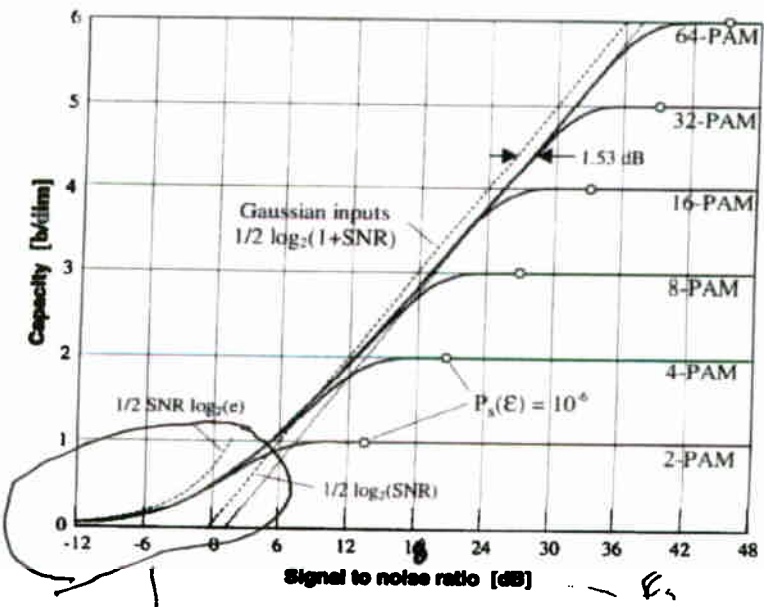
Achieving capacity on the AWGN channel

1) We shall now show how to get close to capacity using practical schemes.

2) First we introduce a powerful error control code that has been shown to get very close to capacity in the range of rates $0 \le R < 1$. We will then show how this code can be used to get very close to capacity for other rates.

3) Separate notes on LDPC



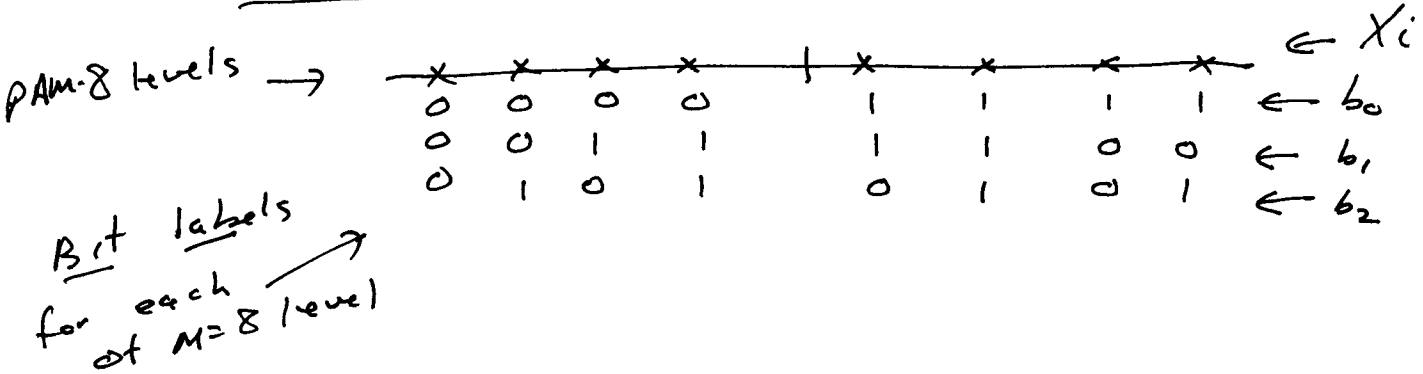
Low density parity check codes get *VERY* close to capacity

4) Achieving capacity for higher rates.

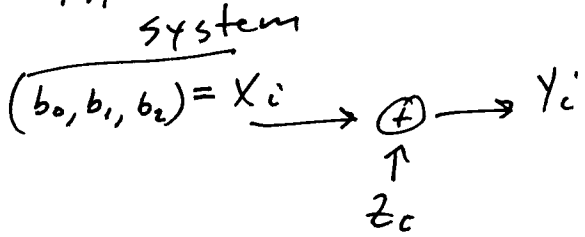
⇒ will integrate good binary codes into PAM modulation.

Motivating example.

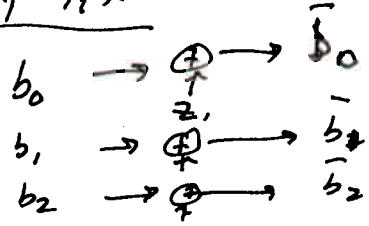
Let $M = 8$:



Note: PAM-8 levels are transmitted across AWGN. We will show that there is an equivalent representation where instead of transmitting PAM symbols, we transmit bits b_0, b_1, b_2



Equivalent binary system

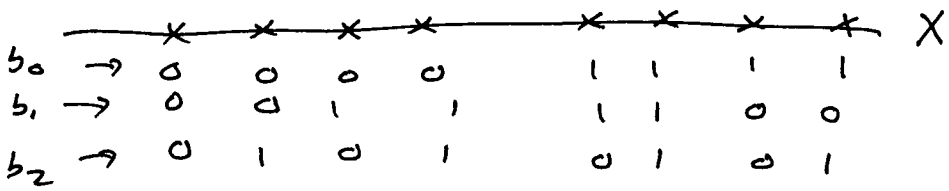


$I(X;Y)$

$\|$ ← pass. bc to show

$I(b_0, \bar{b}_0) + I(b_1, \bar{b}_1) + I(b_2, \bar{b}_2)$

Note: In the following system



b_0 is transmitted across a channel with relatively high SNR. Why? Note that in general we will recover b_0 relatively easy when PAM symbol is sent

b_1 is transmitted across a channel with moderate SNR

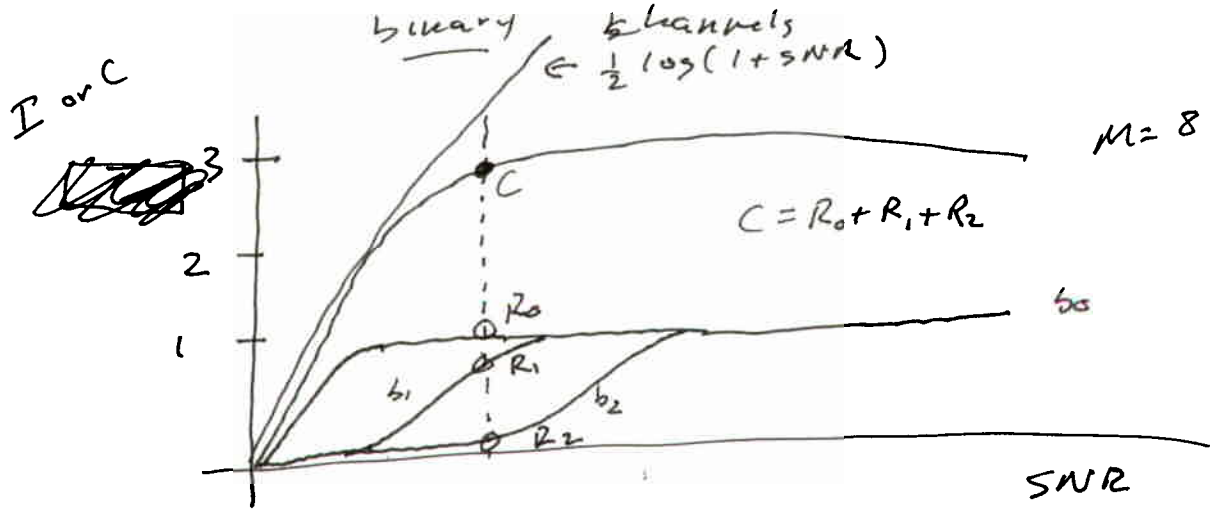
b_2 is transmitted across a channel with relatively low SNR (b_2 is the most unreliable bit.

3 binary channel

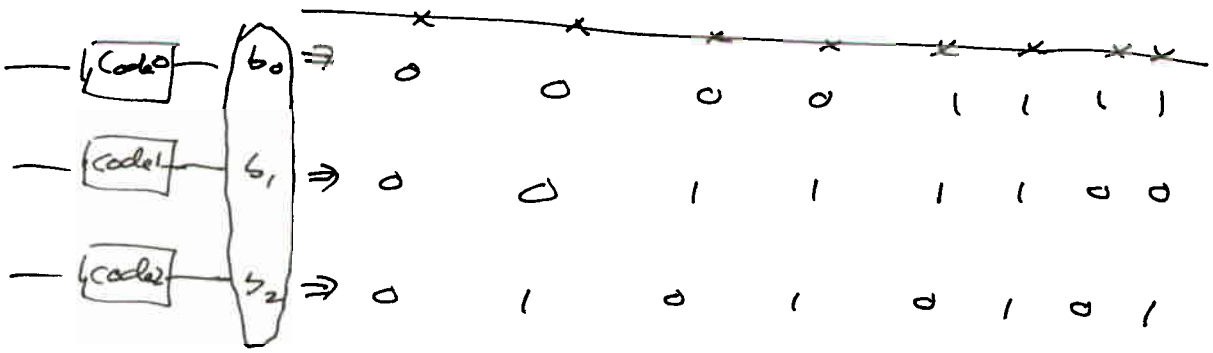
PAM channel

Can be shown that $I(X;Y) = I(b_1, \bar{b}_1) + I(b_2, \bar{b}_2) + I(b_0, \bar{b}_0)$

$\Rightarrow$ Optimal signalling for PAM channel means optimal signalling on 3 parallel binary channels



So: For desired SNR we need to code close to capacity on 3 binary channels



Code 0 ⇒ LDPC at rate R_0
Code 1 ⇒ LDPC at rate R_1
Code 2 ⇒ LDPC at rate R_2

} result is a 8-PAM system that is close to capacity!!

⇒ This is theory it is very easy now to get 8-PAM codes (and other PAM, QAM) to set close to C.

⇒ There are other ways beside the one presented here.

ECE6605 Notes:

Low density parity check codes

Steven W. McLaughlin
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332

December, 2003

Agenda

- **Background: linear block codes**
- **Low density parity check codes**
- **Hard decision decoding**
- **Soft decision decoding:** probability and log domain

Low density parity check codes

- **Gallager** 1962: what's old is what's new
- **MacKay** (Cambridge), mid-late 90's rediscovered them
- **Spielman** (MIT), **Luby** (Berkeley, DigitalFountain) mid-90's, rediscovered hard versions of them
- **Best performing** code: irregular LDPC, blocklength 10^6, R=1/2, within 0.07 dB of capacity
- **Highly parallel decoder**, implemented using table lookup and adds
- **Best applications:** long block lengths (>4k), high data transfer rate

Background: Binary Linear Block Codes

- **Encoding, assume bits**

- **G is generator k x n matrix (e.g. 3 x 7)**

$$G = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

- **G = [I | P] in systematic form**

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

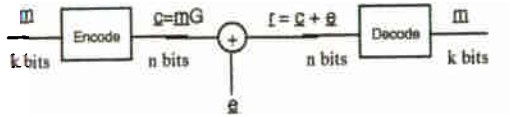
- **Cols of G give the parity checks**

$$\underline{\mathbf{c}} = \underline{\mathbf{m}}\,\mathbf{G} = [m_0\, m_1\, m_2] \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= [\underline{\mathbf{m}}\ \underline{\mathbf{p}}] \qquad \text{where } \underline{\mathbf{p}} = [p_0\, p_1\, p_2\, p_3]$$

$$\begin{aligned} p_0 &= m_0 + m_1 + m_2 \\ p_1 &= \quad m_1 + m_2 \\ p_2 &= m_0 \quad\ + m_2 \\ p_3 &= m_0 + m_1 + m_2 \end{aligned}$$

Decoding



- **Use parity check matrix H**
  - H is an *(n-k) x n* matrix
  - H chosen such that $GH^T = \underline{0}$
  - For any $\underline{c}$, $\underline{c}H^T = \underline{0}$

- **Examples of H**

$$(c_0\ c_1\ c_2\ c_3\ c_4\ c_5\ c_6)$$

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

- **Rows of H are called *parity checks***

  i'th row: which code bits involved i'th parity check

  j'th col: which parity checks involve bit c_j

p_0: $0 = c_0 + c_1 + c_3$

p_1: $0 = c_1 + c_2 + c_4 + c_5 + c_6$

p_2: $0 = c_3 + c_5$

p_3: $0 = c_1 + c_2 + c_4 + c_6$

- **Systematic codes**

$$H = [P^T \mid I]$$

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

-

Overview of LDPC codes

• **Codeword length n and message length k are large (>10000)**

• **Definition of LDPC is based on parity check matrix H:**
- matrix is sparse and randomly chosen

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad t = 2$$

- special case: column weight t is fixed and small (regular LDPC)

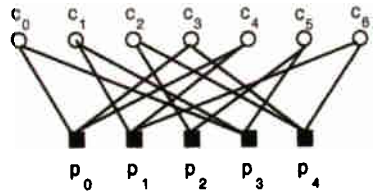
• **Comments**
- Sound familiar?
- Shannon's random coding argument: choose codewords completely at random (with replacement), and the average code is good. The codes are *nonlinear* (they do not in general have G and H matrices)
- LDPCs have high degree of randomness, but still remain linear. They are defined in terms of G and H.

• **Construction of H**
- Sparsity → less than 1% of H has 1's, rest are 0's
- Very easy to construct, need only a random number generator
- Have been shown to get extremely close to capacity of BSC and binary AWGN channels

Decoding?
- If n is large (10,000) and rate is 1/2, number of parity checks (n-k) is 5000. The syndrome $\underline{s}$ is of length n-k, so there are 2^{5000} different syndromes. Cannot use a conventional syndrome based decoder
- Need some other more practical decoder
- Decoder used: *message passing decoder.*

• **Factor graph representation of H:**

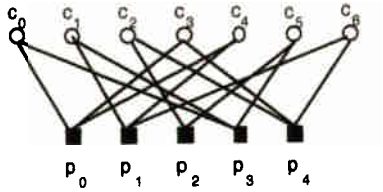
$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$



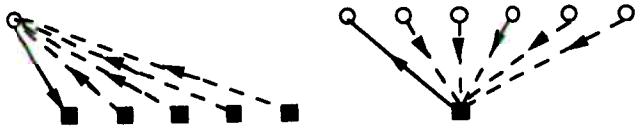
p_0: $0 = c_0 + c_3 + c_4$
p_1: $0 = c_1 + c_4 + c_6$
p_2: $0 = c_2 + c_5$
p_3: $0 = c_0 + c_1 + c_5$
p_4: $0 = c_2 + c_3 + c_6$

Decoding of LDPC codes:
Very basics

- **Decoder given $\underline{r} = \underline{c} + \underline{e}$**
- **$\underline{e}$ is AWGN or bits**
- **Use factor graph to decode:**



- **Each check node and bit node is a processor**
- **Pass messages about what each processor computes**



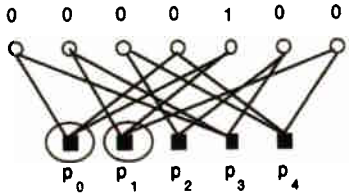
How reliable is the bit given other checks?

How reliable is the parity check?

Decoding of LDPC codes:
<u>Hard channel outputs</u> and a sequential decoder

- **Decoder given $\underline{r} = \underline{c} + \underline{e}$ $\underline{e}$ is bits**
- **Use factor graph to decode:**

assume $\underline{c} = \underline{0}$ and $\underline{r} = 0\,0\,0\,0\,1\,0\,0$



- **A simple sequential decoder**

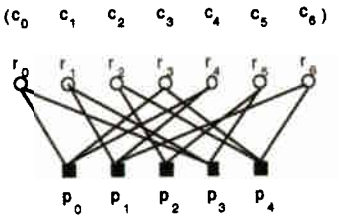
i) identify which parity checks are not satisfied
ii) for each bit node,
  - how many unsatisfied parity checks is it involved in?
  - if > t/2, flip that bit

iii) if all parity checks satisfied, terminate. If not, goto i)

- **If all parity checks are satisfied, decoder is done.**
- **Decoder and codes perform remarkably well**

Decoding of LDPC codes:
Soft outputs and APP decoding

- **Decoder given $\underline{r} = \underline{c} + \underline{e}$, $\underline{e}$ are IID GRV**



- **Infer $\underline{c}$ from $\underline{r}$**

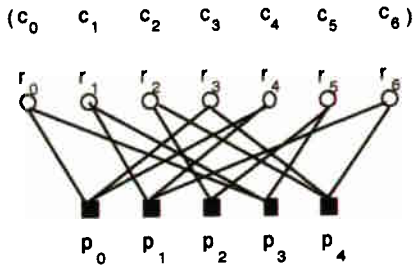
- **Maximum Aposterior (MAP) estimate**

  - compute: $p(c_i = 0|\ \underline{r})$ and $p(c_i = 1|\ \underline{r})$
  - choose largest
  - valid decoding requires parity checks

- **Rewrite aposteriori probability (APP)**

$$p(c_i = 0|\ \underline{r}) = k\ p(\underline{r}\ |\ c_i = 0)\ p(c_i = 0)$$

Decoding LDPC codes:
Message passing algorithm



- **Message passing algorithm:** Converges to true MAP decoder *only if the factor graph has no cycles*

- **LDPCs have cycles,** no way to avoid them

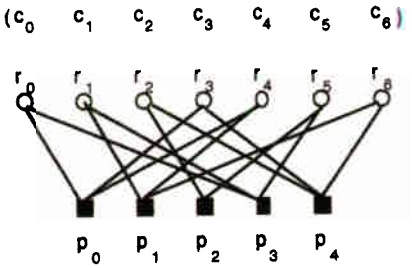
$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

  - easy to eliminate cycles of length 4.

- **Short cycles hurt performance**

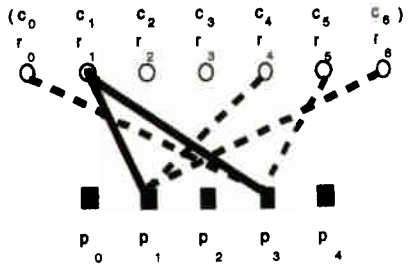
- **Decoder works remarkably well. See longer tutorial for complete derivation.**
  - assumes code bits are independent of each other
  - assumes code is a tree: *for few iterations this is valid*

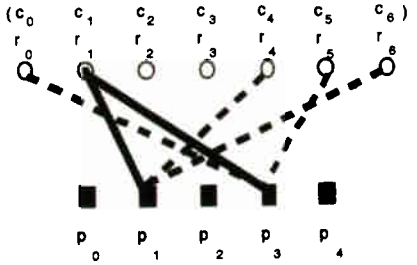
More on message passing algorithm



- **Definitions:**

N(m): set of codeword bits in parity check m
M(n): set of parity checks for codeword bit n
q^x_{in}: probability that $c_n = x$ given information from parity checks other than i.
q^x_n: probability that $c_n = x$ given $\underline{r}$.





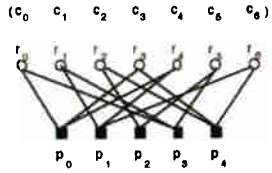
- **Most important idea:** Consider c_j

APP: $q^0_j = p(c_j = 0 | \underline{r})$

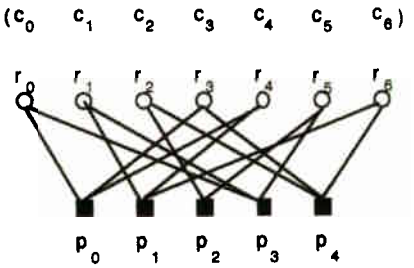
$$q^0_j = K\, p(r_j | c_j = 0) \prod_{i \in M(j)} \left[1 + \prod_{\substack{n' \in N(i) \\ n' \neq j}} (q^0_{in'} - q^1_{in'}) \right]$$

- for each of the parity checks connected to c_j we want to know how likely that parity check is satisfied if c_j=0.

- Consider the *sign* of the rightmost product: if a term is positive then more likely to be 0. Note an even # must be zero to satisfy parity



Iterative APP algorithm



- **Initialization**:** $q^0_{ij} = p(r_j|\, c_j=0)$, $q^1_{ij} = p(r_j|\, c_j=1)$
- **Iteration L:**

for j=0:n-1

$$q^0_j = K'' \; p(r_j|c_j = 0) \prod_{i \in M(j)} \left[1 + \prod_{\substack{n' \in N(i) \\ n' \neq j}} (q^0_{in'} - q^1_{in'}) \right]$$

repeat for q^1_j

normalize $q^0_j + q^1_j = 1$;

for m=1:n-k

$$q^0_{mj} = q^0_j \Big/ \left[1 + \prod_{\substack{n' \in N(m) \\ n' \neq j}} (q^0_{mn'} - q^1_{mn'}) \right]$$

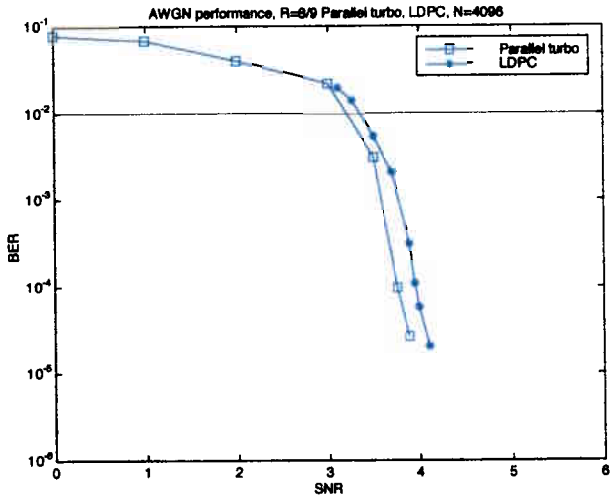
repeat for q^1_{mj}

normalize $q^0_{mj} + q^1_{mj} = 1$;

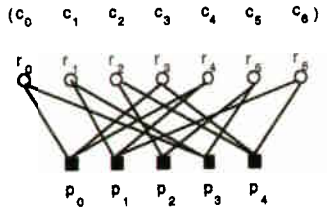
end

end

Performance of LDPCs vs Serial Turbo



Log-domain algorithm



- **Initialization:** $\log \lambda_j / \lambda_j = 2r_j / s^2$
- **Iteration L:**

for j=0:n-1

$$LLR(q^0_j) = 2r_j/\sigma^2 + \sum_{m \in M(j)} s_{mj}\, \Psi(A_{mj})$$

for m=1:n-k

$$LLR(q^0_{mj}) = LLR(q^0_j) - s_{mj}\, \Psi(A_{mj})$$

end

end

- Where $\Psi(x) = \log(|\tanh(x/2)|) = \log(|e^x-1/e^x+1|)$

$$A_{mj} = \sum_{\substack{n' \in N(m) \\ n' \neq j}} \Psi(LLR(q^0_{mn'}))$$

$$s_{mj} = \prod_{\substack{n' \in N(m) \\ n' \neq j}} \mathrm{sgn}\,(LLR(q^0_{mn'}))$$

More comments

- Code design is a real issue: we have described the decoder only. The encoder can be very complex.

- Encoder can be found directly by putting H into systematic form. This is done by row reduction and column reordering of H. The primary problem with this is that the resulting H is likely to be dense (contains many ones). This makes the encoder very complex.

- Gallager showed in his original paper that the minimum distance of the ensemble of LDPC codes grows linearly with blocklength with high probability. However, randomly constructed, moderate length, high rate codes can have small distance.

- There is considerable recent work on non-random constructions.