Elementary Coding Theory
The message is a binary string \((m\text{-tuple})\)

The code word is also a binary string \((n\text{-tuple})\)
Errors

- Error - change in some of the bits in the code word
- Single error - change in only one bit of a code word
• Add a parity check bit

• Message word + check bit = code word

<table>
<thead>
<tr>
<th>Message Word (m-tuple)</th>
<th>Even Parity Check (n-tuple)</th>
<th>Odd Parity Check (n-tuple)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 000</td>
<td>000 000 0</td>
<td>000 000 1</td>
</tr>
<tr>
<td>110 000</td>
<td>110 000 0</td>
<td>110 000 1</td>
</tr>
<tr>
<td>110 111</td>
<td>110 111 1</td>
<td>110 111 0</td>
</tr>
</tbody>
</table>
• Adding a parity check bit allows the detection of all single errors

• All single errors result in an error indication

<table>
<thead>
<tr>
<th>Received 7-tuple</th>
<th>Decoded Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>001 000 1</td>
<td>001 000</td>
</tr>
<tr>
<td>101 010 0</td>
<td>Parity error</td>
</tr>
<tr>
<td>111 111 0</td>
<td>111 111</td>
</tr>
<tr>
<td>111 111 1</td>
<td>Parity error</td>
</tr>
</tbody>
</table>
Parity

- Even (or odd) parity checking is sufficient for most computer purposes
- Limitations:
  - Cannot detect some multiple errors
  - Cannot correct any errors

110 010 1 Code word

111 000 1 Code word
Maximum Likelyhood Decoding

• Assume transmission errors:
  – are rare
  – occur independently in each bit

• Therefore, 2 errors occur less frequently than 1, 3 errors occur less frequently than 2, etc.

• Maximum likelyhood decoding
  – Look for code word that was most likely transmitted
The messages are either 0 or 1

<table>
<thead>
<tr>
<th>Message</th>
<th>Code Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>111</td>
</tr>
</tbody>
</table>

Difference Matrix shows the number of bits a given 3-tuple is different from a code word

<table>
<thead>
<tr>
<th>Code Word</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>111</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Simplest Error Correcting Code (cont.)

- Encoding

<table>
<thead>
<tr>
<th>Message</th>
<th>Code Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>111</td>
</tr>
</tbody>
</table>

- For single error correction, select closest code word from difference matrix
Enhanced Error Detection

Encoding

<table>
<thead>
<tr>
<th>Message</th>
<th>Code Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>111</td>
</tr>
</tbody>
</table>

Alternatively, can detect up to two errors

But error correction then becomes impossible

111 $\sim$ 100 (transmission error)

Don’t know if 111 or 000 was transmitted
The ability of a code to detect or to correct errors depends solely on its set of code words

Suppose 1100 and 0100 are code words in some code

1100 ↛ Error in bit one ↛ 0100

Received code would be decoded as an erroneous message

Suppose 1100 and 0101 are code words in some code

1100 ↛ Error in any single bit ↛ Can never be 0101
Hamming Distance

Let $a$ and $b$ be binary $n$-tuples. The number of places in which $a$ and $b$ differ is called the *Hamming distance* between $a$ and $b$. The Hamming distance between tuples of different length is undefined.

$$H(a, a) = 0$$

If $H(a, b) = 0$, then $a = b$
Metric Properties

\( a, b, c \in N_2^n \) (binary \( n \)-tuples)

- \( H(a, b) \geq 0 \)
- \( H(a, b) = 0 \iff a = b \)
- \( H(a, b) = H(b, a) \)
- \( H(a, c) \leq H(a, b) + H(b, c) \)
Consider a code whose code words are in $N_2^n$. The minimum distance, $d$, for the code is the minimum of Hamming distances $H(a,b)$ where $a$ and $b$ are distinct code words.

If $d = 1$, then the code cannot detect all transmission errors.

If $d = 2$, then the code can detect but not correct all single errors.

If $d \geq 3$, then the maximum likelyhood decoding scheme can correct all single errors.
Error Correcting Example

\[ c_1 = 00000, \; c_2 = 01110, \; c_3 = 10111, \; c_4 = 11001 \]

\[ d = ? \]

Received 5-tuple = 11111 = r

<table>
<thead>
<tr>
<th>( c_i )</th>
<th>( H(r, c_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>5</td>
</tr>
<tr>
<td>01110</td>
<td>2</td>
</tr>
<tr>
<td>10111</td>
<td>1</td>
</tr>
<tr>
<td>11001</td>
<td>2</td>
</tr>
</tbody>
</table>

\( c_3 \) is the unique code word with minimum distance.
A group is a mathematical structure consisting of a set and an operation, 
\([A, \cdot]\) with the following properties:

- For all \(a, b \in A\), \(a \cdot b \in A\) (closure)
- For all \(a, b, c \in A\), \((a \cdot b) \cdot c = a \cdot (b \cdot c)\) (associativity)
- There exists \(e \in A\) such that for all \(x \in A\), \(e \cdot x = x = x \cdot e\) (identity)
- For all \(x \in A\) there exists \(y \in A\) such that \(x \cdot y = e = y \cdot x\) (invertibility)
Group Codes

Group codes facilitate the construction of error correcting codes.

A code whose code words are binary $n$-tuples is a group code if the sum in $N^n_2$ of any two code words is again a code word.

The addition is a component-wise mod 2 addition

<table>
<thead>
<tr>
<th>$+$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

If $c$ is a code word, then $c + c = \mathbf{0}$ (where $\mathbf{0}$ is the element of $N^n_2$ consisting of all zeros.)
The weight of a binary $n$-tuple $a$ is the number of 1s in the $n$-tuple.

$W(1101) = 3$, $W(10001) = 2$, $W(111) = 3$, $W(00000) = 0$

$W(a) = H(a, 0)$

$H(a, b) = W(a + b)$

Let $d$ be the minimum distance for a group code. Then $d$ also equals the minimum of the weights of all code words except $0$. 
### Multiplication Mod 2

<table>
<thead>
<tr>
<th>× 2</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Let $H$ be an $n \times r$ binary matrix.

Suppose that the code words for a code consist of all binary $n$-tuples $c$ such that $c \cdot H = 0_r$. $c \in N^n_2$, and $0_r \in N^r_2$. 
Parity Check Matrices

Example:

\[
H = \begin{bmatrix}
1 & 1 \\
1 & 0 \\
0 & 1
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>3-tuple</th>
<th>(c \cdot H)</th>
<th>Code word?</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>00</td>
<td>yes</td>
</tr>
<tr>
<td>001</td>
<td>01</td>
<td>no</td>
</tr>
<tr>
<td>010</td>
<td>10</td>
<td>no</td>
</tr>
<tr>
<td>011</td>
<td>11</td>
<td>no</td>
</tr>
<tr>
<td>100</td>
<td>11</td>
<td>no</td>
</tr>
<tr>
<td>101</td>
<td>10</td>
<td>no</td>
</tr>
<tr>
<td>110</td>
<td>01</td>
<td>no</td>
</tr>
<tr>
<td>111</td>
<td>00</td>
<td>yes</td>
</tr>
</tbody>
</table>

\(d = 3\)—single error correcting or double error detecting
Another example:

\[ H = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \]

<table>
<thead>
<tr>
<th>3-tuple</th>
<th>( c \cdot H )</th>
<th>Code word?</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>yes</td>
</tr>
</tbody>
</table>

\( d = 1 \)—not even single error detecting!
Group Homomorphism

Let \([N_2^n, +_2]\) be a group.

Let \([N_2^n, +_2] \rightarrow [N_2^r, +_2]\) be a homomorphism. (This homomorphism \(f\) maps wider bitstrings to narrower bitstrings.)

\(\ker f\) is the set of elements in \([N_2^n, +_2]\) that map to to \(0_r\) under \(f\).

\(\ker f\) includes \(0_n\), all of its elements are invertible, it is closed, and associatively obviously still holds; therefore, \(\ker f\) is the set of code words in some group code.
If in $H$ the last $r$ rows form the $r \times r$ identity matrix, then $H$ is a canonical parity check matrix.

\[
\begin{bmatrix}
  c_1 & c_2 & c_3 \\
 1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  1 \\
  0 \\
  1
\end{bmatrix} = \begin{bmatrix}
  1 \cdot c_1 + 0 \cdot c_2 + 1 \cdot c_3 \\
  c_1 + c_3
\end{bmatrix} = [0]
\]

This means the number of 1s in the first and third places is even; thus, an even parity check is performed on bits 1 and 3.
Another Example

\[
\begin{bmatrix}
    c_1 & c_2 & c_3 \\
    1 & 1 & 0 \\
    0 & 1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
    1 & 1 \\
    1 & 0 \\
    0 & 1
\end{bmatrix}
= \begin{bmatrix}
    (c_1 + c_2) & (c_1 + c_3)
\end{bmatrix}
= \begin{bmatrix}
    0 & 0
\end{bmatrix}
\]

This means an even parity check is being performed on bits 1 and 2, and an even parity check is being performed on bits 1 and 3.
Minimum Code Weight

The minimum weight of the code = the minimum number of rows in $H$ that add to $0_r$. 
Hamming Codes

To generate a single error correcting code for $N^m_2 = N^{n-r}_2$ (a subgroup of $N^n_2$):

- The dimension of $H$ is $n \times r$
- no two rows of $H$ can be the same (add to $0_r$)
- each row in $H$ has $r$ elements
- there can be no more than $2^r$ rows
- no row can contain $0_r$, so number of rows $\leq 2^r - 1$
- $n \leq 2^r - 1$
- $m = n - r \leq 2^r - r - 1$

A Hamming code is *perfect* if $m = 2^r - r - 1$. 
Hamming Code Example

\[ m = 2 \]

\[ n = 5 \]

\[ r = 3 \]

\[ H = \begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 
\end{bmatrix} \]
Hamming Code Example (cont.)

\[
\begin{bmatrix}
  c_1 & c_2 & c_3 & c_4 & c_5 \\
  1 & 0 & 1 \\
  1 & 1 & 1 \\
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  101 \\
  111 \\
  100 \\
  010 \\
  001
\end{bmatrix} =
\begin{bmatrix}
  (c_1 + c_2 + c_3) \\
  (c_2 + c_4) \\
  (c_1 + c_2 + c_5)
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$C = \{00000, 01111, 10101, 11010\}$
Decoding

Let $c$ be the received code word.

- If $c \cdot H = 0_r$, then strip off the $r$ check bits and interpret the $m$ message bits as the original message.

- If $c \cdot H \neq 0_r$, then at least one of the bits is non-zero. Find the row in $H$ that matches the received bogus code word. The number of the matching row indicates the bit position of the error in the received code word.
Decoding Example

Received Code Word = 01111

Decoding:

\[
\begin{bmatrix}
0 & 1 & 1 & 1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
(0+1+1+0+0) & (0+1+0+1+0) & (0+1+0+0+1)
\end{bmatrix}
= \begin{bmatrix}
0 & 0 & 0
\end{bmatrix}
\]

Interpretation: Message was 01
Decoding Example 2

Received Code Word = 01101

Decoding:

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
(0+1+1+0+0) & (0+1+0+0+0) & (0+1+0+0+1)
\end{bmatrix} = \begin{bmatrix}
0 & 1 & 0
\end{bmatrix}
\]

Interpretation: Non-zero result: 010 which matches row 4 in \( H \); therefore, error is in bit 4, the code word should have been 01111, and message was 01