



EE 5654 - Digital Communications Spring 2005

Instructor: R. Michael Buehrer

Lecture #19 - Block Codes: Performance Evaluation
and Tradeoffs





Error Correction Codes

- Names:
 - Error Correction Coding
 - Channel Coding
 - Forward Error Correction
- Channel coding provides a practical way to achieve performance which more closely approaches theoretical capacity.
- Channel coding trades bandwidth efficiency for energy efficiency, just like modulation
 - However, trade is more efficient



Encoding and Decoding of Error Correction Codes

- Encoding operation usually easy
 - can be thought of in terms of parity checks.
- Decoding operation is usually complex.
 - This is the key to designing good codes – finding codes which provide good performance *and* are easy to decode!
- Many good practical codes have been developed for use in communication systems. We focus on block codes first.



Terminology for Block Codes

- k = number of input symbols (usually bits)
- n = number of output symbols (usually bits)
- $r = \frac{k}{n} \leq 1$ is the code rate
- $d_{H,\min} = \min_{i \neq j} \{d_H(\underline{c}_i, \underline{c}_j)\}$ is the minimum distance
- $t = \left\lfloor \frac{d_{H,\min} - 1}{2} \right\rfloor$ is the error correcting capability
- Codes can be classified as linear, systematic, and cyclic



Encoding of Block Codes

- Encoding can be viewed in terms of adding parity bits to the information bits.
- This operation can be described in terms of matrix multiplication: $\underline{c} = \underline{m}\mathbf{G}$

where

$\underline{m} = [m_0 \quad \dots \quad m_{k-1}]$ = vector of k input data bits

$\underline{c} = [c_0 \quad \dots \quad c_{n-1}]$ = codeword of n output data bits

\mathbf{G} = $k \times n$ Generator Matrix

- Only requires kn multiplies and additions



The Generator Matrix G

- Let the message bits be a vector $\mathbf{m} = [m_0 \ m_1 \ \cdots \ m_{k-1}]$
- Any linear code can be represented by a *Generator Matrix*

$$\mathbf{G} = \begin{bmatrix} \underline{\mathbf{g}}_0 \\ \underline{\mathbf{g}}_1 \\ \vdots \\ \underline{\mathbf{g}}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix}$$

- The row vectors \mathbf{g} form a *linearly independent* basis for the code \mathcal{C}



Encoding with a Generator Matrix

- The transmitted codeword is generated by the equation
 - $\mathbf{c} = \mathbf{mG}$, where
 - \mathbf{m} is a k element row vector
 - \mathbf{c} is an n element row vector
 - note that \mathbf{c} can be expressed as a linear combination of row vectors

$$\mathbf{m} \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = m_0 \mathbf{g}_0 + m_1 \mathbf{g}_1 + \cdots + m_{k-1} \mathbf{g}_{k-1}$$

Systematic Representation of Codes

- We can create a generator matrix \mathbf{G} that generates an equivalent code if we
 - permute any rows of \mathbf{G}
 - replace any row of \mathbf{G} by a linearly independent combination of rows
- It is always possible to find a systematic representation via row operations:

$$\mathbf{G} = [\mathbf{P} | \mathbf{I}_k] = \left[\begin{array}{cccc|cccc} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} & 1 & 0 & \cdots & 0 \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & 0 \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 0 & \cdots & 0 & 1 \end{array} \right]$$

Example: Generator Matrix for (7,4) Hamming Code

- Note: We have permuted the order of the information and parity check bits from table to correspond with the convention used in Proakis for the Generator Matrix - this is still the same code.

$$\mathbf{G} = \left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right]$$

- Example $\underline{c} = [1001011] = [1011] \cdot \mathbf{G}$

Equivalent Representation

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



Parity Check Matrices

- For each Generator Matrix \mathbf{G} , it is possible to find a corresponding parity check matrix \mathbf{H}
- For a systematic representation:

$$\mathbf{H} = [\mathbf{I}_{n-k} | \overline{\mathbf{P}}']$$

$$= \left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & \overline{p}_{0,0} & \overline{p}_{1,0} & \cdots & \overline{p}_{k-1,0} \\ 0 & 1 & & \vdots & \overline{p}_{0,1} & \overline{p}_{1,1} & \cdots & \overline{p}_{k-1,1} \\ \vdots & & \ddots & 0 & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & \overline{p}_{0,n-k-1} & \overline{p}_{1,n-k-1} & \cdots & \overline{p}_{k-1,n-k-1} \end{array} \right]$$

Parity Check Matrix for (7,4) Hamming Code

$$\mathbf{G} = \left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right]$$

$k \times n$
 4×7

$$\mathbf{H} = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array} \right]$$

$n-k \times n$
 3×7



Decoding of Block Codes

- **Syndrome Decoding**
 - only useful for short codes.
- **Bounded Distance Decoding**
 - Berlekamp-Massey Algorithm
 - Can be used on a large class of linear, *cyclic* codes.
 - Correctly decodes all words which lie within distance t of the correct codeword. May simply fail on some received vectors which are not within distance t of any codeword.
 - Complexity of the decoding is proportional to both block length n and $n - k$. This complexity is a limiting factor in the choice of codes.



Syndrome Decoding of Block Codes

- Let $\mathbf{r} = \mathbf{c} + \mathbf{e}$ be the vector of n received bits
 - \mathbf{e} is sometimes called the *error vector*
- Then \mathbf{s} is the *syndrome* of the received vector, where
$$\mathbf{s} = \mathbf{r}\mathbf{H}' = (\mathbf{c} + \mathbf{e})\mathbf{H}' = \mathbf{c}\mathbf{H}' + \mathbf{e}\mathbf{H}' = \mathbf{e}\mathbf{H}'$$
 - Note that \mathbf{c} is orthogonal to \mathbf{H}'
- Each syndrome corresponds to a unique error pattern when \mathbf{r} is within the error correction capability of a codeword
- Problem: There will be 2^{n-k} (binary case) syndromes in a lookup table
 - For cyclic codes, we will compute a syndrome polynomial

Syndrome Decoding of (7,4) Hamming Code

s			e						
0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	1	0
0	1	1	0	0	0	0	1	0	0
1	1	0	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0



Notes on Berlekamp-Massey Algorithm

- It does not always converge if error correction capability of code is exceeded
- It is particularly efficient to implement this algorithm in a Linear Feedback Shift Register configuration
- The complexity of the location polynomial computation is proportional to t^2
- This algorithm applies to decoding of BCH and Reed-Solomon codes with the following simplifications



Probability of Codeword Error

- We wish to compute the probability $P_c(\varepsilon)$ that a bounded distance decoder will fail
- The decoder can correct up to, but not more than $t = \lfloor (d_{H,\min} - 1)/2 \rfloor$ errors
- We assume that the probability of an individual symbol error is ρ , and that symbol errors occur independently
- The symbol error probability ρ is determined from the modulation type

Probability of Codeword Error (continued)

- If we send n bits, the probability of receiving a specific pattern of i errors and $n-i$ correct bits is:

$$p^i \cdot (1-p)^{n-i}$$

- There are $\binom{n}{i} = \frac{n!}{i! \cdot (n-i)!}$ distinct patterns of n bits

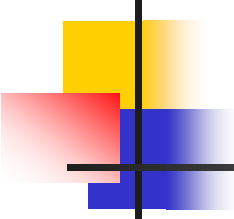
with i errors and $n-i$ correct bits, so the total probability of receiving a pattern with with i errors is:

$$\binom{n}{i} p^i \cdot (1-p)^{n-i}$$

Probability of Codeword Error (continued)

- Since we can correct any pattern of up to t errors, the overall probability of codeword error is:

$$P_C(\varepsilon) = 1 - \sum_{i=0}^t \binom{n}{i} p^i (1-p)^{n-i} = \sum_{i=t+1}^n \binom{n}{i} p^i (1-p)^{n-i}$$



Example: Error Probability of (7,4) Hamming Code

- Assume we are using a (7,4) Hamming Code ($t=1$).
- Assume $p=0.001$
- There are fewer terms, so it is easiest to compute the summation:

$$\begin{aligned}P_C(\varepsilon) &= 1 - \sum_{i=0}^t \binom{n}{i} p^i (1-p)^{n-i} = 1 - \sum_{i=0}^1 \binom{7}{i} (0.001)^i (0.999)^{7-i} \\&= 1 - 1 \cdot (0.999)^7 - 7(0.001)^1 (0.999)^6 \\&= 1 - 0.993 - 0.00696 = 2 \times 10^{-5}\end{aligned}$$



Numerical Evaluation of $P_c(\varepsilon)$

- May need to use higher numerical precision if we are evaluating the form:

$$P_c(\varepsilon) = 1 - \sum_{i=0}^t \binom{n}{i} p^i (1-p)^{n-i}$$

- In general, $\binom{n}{i}$ can be very large and $p^i (1-p)^{n-i}$

can be very small. You may need to evaluate them jointly in order to avoid overflow or underflow (Matlab is pretty good about avoiding this)

- Frequently the term $i=t+1$ and the first few terms thereafter are the most significant



Matlab Functions

- For evaluating $n!$: fact.m

```
function y=fact(n)
```

```
y=1;
```

```
for i=1:n y=y*i; end;
```

- For evaluating $\binom{n}{i}$: binom.m

```
function y = binom(n,i);
```

```
y=fact(n)/(fact(i)*fact(n-i));
```



Example:

- Find the error probability of a (63,45) BCH code with $t=3$ for $p=0.001$.
 - 1 term:
 - $P_c=0; p=0.001;$
 - for $i=4:4$
 - $P_c=P_c+\text{binom}(63,i)*p^i*(1-p)^{(63-i)};$
 - end;
 - $P_c = 5.6152e-007$
 - 2 terms:
 - $P_c=0; p=0.001;$
 - for $i=4:5$
 - $P_c=P_c+\text{binom}(63,i)*p^i*(1-p)^{(63-i)};$
 - end;
 - $P_c = 5.6815e-007$
 - 3 terms: $P_c = 5.6822e-007$



An Important Point about E_b/N_o

- We frequently want to evaluate performance in terms of E_b/N_o
- When using coding, we send extra bits which contain no information at all. In order to make a fair comparison with uncoded systems, we must penalize ourselves by the extra energy used to send those bits.
- We will need to replace E_b/N_o by $r E_b/N_o$ in all our error formulas for different modulation types



Example

- Suppose BPSK modulation is employed and we have $E_b/N_o = 10dB$. Find the probability of error both for an uncoded system and for a system with a (63,45) BCH code:

- Uncoded System: $P_b(e) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) = 3.8794e - 006$

- Coded System: $p = Q\left(\sqrt{\frac{2rE_b}{N_0}}\right) = 7.8625e - 005$

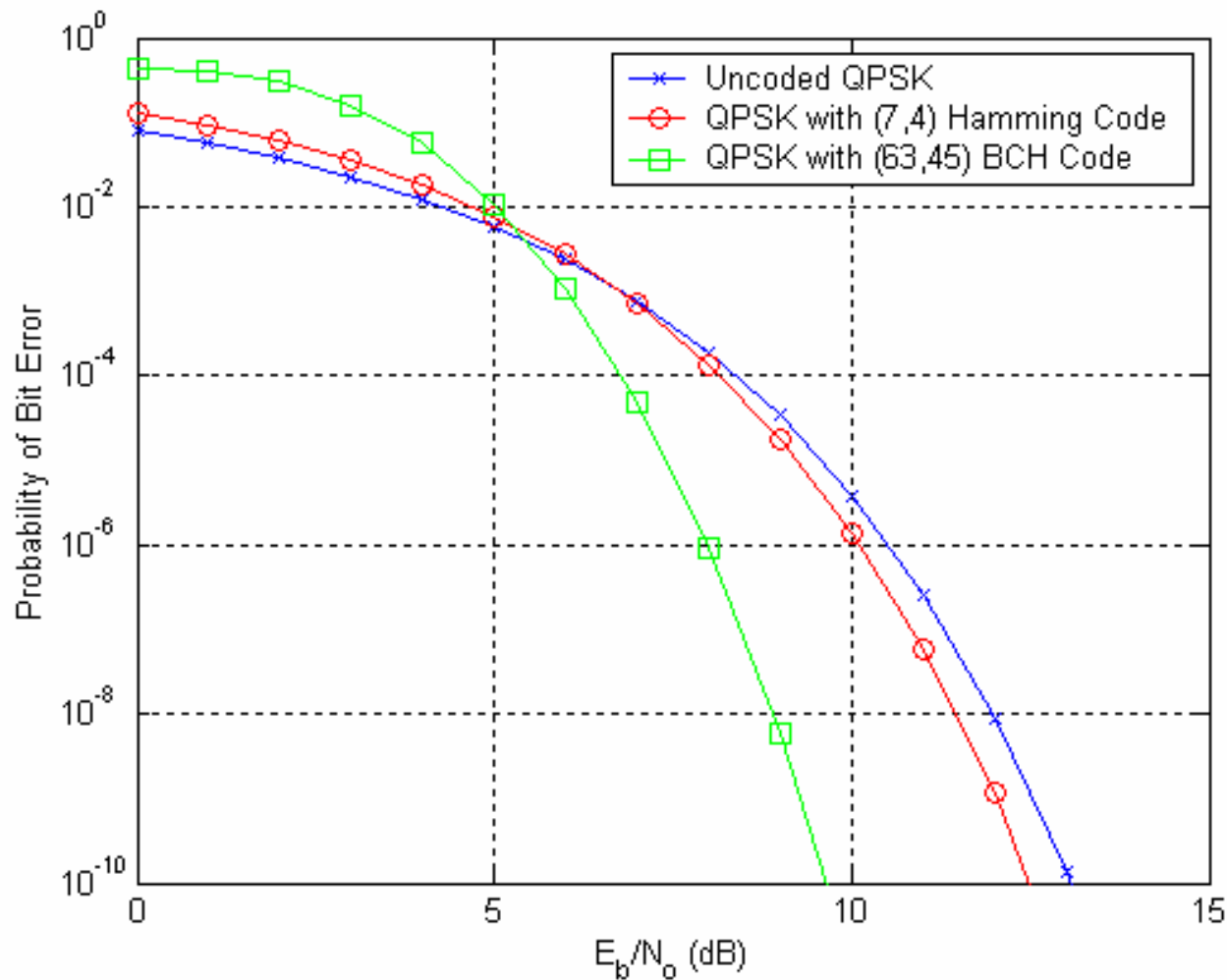
$$P_C(\varepsilon) \approx \sum_{i=4}^8 \binom{63}{i} (7.86 \times 10^{-5})^i (1 - 7.86 \times 10^{-5})^{63-i}$$
$$= 2.2679e - 011$$



Relating Codeword Error Rate and Bit Error Rate

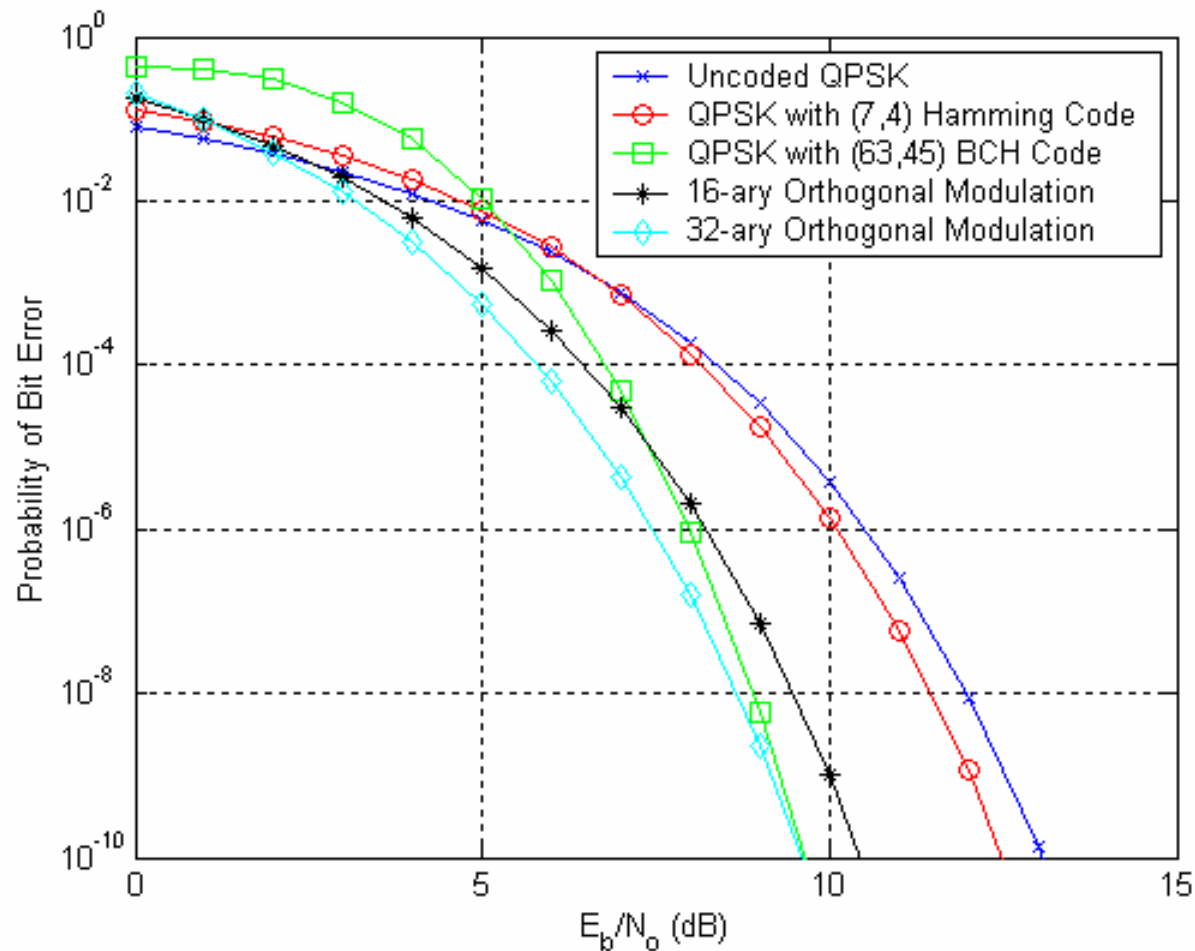
- If the codeword is correctly received, all bits will be correctly received.
- Note that the probability of receiving a block of 45 uncoded bits with no errors is:
$$1 - (1 - 3.8794e-006)^{45}$$
$$\text{ans} = 1.7456e-004$$
- If a codeword is incorrectly decoded, a good approximation is that 1/2 of the bits will be in error.
- More exact analytical evaluation of bit error rate is tedious for block codes.

Example



- **(7,4) Hamming Code is very weak**
- **BCH Code much stronger**
- **For sufficiently low E_b/N_0 coding *degrades* performance**

Example (cont.)



- **Bandwidth expansion is $7/4=1.8$ and $63/45=1.4$**
- **To get equivalent performance with modulation we must increase the bandwidth by a factor of $16/4=4$ or $32/5=6.4$**