1}} & \cdots & I_{N_1^+ \times N_1^+} \\ 0_{N_{z,1}^+ \times N_M^+} & 0_{N_{z,1}^+ \times N_{M-1}} & \cdots & 0_{N_{z,1}^+ \times N_1^+} \end{bmatrix}.$$

New autocorrelation is nonsingular

- Form the individual bands from time-domain signals

$$\widetilde{\boldsymbol{X}} = \begin{bmatrix} \widetilde{\boldsymbol{X}}_M \\ \vdots \\ \widetilde{\boldsymbol{X}}_1 \end{bmatrix} = \underbrace{\begin{bmatrix} Q_{\bar{N}_M} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Q_{\bar{N}_1} \end{bmatrix}}_{\tilde{Q}} \cdot \begin{bmatrix} \boldsymbol{u}_M \\ \vdots \\ \boldsymbol{u}_1 \end{bmatrix}$$

- Real baseband case has to impose conjugate symmetry - see notes

- Each band has an $R_{\boldsymbol{uu}}(i)\ ;\ i = 1, \ldots, M$

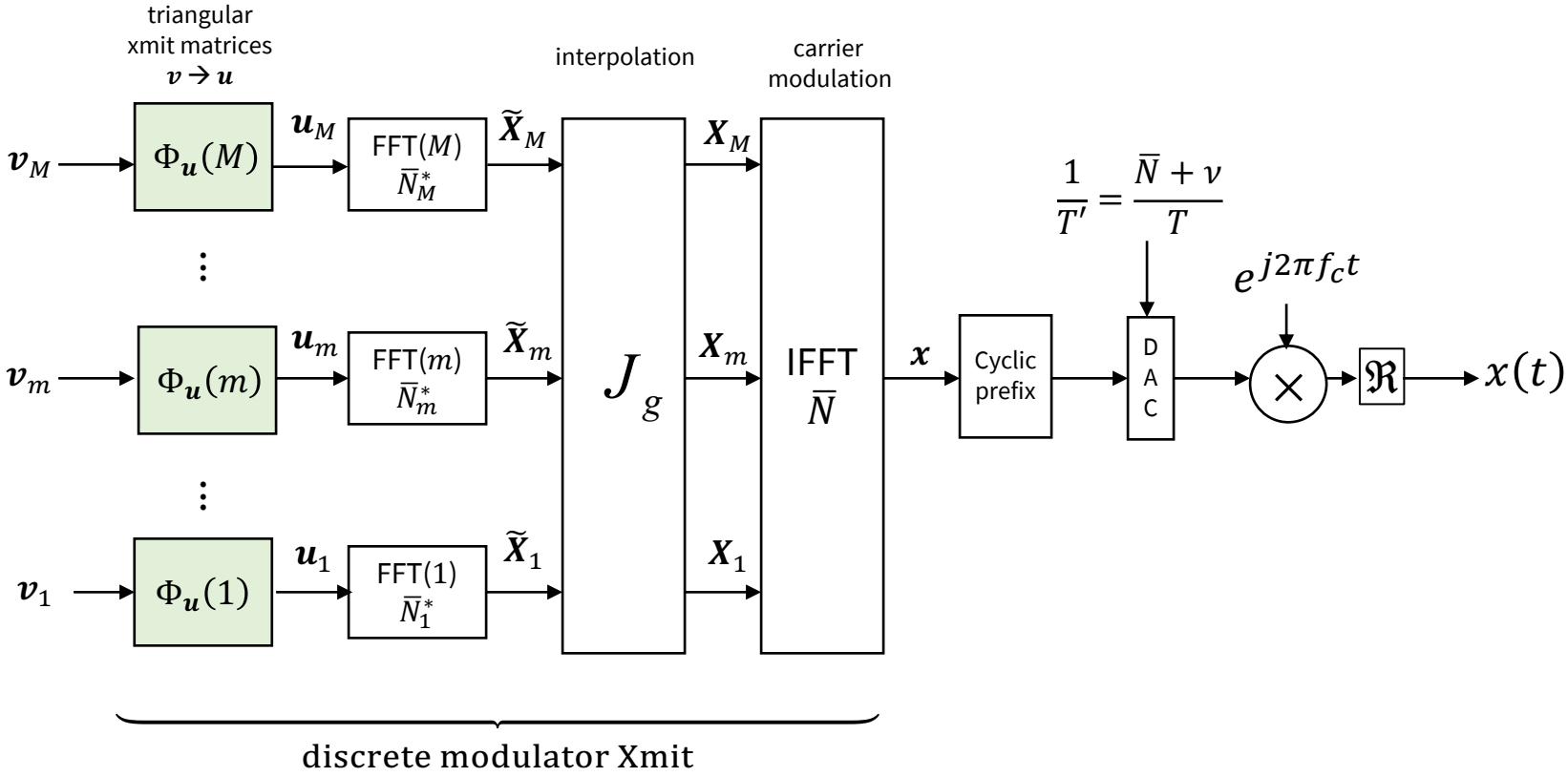
$$R_{\boldsymbol{uu}} = \begin{bmatrix} R_{\boldsymbol{uu}}(M) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R_{\boldsymbol{uu}}(1) \end{bmatrix} = \begin{bmatrix} \Phi(M) \cdot \Phi^*(M) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \Phi(1) \cdot \Phi^*(1) \end{bmatrix}$$

$$R_{\boldsymbol{uu}}(i) = \Phi(i) \cdot \Phi^*(i) = G_x(i) \cdot S_x(i) \cdot G_x^*(i)$$

- So far, the input is

$$\boldsymbol{x} = Q^* \cdot J_g \cdot \tilde{Q} \cdot \boldsymbol{u} \qquad = Q^* \cdot J_g \cdot \tilde{Q} \cdot \Phi \cdot \boldsymbol{v}$$

The CDFE Transmitter(s)

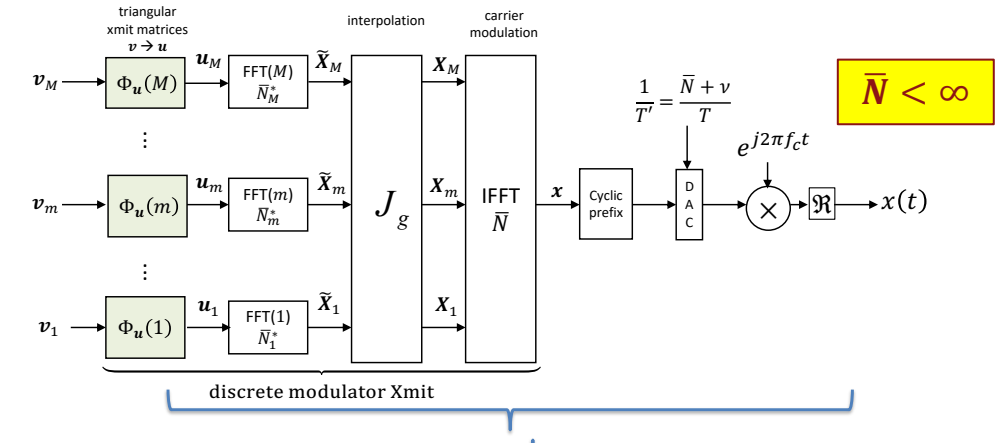


Cyclic Convergence (Toeplitz Dist'n) – Each Band

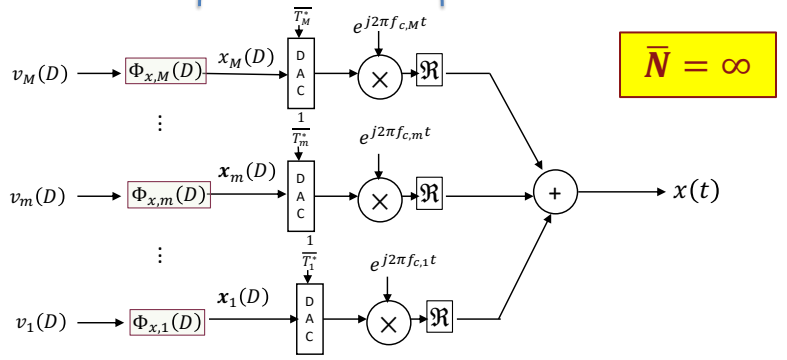
- Applies to temporal dimensionality (time-frequency) when (block) stationary, so $R_{\boldsymbol{xx}}$ is (block) Toeplitz
- Start with linear prediction (Cholesky Factors define this)
  - $\boldsymbol{v}_N = \boldsymbol{x}_N - \boldsymbol{\phi}_1^* \cdot \boldsymbol{x}_{N-1} - \cdots - \boldsymbol{\phi}_{N-1}^* \cdot \boldsymbol{x}_0$
- Found through orthogonality principle
  - $\mathbb{E}\left[\boldsymbol{v}_N \cdot \boldsymbol{x}_{N-i}^*\right] = 0 \;\; \forall\, i = 1, \ldots, \bar{N} - 1$
- Which has limiting value as the stationary predictor in D-Transform notation
  - $\lim_{N\to\infty} [I \quad \boldsymbol{\phi}_1^* \quad \cdots \quad \boldsymbol{\phi}_{N-1}^*] = \boldsymbol{\Phi}^*(D)$
- With energy per sample (or one block of block Toeplitz autocorrelation)
  - $\lim_{N\to\infty} R_{\boldsymbol{vv}}(i) = \mathbb{E}[\boldsymbol{v}_i \cdot \boldsymbol{v}_i^*] = S_v = \mathcal{E}_{\boldsymbol{x}} = trace\left\{R_{\boldsymbol{x}_i \boldsymbol{x}_i}\right\} \forall\, i \geq 0$

Transmitters as $\bar{N} \to \infty$

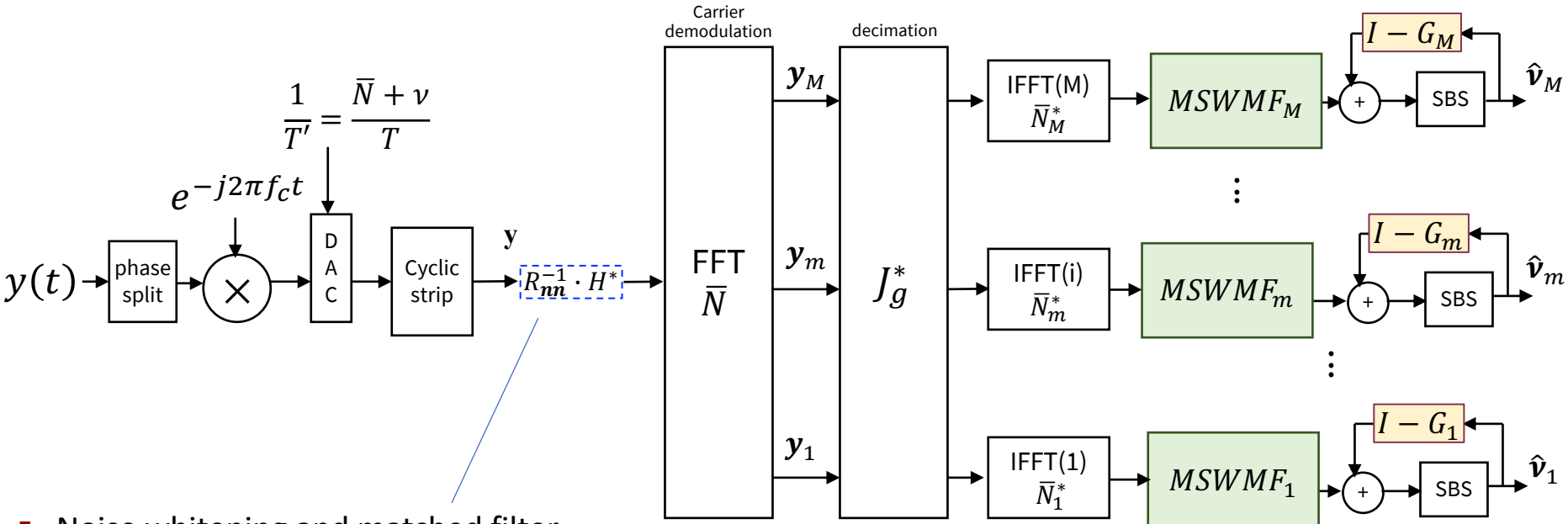
- Upper diagram is digital signal processing
  - 3GPP's Uplink uses this
  - localized mapping (one $G_{\boldsymbol{u}}(m)$ is active)
    - Think of this as $[I \quad \boldsymbol{\phi}_1^* \quad \cdots \quad \boldsymbol{\phi}_{N-1}^*]$ from previous page
  - distributed mapping (>1 $G_{\boldsymbol{u}}(m)$ are active)



- Lower diagram is continuous time (& infinite block)
  - Cyclic prefix unnecessary with infinite block length
  - 3GPP SC-OFDM uses cp with finite length
  - Must synchronize between devices (think MAC)
  - The $G_x(D) = \boldsymbol{\Phi}^*(D)$ from previous page



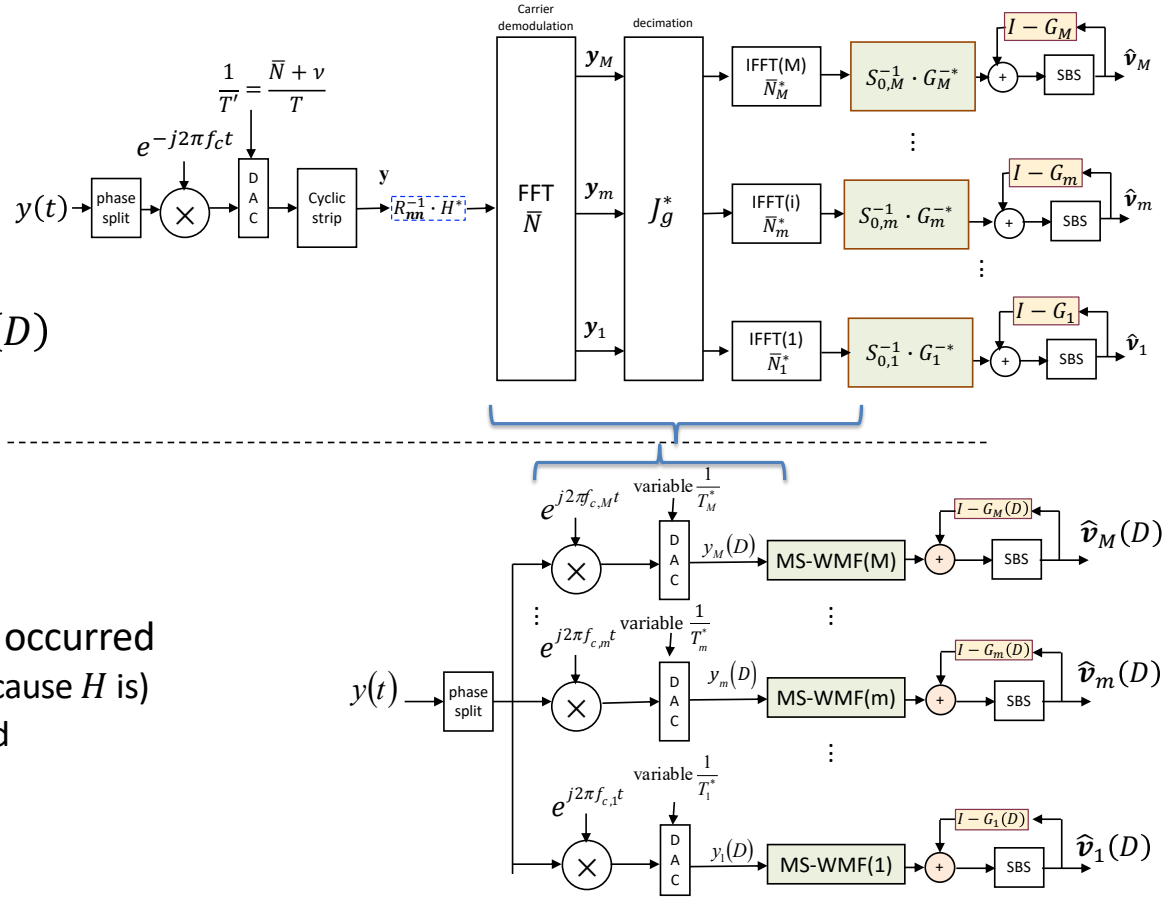
The CDFE Receiver(s)



- Noise whitening and matched filter
  - Can commute if large $\bar{N}$ for each band
  - Close enough to cyclic so absorbed into (green) FF matrix filter

Compare Receivers

- Clear multi-band structure
- Remember GDFE is conditional linear prediction problem
- So $[I \quad \boldsymbol{\phi}_1^* \quad \cdots \quad \boldsymbol{\phi}_{\bar{N}-1}^*] \rightarrow G_i(D)$



- Commutation of MS-WMF has occurred
  - Matched filter is stationary (because H is)
  - $G_i(D)$ is stationary in each band

Revisit optimum 1+.9D^{-1} from L13:12

```
>> [gn,en_bar,bn_bar,Nstar,b_bar]=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
>> 10*log10(2^(2*b_bar) -1) = 7.6247 dB
>> rXX=diag([en_bar(8:-1:1)]) =
   1.2329        0        0        0        0        0        0
        0   1.1916        0        0        0        0        0
        0        0   0.9547        0        0        0        0
        0        0        0        0        0        0        0
        0        0        0        0   0.9547        0        0
        0        0        0        0        0   1.1916        0
        0        0        0        0        0        0   1.2329
        0        0        0        0        0        0        0
>> rXXbar=diag([en_bar(8:-1:6) en_bar(4:-1:1)]) =
   1.2329        0        0        0        0        0        0
        0   1.1916        0        0        0        0        0
        0        0   0.9547        0        0        0        0
        0        0        0   0.9547        0        0        0
        0        0        0        0   1.1916        0        0
        0        0        0        0        0   1.2329        0
        0        0        0        0        0        0   1.2415
>> J=hankel([zeros(1,7),1]);
>> Q=(1/sqrt(8))*J*fft(J);
>> J7=hankel([zeros(1,6),1]);
>> Qtilde=(1/sqrt(7))*J7*fft(J7);
>> ruu=real(Qtilde'*rXXbar*Qtilde); (avoid finite-prec error on imag part)
>> Phibar=lohc(ruu) ;
```

```
>> A=real(Q'*Jg*Qtilde*Phibar) =
   0.9690  -0.0430   0.0265  -0.0400   0.0643  -0.1047   0.2044
   0.3040   0.9208  -0.1373   0.0784  -0.0748   0.0937  -0.1427
  -0.1969   0.4609   0.8251  -0.1903   0.1093  -0.0932   0.1060
   0.1576  -0.2170   0.6189   0.6929  -0.2052   0.1182  -0.0938
  -0.1314   0.1408  -0.2126   0.7640   0.5365  -0.1870   0.1060
   0.1064  -0.0937   0.1084  -0.1770   0.8833   0.3691  -0.1427
  -0.0729   0.0515  -0.0489   0.0606  -0.1068   0.9658   0.2044
  -0.0000  -0.0000  -0.0000  -0.0000  -0.0000  -0.0000   1.0000
>> C=[.9 ; zeros(6,1) ; 1];
>> R=[.9 1 0 0 0 0 0 0];
>> Ht=(1/sqrt(.181))*toeplitz(C,R);
>> [snrGDFEu, GU, WU, S0, MSWMFU] = computeGDFE(Ht, A, 2, 9)
(note use of Nx/Lx = 9 because nu = 1, but also A reduced to 7 dimensions)
snrGDFEu =  7.6247 dB
GU =
   1.0000   0.4654  -0.0309  -0.0024   0.0245  -0.0574   0.4464
        0   1.0000   0.5340  -0.0310  -0.0052   0.0327  -0.2178
        0        0   1.0000   0.5510  -0.0307  -0.0091   0.1120
        0        0        0   1.0000   0.5554  -0.0289  -0.0499
        0        0        0        0   1.0000   0.5549  -0.0102
        0        0        0        0        0   1.0000   0.5555
        0        0        0        0        0        0   1.0000
MSWMFU =
   0.2144   0.0140  -0.0036   0.0019  -0.0022   0.0042  -0.0120   0.1767
   0.0788   0.2646   0.0434  -0.0124   0.0080  -0.0090   0.0157  -0.0972
  -0.0572   0.0191   0.2737   0.0812  -0.0222   0.0152  -0.0179   0.0609
   0.0413  -0.0281  -0.0237   0.2623   0.1233  -0.0296   0.0215  -0.0421
  -0.0318   0.0250  -0.0033  -0.0541   0.2364   0.1663  -0.0321   0.0320
   0.0279  -0.0225   0.0093   0.0179  -0.0731   0.1999   0.2058  -0.0254
  -0.1112   0.0652  -0.0290  -0.0061   0.0501  -0.1141   0.2104   0.1753
>> b' =
   1.8981   1.8022   1.7816   1.7762   1.7752   1.7762   1.6233
>> bbar =   1.3814
```

7 Dimensions

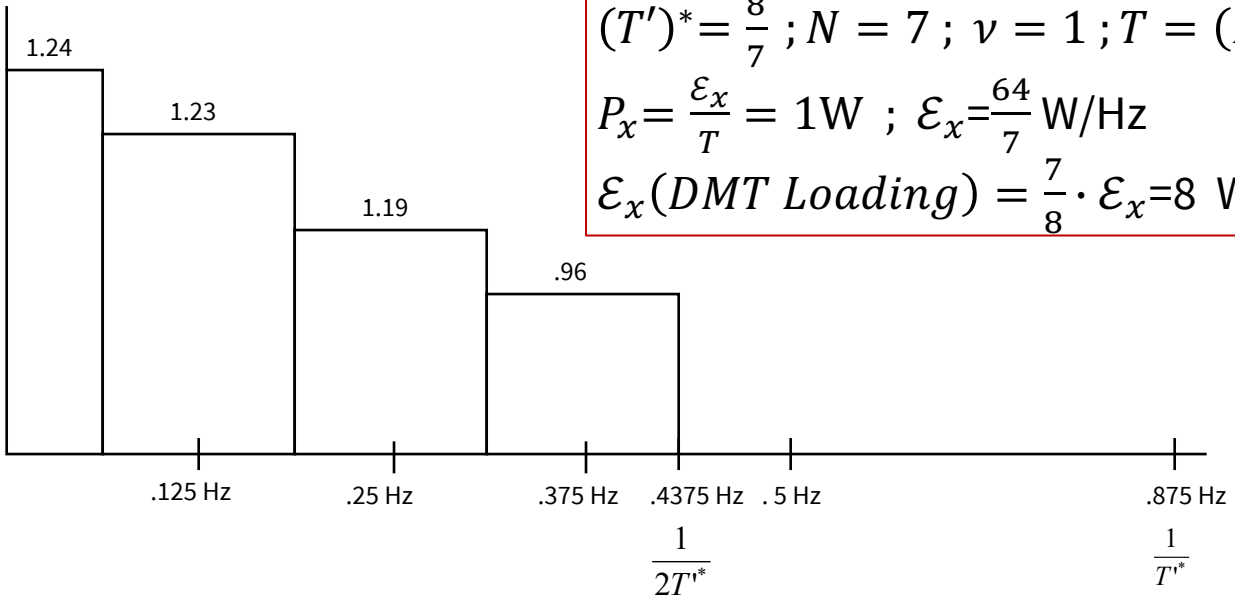
Better Perf (input opt)

Converges On GU

(and WU Not shown)

```
>> Jg = [
   1   0   0   0   0   0   0
   0   1   0   0   0   0   0
   0   0   1   0   0   0   0
   0   0   0   0   0   0   0
   0   0   0   1   0   0   0
   0   0   0   0   1   0   0
   0   0   0   0   0   1   0
   0   0   0   0   0   0   1 ];
```

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



$$(T')^* = \frac{8}{7}\,;\; N = 7\,;\; \nu = 1\,;\; T = (N+\nu)\cdot(T')^* = \frac{64}{7}\,;$$

$$P_x = \frac{\mathcal{E}_x}{T} = 1\text{W}\;;\; \mathcal{E}_x = \frac{64}{7}\ \text{W/Hz}$$

$$\mathcal{E}_x(DMT\ Loading) = \frac{7}{8}\cdot\mathcal{E}_x = 8\ \text{W/Hz}$$

- No more zeroed bands

Resampled Design Commands

```
>> D=exp(j*[0:100]*(7/8)*.01*pi);
>> H7=sqrt(8/7)*(ones(1,101)+.9*D);
>> H7=[H7,conj(H7(101:-1:2))];
>> h7=real(ifft(H7));
>> h=[h7(200:201),h7(1:5)] =
  -0.1011   0.9393   1.1979  -0.0603   0.0394  -0.0292   0.0232
>> H=toeplitz([h(1),h(7:-1:2)]',h) =
  -0.1011   0.9393   1.1979  -0.0603   0.0394  -0.0292   0.0232
   0.0232  -0.1011   0.9393   1.1979  -0.0603   0.0394  -0.0292
  -0.0292   0.0232  -0.1011   0.9393   1.1979  -0.0603   0.0394
   0.0394  -0.0292   0.0232  -0.1011   0.9393   1.1979  -0.0603
  -0.0603   0.0394  -0.0292   0.0232  -0.1011   0.9393   1.1979
   1.1979  -0.0603   0.0394  -0.0292   0.0232  -0.1011   0.9393
   0.9393   1.1979  -0.0603   0.0394  -0.0292   0.0232  -0.1011
>> H=sqrt(1/.181)*H;
J7=hankel([zeros(1,6),1]');
>> Q7=(1/sqrt(7))*J7*fft(J7);
>> rXX=diag([1.23,1.19,.96,.96,1.19,1.23,1.24]);
>> rxx=real(Q7'*rXX*Q7);
>> Phibar=lohc(rxx);
>> A=Phibar;
```

```
>> [snrGDFEu, GU, WU, S0, MSWMFU, b, bbar] = computeGDFE(H, A, 2,
8)

snrGDFEu =   9.1416 dB (higher, but at lower symbol rate)
GU =
   1.0000   0.4783  -0.0492  -0.0074   0.0208  -0.0760   0.4583
        0   1.0000   0.5663  -0.0507  -0.0100   0.0385  -0.2577
        0        0   1.0000   0.5952  -0.0517  -0.0208   0.1470
        0        0        0   1.0000   0.6049  -0.0463  -0.0833
        0        0        0        0   1.0000   0.6042  -0.0105
        0        0        0        0        0   1.0000   0.6074
        0        0        0        0        0        0   1.0000
MSWMFU =
  -0.0173   0.0040  -0.0050   0.0068  -0.0103   0.2054   0.1611
   0.1971  -0.0222   0.0069  -0.0089   0.0125  -0.1032   0.1701
   0.1583   0.2097  -0.0250   0.0095  -0.0127   0.0677  -0.0866
  -0.0808   0.1539   0.2145  -0.0268   0.0118  -0.0445   0.0590
   0.0559  -0.0786   0.1520   0.2166  -0.0283   0.0302  -0.0397
  -0.0401   0.0562  -0.0788   0.1522   0.2163  -0.0258   0.0308
   0.0724  -0.0769   0.0956  -0.1281   0.2194   0.1136  -0.0825

>> b' =
   0.9313   0.8775   0.8700   0.8691   0.8690   0.8697   0.7970
>> bbar =   1.6013
>> R=bbar*(7/8) = 1.4718   (bits/sec) > 1.3814 bits/sec  (same as interp)
```

Also
7 Dimensions

Converges
On GU

(and WU
Not shown)

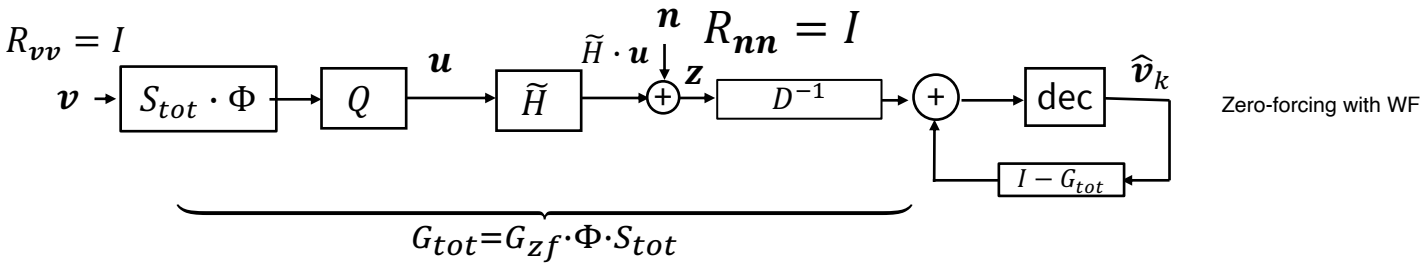
- See also the two-band example in Section 5.3
  - Tedious but could be helpful in following details for a multiband CDFE (e.g. – uplink carrier aggregation with multiple resource blocks in Cellular)

ZF/MMSE convergence conditions

Section 5.3.5

Use Zero Forcing GDFE with Water-fill?

- **To Show:** MMSE GDFE's triangular shaping G equal the triangular factor of $\widetilde{H} = D \cdot G_{zf} \cdot Q^*$?
  - **IF the input is water-fill and nonsingular (so resampled):**
- $D^{-1} \cdot \widetilde{H} = F \cdot \Lambda \cdot M^*$ with energies $diag\{\mathcal{E}\} = K - \Lambda^{-2}$ → $R_{uu} = M \cdot (K - \Lambda^{-2}) \cdot M^* = Q \cdot \Phi \cdot \Phi^* \cdot Q^*$
  - To form this, do Cholesky on: $Q^* \cdot R_{uu} \cdot Q = \Phi \cdot \Phi^*$
  - Define monic triangular: $G_{tot} = G_{zf} \cdot \Phi \cdot S_{tot}$ =and note corresponding NS-WF ZF-GDFE with $D^{-1}$is



Zero-forcing with WF

always	$\widetilde{H} = F \cdot \Lambda \cdot M^* = D \cdot G_{zf} \cdot Q^*$
nonsingular	Cholesky: $Q^* \cdot R_{uu} \cdot Q = \Phi \cdot \Phi^*$

This zero-forcing actually has G_{tot} , not G_{zf}, as feedback; however designers often use flat input $S_{tot} = \overline{\mathcal{E}}_x \cdot I$; Where $K \approx \overline{\mathcal{E}}_x$ which sets $G_{tot} = G_{zf}$ ($M = Q$, so $\Phi = I$)

Equivalently, the input v as $N \to \infty$ has $S_{tot} \to$ constant, so then exactly true.

The two water-fill receivers

Rearrange Water-fill

$$R_{uu}=M\cdot[K\cdot I-\Lambda^{-2}]\cdot M^{*}=Q\cdot\Phi\cdot\Phi^{*}\cdot Q^{*}$$

Noting

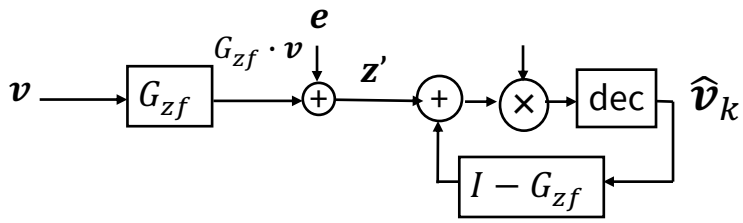
$$M\cdot\Lambda^{-2}\cdot M^{*}=\tilde{H}^{-1}\cdot D^{2}\cdot\tilde{H}^{-*}=R_{f}^{-1}$$

$$K\cdot I=Q\cdot\Phi\cdot\Phi^{*}\cdot Q^{*}+Q\cdot G_{zf}^{-1}\cdot G_{zf}^{-*}\cdot Q^{*}$$

$$K\cdot G_{zf}\cdot G_{zf}^{*}=\underbrace{G_{tot}\cdot S_{tot}^{-2}\cdot G_{tot}^{*}}_{R_f}+I$$

This is R_b^{-1} !

Equivalent MMSE with WF (same $G = G'_{zf}$):
Performance still not same, MMSE slightly better



$G = G_{zf}$ so water-filling leads to MMSE having same feedback as the water-fill zero-forcing, at least the flat-energy approximation to wf

However, G_{tot} , is really the ZF cascade when non flat or for finite symbol length.

- The K calculation needs to be positive (or increase K so slightly positive and then scale down the resulting energies); the water-fill input was not full rank so there is a loss; can be small in wireless
- However, MMSE still has (slightly) higher SNR so use of G_{zf} with waterfill as feedback, not G_{tot}, is highest SNR
  - Note carefully on previous slide that the ZF-GDFE feedforward filter is not the same as MMSE-GDFE, even if feedback same

Worst-Case Noise equates ZF and MMSE

- Easy proof: WCN diagonalizes the primary-user receivers from BC in Chapter 2

  - Step 1: Separates noise-whitened-noise-matched channel's triangular part
    - $\widetilde{H} \triangleq R_{wcn}^{+} \cdot H = R_{zf} \cdot Q_{zf}^{*} = \begin{bmatrix} 0 & R_1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} q \\ Q_1^{*} \end{bmatrix}$

  R_1 is triangular part
  Q_1^{*} is corresponding column set

  - Step 2: cascade that triangular part with Cholesky of rotated input
    - $R_{\tilde{x}\tilde{x}} = Q_1^{*} \cdot R_{xx} \cdot Q_1 = \Phi \cdot \Phi^{*}$ where Φ is triangular Cholesky factor
    - $A = Q_1 \cdot \Phi$

  - Step3: find the channel gains/SNR and feedback section
    - Cascade of receiver triangular inverse $D_A \cdot G_{zf} = R_1 \cdot \Phi$

Example of WCN's RCVR Diagonalization

```
>> H =
   0.9000   1.0000        0        0
        0   0.9000   1.0000        0
        0        0   0.9000   1.0000
H=(1/sqrt(.181))*H;
>> Rxx=eye(4);
>>  [Rwcn,b]=wcnoise(Rxx,H,1);
>> b =   4.8024
>> Htilde=inv(Rwcn)*H;
>> [R,Q]=rq(Htilde);
>> R =
        0  -2.7323  -1.4039   0.0071
        0        0  -2.7077  -1.2346
        0        0        0  -3.0719
Q1=Q(:,2:4);
Rxxrot=Q1'*Rxx*Q1;
Rup=R(:,2:4) ;
Phibar=lohc(Rxxrot);
DA=diag(diag(Rup*Phibar));
>> G=inv(DA)*Rup*Phibar
   1.0000   0.5138  -0.0026
        0   1.0000   0.4559
        0        0   1.0000
>> A=Q1*Phibar;
>> sri=inv(sqrtm(Rwcn));
```

Rwcn is nonsingular here

Finding ZF modulator G

```
>> [snrGDFEu, GU, WU, S0, MSWMFU,b,bbar] = computeGDFE(sri*H, A, 2, 4)
snrGDFEu =   1.1340 dB

GU =
   1.0000   0.5826  -0.0030
        0   1.0000   0.5160
        0        0   1.0000
WU =
   0.1339        0        0
  -0.0676   0.1317        0
   0.0244  -0.0470   0.1031
S0 =
   8.4657        0        0
        0   8.5956        0
        0        0  10.7006
MSWMFU =
  -0.3657  -0.0128   0.0054
  -0.0125  -0.3560  -0.0125
   0.0047  -0.0111  -0.3164
>> MSWMFU*sri' =
  -0.3660  -0.0000   0.0000
  -0.0000  -0.3565   0.0000
   0.0000  -0.0000  -0.3167
>> b' =   1.5408   1.5518   1.7098
>> bbar =   1.2006
>> sum(b) =   4.8024 (checks)
```

Diagonal !
(so ZF = MMSE GDFE)

Only need
RQ fact and lohc

- See Example 5.3.6 with non-white Rxx

Some Final Comments

- The GDFE is canonical – capacity rate is reliably achievable with $\Gamma = 0$ (or capacity less shaping loss)
- GDFE can have error propagation (limited to $\bar{N}$) if $\Gamma > 0$ dB
  - Unless it is VC (~DMT) , which is ML decoder uniquely amoung all GDFEs
  - Other GDFE's becoming increasingly less favorable performance relative to VC/DMT as gap grows
- The DMT form benefits from FFT algorithms so also more cost effective than the others
- By Separation Theorem, Coded-OFDM can capture the DMT benefits also without error propagation
  - But will lose more rapidly lose performance relatively if input is not water filling
- The MMSE-DFE is limiting (stationary) case of the CDFE and can be canonical
  - Set of MMSE-DFE's for each of which PWC holds
  - Has unlimited error propagation (use precoder) and also degrades more rapidly for nonzero-gap codes

Eventual Global Conclusion: Use DMT (wireline) or C-OFDM (wireless) on almost all difficult single-user transmission systems

End Lecture 13

The image part