**Outline**

- Information Theory
- Codes
- Linear Codes
- Hamming Codes
- Cyclic Codes *

---

**Claude Elwood Shannon (1916–2001)**

Claude Elwood Shannon was a mathematician and electrical engineer who made significant contributions to the fields of information theory, communication theory, and cryptography. He is widely regarded as the father of information theory.

**Shannon’s Information Theory**

Two crucial papers:

- *A Mathematical Theory of Communication*
- *Communication Theory of Secrecy Systems*

First paper introduces the notion of a bit. Since Shannon was working for the phone company he was interested in how many phone calls could be transmitted in parallel on the same copper wire. His work relies heavily on probability theory.

---

**Measuring Information**

**Hard Question:** How can we assign a numerical measure to information content?

This may seem like a hopeless task, but it becomes manageable if one stays away from semantics: figuring out whether a text by Immanuel Kant has more information than a text by Albert Einstein is indeed hopeless.

**Key Idea:** Information has to do with surprise: the more surprising an event is the more information it carries.

Technically, this approach allows us to study probabilities rather than embark on the hopeless enterprise of defining a semantics for natural languages.

---

**Axioms for Information**

Consider some event $E$ that occurs with probability $\Pr[E]$. Here are some simple axioms for the information content $I(E)$.

- $\Pr[E] = 1$ implies $I(E) = 0$
- $\Pr[E] = 0$ implies $I(E) = \infty$
- $\Pr[E_0] \leq \Pr[E_1]$ implies $I(E_0) \geq I(E_1)$
- $E_0, E_1$ indep. implies $I(E_0 \land E_1) = I(E_0) + I(E_1)$

From these basic assumptions it follows easily that

$$I(E) = -\log_b \Pr[E]$$

Base $b = 2$ is the standard choice, corresponding to Yes/No answers.
In most applications, Alice sends a stream of symbols to Bob that constitute a message. The symbols are chosen from a fixed collection of symbols, an alphabet \(\{a_1, a_2, \ldots, a_q\}\). Let’s call this scenario a channel. Many symbols are transmitted, so it makes sense to speak of \(a_i\) being sent with probability \(p_i\), where \(\sum p_i = 1\).

**Key Question:** How much information is contained in a message?

Well, it’s the average information content of the symbols that make up the message.

This is called the entropy of the channel:

\[
H = -\sum_i p_i \log p_i
\]

### Uniform Probabilities

Here is a simple special case: uniform probabilities \(p_i = 1/q\).

\[
H = -\sum \frac{1}{q} \log \frac{1}{q} = \log q
\]

In particular for \(q = 2\) we have \(H = 1\). Makes some intuitive sense.

How about \(q = 2\) with biased probabilities?

\[
H = -p \log p - (1 - p) \log (1 - p)
\]

This is the binary entropy function (and very useful in some combinatorial arguments).

### Redundancy

Redundancy is very closely related to entropy and measures the amount of superfluous information being transmitted.

High redundancy can be a blessing (conversation in a noisy room) or a curse (expensive transmission via semaphores).

The redundancy for a \(q\)-symbol channel is given by

\[
R = 1 - \frac{H}{\log q}
\]

Here \(\log q\) represents the maximum entropy of any \(q\)-symbol channel. So in the uniform case we have redundancy 0.

### Example: DNA

Here \(q = 4\) for the bases adenine, thymine, cytosine, and guanine.

Suppose counting in a DNA sample yields probabilities as follows (these numbers are made up, not measured):

\[
p_A = 0.5 \quad p_T = 0.25 \quad p_C = 0.125 \quad p_G = 0.125
\]

This produces

\[
H = 1.75 \quad R = 0.125
\]

Redundancy is rather low, so copying must be nearly error-free to avoid disaster.
We can think of English text as being composed of 26 letters (or 27 if we include space, or more if we allow punctuation, capitalization, ...). If all letters were equally likely and we use a 26 letter alphabet we get $H = 4.7$.

Alas, this is a lousy approximation since:
- English text consists of sequences of blocks of letters (aka words),
- the letters within a block are not equidistributed,
- the arrangement of blocks is controlled by grammar and far from random.

So there is a big problem in calculating entropy of English, or any other natural language.

One approach is to determine probabilities of the $N+1$ letter assuming knowledge of the current $N$ letters (think of assigning probabilities to the edges of a de Bruijn graph). This is easy for small $N$ and the hope is that the numbers converge rapidly to a limit of sorts.

Shannon did concrete calculations and came up with the following estimates:

<table>
<thead>
<tr>
<th>$N$</th>
<th>$H_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.7</td>
</tr>
<tr>
<td>1</td>
<td>4.14</td>
</tr>
<tr>
<td>2</td>
<td>3.56</td>
</tr>
<tr>
<td>3</td>
<td>3.3</td>
</tr>
</tbody>
</table>

If all 3-letter blocks were equally likely we would get $H = 14.1$. Redundancy is high at 0.77.

The last value comes from Shannon’s analysis of words in a dictionary.

Rolf Landauer in 1996:

Information is not a disembodied abstract entity; it is always tied to a physical representation. It is represented by an engraving on a stone tablet, a spin, a charge, a hole in a punched card, a mark on paper, or some other equivalent. This ties the handling of information to all the possibilities and restrictions of our real physical world, its laws of physics and its storehouse of available parts.

Landauer realized that only erasing a bit carries a thermodynamical cost (dissipates heat), all other operations are free – in principle, not using current technology where a computer consists mostly of a cooling unit (as opposed to reversible computation).

How much information is needed to specify an apple completely?

In classical physics this question makes little sense (how do you describe the position of an electron?).

But according to quantum physics there is a certain granularity in all physical systems – the real world is actually discrete.

For each particle making up the apple we need a few bits (order of a few hundred at most) to describe its state. So the total number of bits is approximately the same as the number of particles making up the apple, something like

$$10^{23} = 100000000000000000000000000$$

Not bad at all. For a computer scientist this is a tiny number, think $A(10, 10)$. OK, so it’s more information than stored in all books and computers on planet Earth, but not hugely much more.
Shannon’s Model

A message \( a \) is generated by some source, converted into a signal by the sender (transmitter) which is sent over the channel. The receiver turns the signal into a received message \( a' \) which reaches the destination.

Two major scenarios:

- The channel is noiseless: \( s = s' \).
  Interesting question: how to protect against eaves-dropping, cryptography.

- The channel is noisy: \( s \approx s' \).
  Interesting question: how to recover the original message, coding theory.

Noisy Channels

The coder turns a message \( a \) is turned into a code \( c \), which is sent over a noisy channel.

At the other end, the decoder receives \( c' = c + e \) where \( e \) is the error introduced by the channel (for the moment, don’t worry what exactly addition means here).

It outputs a decoded message \( a' \) which is hopefully identical to \( a \). If only . . .

Symmetric Channels

Suppose we have an \( m \)-ary channel.

We assume that there is a single parameter \( p \) that describes error probabilities:

- \( p \) is the probability that the channel will transmit any symbol \( a \) as a different symbol \( b \).

Thus the probability depends neither on \( a \) nor on \( b \), it is uniform for all \( a \neq b \).

- The probability that \( a \) is wrongly transmitted is thus \( (m - 1)p \).

- The probability that \( a \) is correctly transmitted is \( 1 - (m - 1)p \).

We will always assume that \( p \) is small relative to \( m \) (see below). If not, the problem is really one of channel design, not of coding theory.

We will not deal with insertions and deletions of symbols.

Binary Symmetric Channel

In particular in a binary channel we have

- \( P(1|0) = p \)
- \( P(0|1) = p \)
- \( P(0|0) = 1 - p \)
- \( P(1|1) = 1 - p \)

Block Codes

We will think of messages and codes as words over an alphabet \( Q \):

- \( a = a_1a_2\ldots a_k \)
- \( c = c_1c_2\ldots c_n \)

We are mostly interested in the binary case \( Q = 2 \), but larger alphabets can be handled similarly.

Useful Trick: We can always think of \( Q \) as being some finite field.

So a message and its code are just vectors of field elements. In particular in the binary case we are just dealing with bit-vectors.

Errors

One advantage of the vector model is that we can model the error as just another vector:

- \( c \mapsto c + e \)

The key challenges are

- **Error Detection**
  Detect when \( e \neq 0 \) (and ask for a re-transmit).

- **Error Correction**
  Detect an error and correct it (by adding \(-e\) to the transmitted message).

Again, error detection and correction only work if the channel is reasonably well behaved. In practice, the major bottleneck is usually the efficiency of the decoder.
We have $q^k$ possible messages and need to assign a unique codeword to each.

**Definition**
A \((q\text{-ary})\) code is a subset $C \subseteq \mathbb{F}^n$ of size $q^k$. Here $k$ is the dimension of the code and $n$ is its length. $\mathbb{F}^n$ is the codespace. The elements of $C$ are codewords.

Of course, equal cardinality of message space and code is not enough: we need to worry about coding algorithms that translate a message $x \in \mathbb{F}^k$ into the corresponding codeword $f(x) \in \mathbb{F}^n$ and decoding algorithms that go in the opposite direction, and deal with errors.

As we will see, there is usually quite a bit of flexibility in assigning codewords to messages; there is no magic connection between the two.

Before we talk about specific codes let’s look at some fundamental properties of this setup.

**Hamming Distance**
There is a natural way to introduce geometry in the codespace $K = \mathbb{F}_q^n$: we can measure distances between points.

**Definition**
The Hamming distance between two vectors $x, y \in K$ is defined by
$$\text{dist}(x, y) = |\{i \mid x_i \neq y_i\}|$$

The weight (or support) of a vector $x \in K$ is defined by
$$w(x) = |\{i \mid x_i \neq 0\}|$$

Thus $\text{dist}(x, y) = w(x - y)$.

**Exercise**
Check that $d$ really is a metric.

A basic error in the channel is thus represented by an error vector of weight 1:
$$e = (0, \ldots, 0, x, 0, \ldots)$$

We can think of the channel as introducing a number of basic errors sequentially.

An error vector of weight $e$ is thus referred to as “$e$ errors.” This makes perfect sense in symmetric channels where each flip is independent and equally likely.

**Goal**: We would like simple codes that can correct reasonably many errors, the more the merrier.

**Strategy**: The main strategy is also clear: we will used redundancy to protect against errors. Alas, the details are complicated.

In coding theory texts, when one says a code detects $e$ errors, what is actually meant is this:

For any codeword $x$ and any error $e$ of weight at most $e$:
- one can tell that $y = x + e \notin C$, and
- one can determine the weight of $e$ from $y$.

Being able to determine whether some symbol sequence is a codeword or not is still useful, but this terminology actually makes more sense.

Suppose the received message $c + e$ winds up “between” two codewords at distance 6.

This transmission could be caused by an error of weight 2 or an error of weight 4. We can only detect errors of weight at most 3 in this case; we can correct errors of weight at most 2.
Conditional Probabilities

Suppose we have an $m$-ary symmetric channel with (single) error probability $(m-1)p$.
The probability that $y$ is received when $x$ has been transmitted is
\[ \Pr[y|x] = p^d(1-(m-1)p)^{n-d} \]
where $d = \text{dist}(x,y)$.

As long as $p < 1/m$ this function is sharply decreasing in $d$, so it is unlikely that we will hit a $y$ far away from $x$.
As already mentioned, if $p$ is too large one has to redesign the channel.

Example: $m=3$, $n=5$, $p=0.1$ and $p=0.01$.

This behavior is intuitively clear, but it never hurts to check the math.

Maximum Likelihood Decoding

If $x$ is transmitted and $y \notin C$ is received the natural decoding strategy based on $\Pr[y|x]$ is to find the codeword $c \in C$ closest to $y$.
This makes sense in particular when this $c$ is uniquely determined.

Balls and Errors

Suppose we have chosen a code $C$. We define the minimal distance of $C$ to be
\[ \text{md}(C) = \min \left( \text{dist}(x,y) \mid x \neq y \in C \right) \]
Thus a ball of radius $\text{md}(C) - 1$ around a codeword in $C$ contains no other points in $C$.

Proposition

- If $\text{md}(C) \geq 2e$ then $C$ detects $e$ errors.
- If $\text{md}(C) \geq 2e + 1$ then $C$ corrects $e$ errors.

Note the active voice; we really should have said: it is possible to detect/correct—we don’t yet know how to do this efficiently.

Hamming Bound

Lemma (Hamming)

Suppose $q$-ary code $C$ has size $M$, length $n$ and corrects $e$ errors. Then
\[ M \sum_{i=0}^{e} \binom{n}{i} (q-1)^i \leq q^e \]

Proof.
To see this, consider (necessarily disjoint) balls of radius $e$ centered at $x \in C$.
The number of elements in a ball at distance $i$ to the center is $\binom{n}{i}(q-1)^i$.

Note that the code size $M$ is forced to be smaller when $e$ becomes larger.

Example 1: Repetition Code

This is a bit embarrassing, but bear with me. Let $Q=2$ and code
\[ x \mapsto (x,x,\ldots,x) \]
So the only codewords are $00\ldots0$ and $11\ldots1$.
A repetition code detects $n-1$ errors.

More importantly, it corrects $\lfloor (n-1)/2 \rfloor$ errors: we use a majority count to determine the original codeword.

The obvious fatal problem with this is the low rate of transmission: the codeword is $n$-times longer than the original message.
Example 2: Parity Check Code

For any \( Q \), code

\[
x \mapsto (x, \sum x_i)
\]

so that \( n = k + 1 \).

A parity check code detects 1 error.
Alas, it corrects none.

This may sound useless, but it is not: variants of this scheme are used in ISBN.

Example 3: Correcting One Error

Let \( q = 2 \) and \( k = 4 \).

We transmit 7 bits \( c = (c_1, c_2, c_3, c_4, c_5, c_6, c_7) \)

Here \( c_3 = x_1, c_5 = x_2, c_6 = x_3, c_7 = x_4 \) are the original message bits. Then

\[
\begin{align*}
\gamma &= c_1 + c_3 + c_5 + c_7 = 0 \\
\beta &= c_2 + c_3 + c_6 + c_7 = 0 \\
\alpha &= c_4 + c_5 + c_6 + c_7 = 0
\end{align*}
\]

For decoding, calculate \( \alpha, \beta \) and \( \gamma \). If there is a single error then \( \alpha \beta \gamma \) is the binary expansion of the place where the error occurred. Nice hack.

Example 4: . . . and Detecting Two

The last code does not handle 2 errors in a civilized way: the following codewords have identical 1-error and 2-error versions:

\[
\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1
\end{array} =
\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0
\end{array}
\]

To fix this problem we can add one more bit

\[c_0 = c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7\]

This extended code detects two errors (and still corrects one as before).

Using Algebra

So far, everything is a bit ad-hoc. Here comes the first real idea:

Definition

A linear code is a linear subspace \( C \subseteq K = F^n \).
If \( C \) has dimension \( k \) we refer to it as an \( [n, k] \) code.
A generator matrix for \( C \) is a \( k \times n \) matrix over \( F_q \) whose rows form a basis for \( C \).

It follows that \( C \) is the row space of \( G \):

\[ C = \{ x \in G \} \]

Example: \([6, 3]\) Code

Let \( Q = 2, k = 3, n = 6 \) and

\[
G = \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\]

Then

\[
x \mapsto y = (x_1, x_2, x_3, x_2 + x_5, x_1 + x_3, x_1 + x_2)
\]

Decoding in the absence of errors is entirely trivial, but what if the decoder receives \( y + e \) instead?
Note that our generator matrix has the form
\[ G = (I_k P) \]
where \( I_k \) is the identity matrix of order \( k \) (so \( G \) is in reduced echelon form). Such codes are said to be in standard form. A slightly weaker condition is that the columns of \( I_k \) can be spread out through \( G \); this is called systematic form. This is no real constraint, one can always permute symbols in the codewords to get standard form.

Correspondingly one calls
- the first \( k \) bits information symbols
- the last \( n-k \) bits parity check symbols
In this sense all linear codes are parity check codes.

Let \( G = (I_k P) \) be the \( k \times n \) generator matrix of a standard form \([n,k]\) linear code. Define the \((n-k) \times n\) matrix \( H \) by
\[ H = (P^T I_{n-k}) \]
Definition
\( H \) is the parity check matrix for code \( C \).
The vector \( \text{syn}(x) = x \cdot H^T \) is the syndrome of \( x \).
Since we are working in characteristic 2 we have \( G \cdot H^T = 0 \) so that
\[ x \in C \iff (x) = 0 \]
and we can test membership in \( C \) via a simple matrix multiplication.

For the \([6,3]\) code from above we have
\[ G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \]
and
\[ H = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \]
In this case, the \( P \)-part of \( G \) is symmetric and thus invariant under transpose.

Decoding here hinges on the following observation (which is not hard to prove):

Claim
\( (s_1, s_2, s_3) \neq (1,1,1) \) there is a unique choice for such an error vector \( e \) (of weight at most 1).

Alas, if \( (s_1, s_2, s_3) = (1,1,1) \) all the following 3 weight 2 vectors work:
\( (1,0,0,0,1,0), (0,1,0,0,1,0), (0,0,1,0,0,1) \).
In this case we can either ask for a retransmit, or we can simply pick one of the 3 possibilities–which will be right \( 1/3 \) of the time.
Determining Minimum Distance

According to our definition the minimum distance is the least \( \text{dist}(x, y) \) where \( x \neq y \in C \). In a linear code we can do better.

Lemma

For a linear code we have

\[
\text{md}(C) = \min \left\{ w(x) \mid 0 \neq x \in C \right\}
\]

This may not look too impressive, but at least it yields a linear time method as opposed to the obvious quadratic one that works for any code.

Orthogonality

Definition

For \( x, y \in F^n \) define the inner product

\[
x \cdot y = \sum x_i y_i
\]

Given a subset \( X \subseteq F^n \) define its orthogonal complement by

\[
X^\perp = \{ y \in F^n \mid \forall x \in X \ (x \cdot y = 0) \}
\]

This is similar to the notion of orthogonal complements in real spaces but the geometry here is more complicated. E.g., it may happen that \( X^\perp = X \). For example, consider \( X = \{ (x_1, x_2, x_2, \ldots, x_k, x_k) \mid x \in F^2 \} \).

Dual Codes

Recall that a code with a \( k \times n \) generator matrix has a \( (n - k) \times n \) parity check matrix.

Here is a good way to think about check matrices: there are actually generators for a closely related code.

Definition

Let \( C \) be a linear \( [n, k] \) code. The dual code is \( C^\perp \).

- The dual code has dimension \( n - k \).
- The dual of the dual is the code: \( C^{\perp \perp} = C \).
- If \( C \) has generator matrix \( G \) then \( C^\perp \) has parity check matrix \( G \).
- If \( C \) has parity check matrix \( H \) then \( C^\perp \) has generator matrix \( H \).

Decoding

How do we decode a linear code systematically?

Suppose \( c \in C \) is sent but \( x = c + e \) is received.

Obviously it suffices to compute the error vector \( e \). Note that

\[
x + C = e + C
\]

so the error vector lies in the same coset as the received message. Since \( C \) is a subgroup of the additive group of \( F^n \), we can decompose \( F^n \) into \( n - k \) disjoint cosets

\[
a_1 + C, a_2 + C, \ldots, a_{n-k} + C
\]

Definition

A coset leader is an element of the coset of minimal weight.
So if we want to do maximum likelihood decoding we should pick a minimum weight vector in the coset, the coset leader, as the error vector.

Note that $x$ and $y$ are in the same coset iff they have the same syndrome:

$$\text{syn}(x) = \text{syn}(y) \iff x - y \in C$$

So we have to precompute a minimum weight vector for each possible syndrome.

For our standard $[6,3]$ example we have the following coset leaders, parametrized by syndrome (which is none other than the $s$-vectors from above):

<table>
<thead>
<tr>
<th>Syndrome</th>
<th>Leader</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000000</td>
</tr>
<tr>
<td>001</td>
<td>000001</td>
</tr>
<tr>
<td>010</td>
<td>000010</td>
</tr>
<tr>
<td>100</td>
<td>000100</td>
</tr>
<tr>
<td>011</td>
<td>001000</td>
</tr>
<tr>
<td>101</td>
<td>010000</td>
</tr>
<tr>
<td>110</td>
<td>001000</td>
</tr>
<tr>
<td>111</td>
<td>001001</td>
</tr>
<tr>
<td>010010</td>
<td></td>
</tr>
<tr>
<td>100100</td>
<td></td>
</tr>
</tbody>
</table>

As we have seen, the syndrome function is constant on each coset and differs on each coset (if you like, the coset partition is just the kernel relation induced by the syndrome function).

But then it suffices to store a syndrome table: for each syndrome store the corresponding coset leader.

To decode $x$:

- Compute $s = \text{syn}(x)$.
- Then look up the corresponding $e$.
- Return $x' = x - e$.

The purpose of computation is insight, not numbers.
Errors and Check Columns

Suppose $H$ is the check matrix of a binary $[n,k]$ code.
Note that the syndrome of a received word $x = c + e$ is the sum of the columns of $H$ in positions where an error occurred:

$$\text{syn}(x) = \text{syn}(c) + \text{syn}(e) = He^T$$

In the special case where there is only one error we just get a single column in $H$. If the columns are all distinct this tells us the position of the error and we can correct it.

General Case

A little care is needed to lift the binary example to the $q$-ary case.

Definition
A linear code over $\mathbb{F}_q$ is a Hamming code of redundancy $r$ if it has a check matrix $H$ that contains a unique column $ax$ for all $0 \neq x \in \mathbb{F}_q^r$ and some $0 \neq a \in \mathbb{F}_q$.

Thus every non-zero vector in $\mathbb{F}_q^r$ appears as exactly one column in $H$, up to multiplication by a non-zero scalar.

One can determine the multiplicative factor for example by insisting that the first non-zero entry in each column be 1.

Hamming Length and Dimension

Theorem
Let $C$ be a Hamming code of redundancy $r \geq 2$. Then $C$ is a $[q^{r-1},q^{r-1}-r]$ code. The minimum distance is 3.

Proof. There are $q^r - 1$ non-zero vectors of length $r$ over $\mathbb{F}_q$. Since each column in $H$ is a non-zero multiple there are $n = (q^r - 1)/(q - 1)$ columns, the length of the code. The dimension then is $n - r$.

The claim about distance follows from the lemma about linear dependence of columns in $H$ above.

An Improvement

We can add a parity bit to the Hamming $[7,4]$ to get a $[8,4]$ code. This code

- has minimum distance 4,
- corrects 1 error,
- detects 2 errors (in the strong sense above, 3 in the weak sense).

This type of code is sometimes called a SECDEC code (single error correction, double error detection) and used in memory systems.

Perfect Codes

So far we have focused on codewords being sufficiently far from each other. Another reasonable condition is that no point in codespace should be far from a codeword.

Definition
A code with minimum distance $2e + 1$ is perfect if for all $z \in \mathbb{F}_q^n$ the ball of radius $e$ contains exactly one codeword.

In other words, a code is perfect if we have equality in Hamming’s bound.

Here is a bad example for a perfect code: the repetition code of length $n = 2e + 1$ has $\rho(C) = e$ and $\text{md}(C) = 2e + 1$. 


Let $Q = 2$, $k = 4$, $n = 7$ and define a code $C$ by

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}, \quad G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$H$ is organized so as to make $G$ simple (standard form, the first 4 bits are information bits). Note, though, that every 3-bit word other than 0 appears as a column in $H$.

It is not hard to see that $\text{md}(C) = 3$, so we can correct one error by computing the syndrome.
Recall that in the $q$-ary case we have length $n = (q^k - 1)/(q - 1)$ and dimension $n - r$.

Claim
Every Hamming code is perfect.

We have minimum distance $3$, so $c = 1$. We need to check that for any sphere $S$ of radius $1$ we have $|C||S| = q^n$. But the latter is equivalent to

$q^n - (1 + (q - 1)n) = q^n$

which is easily verified.

Definition
A linear code $C$ is cyclic if for all $c \in C$ the cyclic shift of $c$ is also in $C$.

As we will see, cyclic codes have a very simple description (a single generator). They are used mostly for error detection: cyclic redundancy check (CRC). A retransmit is requested if an error is found.

Example
$C = \{000, 011, 101, 110\}$ is cyclic code of length $3$.

This generalizes to all words in $F^k$ with even weight.

Instead of thinking about vectors $a \in F^n$ we will interpret codewords as polynomials:

$a(x) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1}$

This is also called a code polynomial.

Write $a'$ for the cyclic right-shift of $a$ and note that the corresponding code polynomial has the form

$a_{n-1} + a_0x + a_1x^2 + \ldots + a_{n-2}x^n$

But then $a'(x) = x \cdot a(x) \bmod (x^n - 1)$.

So if we think of code polynomials as elements of the quotient ring $F[x]/(x^n - 1)$, then shift corresponds simply to multiplication by $x$. Very nice.

Is there a good description for cyclic codes, viewed as subsets of $F_q[x]/(x^n - 1)$?

Lemma
A linear code $C$ is cyclic iff $C$ is an ideal in $F_q[x]/(x^n - 1)$.

Proof. If $C$ is cyclic then the set of corresponding code polynomials is closed under multiplication by $x$, and hence under multiplication by any polynomial. Closure under addition follows from $C$ being a code.

In the opposite direction, $C$ must be additively closed since the ideal is so closed. And $C$ must be invariant under shifts since the ideal is closed under multiplication by $x$.

It is known from algebra that $F_q[x]/(x^n - 1)$ is a principal ideal domain: all ideals in this ring are principal ideals. Thus, $C = \langle g(x) \rangle = F_q[x]/g(x)$ for some polynomial $g(x)$.

More precisely, the generator polynomial $g(x)$ is a monic polynomial in $C - \{0\}$ of minimal degree (this polynomial is uniquely determined by $C$).

Note that $C$ has dimension $n - \deg(g)$.

Once we have a generator polynomial $g(x)$, coding is very easy:

$a \mapsto \left( \sum a_i x^i \right) g(x)$

This can be implemented with the help of dedicated hardware (feedback shift register), so coding can be very fast.
The generator polynomial $g(x)$ must divide $x^n - 1$: otherwise $x^n - 1 = g(x)h(x) + r(x)$ where $\deg(r) < \deg(g)$. But $r(x) \in C$, contradiction.

So let $g(x)h(x) = x^n - 1$, so $g$ and $h$ are zero-divisors in $\mathbb{F}_q[x]/(x^n - 1)$.

But then $c \in C \iff c(h(x)) = 0$ in $\mathbb{F}_q[x]/(x^n - 1)$.

More generally, we can compute the syndrome of a received message $y$ by multiplying $y(x)$ and $h(x)$, and reducing modulo $x^n \rightarrow 1$. Note that this reduction is easier than the ones we saw in connection with finite fields.

Let $q = 2$ and $n = 9$. Then

$$x^9 + 1 = (x + 1)(x^2 + x + 1)(x^6 + x^3 + 1)$$

The maximum code based on $g(x) = x^6 + x^3 + 1$ has codewords

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

and minimum distance 3.

The minimum code based on $h(x) = (x + 1)(x^2 + x + 1)$ is a $[9, 6]$ code with minimum distance 2; it has codewords

$$(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$$

Note that these codewords are in fact cyclic (which is not entirely obvious from the definition).

From now on assume that the length of the code $n$ and $p$, the characteristic of the underlying field, are coprime. In this case $x^n - 1$ factors into distinct irreducible polynomials:

$$x^n - 1 = f_1(x) f_2(x) \cdots f_{t}(x) f_t(x)$$

We can produce $2^t$ generators by choosing $I \subseteq [t]$: 

$$g(x) = \prod_{i \in I} f_i(x)$$

A maximal cyclic code has $I = \{i\}$: we pick exactly one of these irreducible factors.

Dually, a minimal cyclic code has $I = [n] - \{i\}$, we choose all but one.

Let $g = 2$ and $n = 9$. Then

$$x^9 + 1 = (x + 1)(x^2 + x + 1)(x^6 + x^3 + 1)$$

The maximum code based on $g(x) = x^6 + x^3 + 1$ has codewords

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

and minimum distance 3.

The minimum code based on $h(x) = (x + 1)(x^2 + x + 1)$ is a $[9, 6]$ code with minimum distance 2; it has codewords

$$(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$$

Note that these codewords are in fact cyclic (which is not entirely obvious from the definition).

We can compute a cyclic code by polynomial multiplication $c(x) = a(x)g(x) \mod (x^n - 1)$.

Slightly better performance can be obtained by coding $a \rightarrow (a, -a)$

where $a$ is determined by

$$x^{n-k}a(x) = f(x)g(x) + s(x)$$

The reason this is slightly better is that one can send the first $k$ information symbols while computing the following $n - k$ parity symbols.