Lecture 4:
Linear Codes
Linear codes over $\mathbb{F}_q$

- We let $\mathcal{X} = \mathbb{F}_q$ for some prime power $q$. Most important case: $q = 2$ (binary codes).

- Without loss of generality, we may represent the information message as a sequence of $k$ symbols from $\mathbb{F}_q$.

- We have $|C| = q^k$, and $R = \frac{k}{n} \log_2 q$ bits/symbol.

**Definition 22.** A $(q^k, n)$ block code over $\mathcal{X} = \mathbb{F}_q$ is called a linear $(n, k)$ code if its codewords form a $k$-dimensional vector subspace of the vector space $\mathbb{F}_q^n$. ◇
Elementary properties of linear codes

- The code $C$ is an additive group, in particular, if $c, c' \in C$ then $c + c' \in C$ and $-c \in C$.

- The all-zero vector is a codeword: $0 \in C$.

- Linear combination of codewords are codewords: $c_1, \ldots, c_\ell \in C$ and $a_1, \ldots, a_\ell \in \mathbb{F}_q$, then
  $$a_1 c_1 + \cdots + a_\ell c_\ell \in C$$

- There exist (non-unique) sets of $k$ linearly independent codewords that generate the whole code, i.e.,
  $$C = \left\{ \sum_{\ell=0}^{k-1} u_\ell g_\ell : u_0, \ldots, u_{k-1} \in \mathbb{F}_q \right\}$$

  where $g_0, \ldots, g_{k-1}$ are codewords that form a basis for the code $C$. 
• We can arrange the basis as rows of a $k \times n$ matrix

$$G = \begin{bmatrix}
g_{0,0} & g_{0,1} & g_{0,2} & \cdots & g_{0,n-1} 
g_{1,0} & g_{1,1} & g_{1,2} & \cdots & g_{1,n-1} 
\vdots & & & & \vdots 
g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1}
\end{bmatrix}$$

This is called a generator matrix for the code. Letting $u = (u_0, \ldots, u_{k-1})$ we can write the encoding function as

$$c = uG$$
(7, 4) *binary Hamming code*

\[ G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \]
Repetition and single-parity-check codes

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

\[
\begin{bmatrix}
1 & 1 & 1
\end{bmatrix}
\]
Systematic encoding

- Sometimes, a desirable property of the encoding function is that the vector of information symbols appears explicitly as part of the codeword.
- For linear codes, a generator matrix in systematic form is given by

\[
G = [P | I_k]
\]

where \( P \in \mathbb{F}^{k \times (n-k)}_q \) and \( I_k \) denotes the \( k \times k \) identity.
- In this way, we have \( c = uG = [uP | u] \).
Parity-check matrix

• Since $C$ is a vector subspace of $\mathbb{F}_q^n$ of dimension $k$, then it can be seen as the Kernel of some linear transformation $\mathbb{F}_q^n \rightarrow \mathbb{F}_q^{n-k}$.

• The matrix of such linear transformation is called **parity-check matrix** and it is denoted by $H^T$, where $H \in \mathbb{F}_q^{(n-k) \times n}$:

\[
C = \{ c \in \mathbb{F}_q^n : cH^T = 0 \}
\]

• In particular, the rows of $H$ are $n$-vectors in $\mathbb{F}_q^n$ that are orthogonal to all codewords. Indeed, they generate the orthogonal subspace $C^\perp$, also known as the **dual code** of $C$.

• If $G = [P | I_k]$, then $H = [I_{n-k} | -P^T]$, in fact, we have

\[
GH^T = [P | I_k] \begin{bmatrix} I_{n-k} \\ -P \end{bmatrix} = P - P = 0
\]
$q$-ary symmetric channels

- A $q$-ary symmetric channel can always be represented as an additive noise channel over $\mathbb{F}_q^n$, such that
  \[ y = c + z \]
  where $z \in \mathbb{F}_q^n$.

- The “noise” pmf is given by
  \[
P_Z(z) = \begin{cases} 
1 - \delta & \text{for } z = 0 \\
\delta/(q - 1) & \text{for } z \neq 0
\end{cases}
\]

- The syndrome of the error vector is given by
  \[ s = zH^\top \]
Syndrome decoding

- Notice that the decoder can compute the syndrome even though it does not know \( z \), in fact,

\[
yH^T = (c + z)H^T = cH^T + zH^T = zH^T = s
\]

- Therefore, the syndrome of the error vector is an index that can be used by the decoder to “undo” the bit-flips.

- If \( s = 0 \), then \( y \in C \). In this case, we let \( \hat{c} = y \).

- If \( s \neq 0 \), then \( y \notin C \). In this case the decoder knows that an error has occurred (detectable error).
• In order to **correct** the error, the decoder needs to find $\hat{z}$, an estimate of $z$, and correct the errors as $\hat{c} = y - \hat{z}$.

• Unfortunately, the system of equations

$$s = zH^\top$$

where $s$ is known (the syndrome) and $z$ is unknown, is **underdetermined** ($n$ unknowns and $n - k$ equations).

• For every syndrome $s \in \mathbb{F}_q^{n-k}$, we have $q^k$ possible error vectors.

• We have to solve this problem in a probabilistic sense ... for each set of $q^k$ possible error vectors corresponding to a given syndrome, we shall pick the most likely.
Standard array

• The linear map $\mathbb{F}_q^n \rightarrow \mathbb{F}_q^{n-k}$ such that $y \mapsto s = yH^T$ has Kernel

$$\text{Ker}(H^T) = C$$

• Linear maps are group homomorphisms (they preserve the group operation, that in this case is componentwise addition in $\mathbb{F}_q$).

• A coset of $C$ in $\mathbb{F}_q^n$ is given by the translate $v + C$ for some $v \in \mathbb{F}_q^n$.

• The factor group (group of cosets), with respect to the coset addition, satisfies (canonical isomorphism):

$$\mathbb{F}_q^n / C \cong \mathbb{F}_q^{n-k}$$

• The standard array is the correspondence between the syndromes $s \in \mathbb{F}_q^{n-k}$ and the cosets of $C$ in $\mathbb{F}_q^n$, induced by this isomorphism.
Building the standard array

<table>
<thead>
<tr>
<th>( s_0 = 0 )</th>
<th>( c_0 = 0 )</th>
<th>( c_1 )</th>
<th>( \cdots )</th>
<th>( c_{q^k-1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>( v_1 )</td>
<td>( v_1 + c_1 )</td>
<td>( \cdots )</td>
<td>( v_1 + c_{q^k-1} )</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>( v_2 )</td>
<td>( c_2 + c_1 )</td>
<td>( \cdots )</td>
<td>( v_2 + c_{q^k-1} )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( s_{q^{n-k}-1} )</td>
<td>( v_{q^{n-k}-1} )</td>
<td>( v_{q^{n-k}-1} + c_1 )</td>
<td>( \cdots )</td>
<td>( v_{q^{n-k}-1} + c_{q^k-1} )</td>
</tr>
</tbody>
</table>

- For every new row, find the vector in \( \mathbb{F}_q^n \) with minimum Hamming weight that has not yet appeared in the array.
- The corresponding row is obtained by adding this vector to all codewords.
- All row are distinct, and yield the same syndrome.
- These vectors of minimum weight are called coset leaders.
Example

• Consider the SPC \((3, 2)\), with parity-check matrix

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

• Exercise: build the standard array for the Hamming \((7, 4)\) code.
Error correction algorithm

1. Compute the syndrome \( s = yH^T \).

2. Use the standard array and find the most likely error vector \( \hat{z} \) compatible with \( s \) (coset leader).

3. The minimum Hamming distance decision rule is given by

\[
\hat{c} = y - \hat{z}
\]

4. A \((n, k)\) linear code is able to correct \(q^{n-k}\) error patterns (error vectors)
The Tanner Graph of a linear code

- Consider a linear \((n, k)\) block code \(C = \{c : cH^T = 0\}\).

- It is defined by the set of parity-check equations

\[
\begin{align*}
h_{1,1}x_1 + h_{1,2}x_2 + \cdots + h_{1,n}x_n &= 0 \\
h_{2,1}x_1 + h_{2,2}x_2 + \cdots + h_{2,n}x_n &= 0 \\
&\vdots \\
h_{n-k,1}x_1 + h_{n-k,2}x_2 + \cdots + h_{n-k,n}x_n &= 0
\end{align*}
\]

- The \textbf{Tanner graph} of the code is a bipartite graph with \(n\) “bit-nodes” and \(n - k\) “check-nodes”, such that an edge \((i, j)\) exists if \(h_{i,j} = 1\), that is, if bit \(x_j\) participate in the parity-check equation \(i\).
• Parity-check equations of the Hamming \((7, 4)\) code:

\[
\begin{align*}
    x_1 + x_2 + x_4 + x_5 &= 0 \\
    x_1 + x_3 + x_4 + x_6 &= 0 \\
    x_2 + x_3 + x_4 + x_7 &= 0
\end{align*}
\]
LDPC codes

- Low-Density Parity-Check (LDPC) codes are linear binary codes with the characteristic that their parity-check matrix is sparse: the number of “ones” in the matrix is proportional to the block length $n$.

- Notice that a randomly generated binary matrix $H$ with dimensions $n(1-R) \times n$ has an average number of ones equal to $n^2(1-R)/2$, i.e., quadratic with $n$.

- A regular $(d_\ell, d_r)$ LDPC code has Tanner graph with constant left and right degrees $d_\ell$ and $d_r$, respectively.

- Example: a $(3, 6)$ regular LDPC code of length $n = 10$, given by the parity-check matrix

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

Copyright G. Caire
(3, 6) LDPC code with \( n = 10 \)

Figure 2.11: Tanner graph for an instance of the Gallager code of length 10.
Introducing memory

- A binary linear block encoder is a linear transformation $\mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$.

- What about using a Linear time-invariant linear system for encoding?

- Convolutional codes consider small $k$ and $n$, but introduce memory into the encoding process of a sequence of consecutive blocks.

- We may see this as the convolution of a sequence of information blocks $\{u_i\}$ with a matrix $G$ of impulse responses, in order to generate a sequence of coded blocks $\{c_i\}$. 
A $k \times n$ Moving Average (MA) system is defined by:

$$c_i^{(1)} = \sum_{\ell=0}^{m_{1,1}} g_{\ell}^{(1,1)} u_{i-\ell}^{(1)} + \cdots + \sum_{\ell=0}^{m_{k,1}} g_{\ell}^{(k,1)} u_{i-\ell}^{(k)}$$

$$c_i^{(2)} = \sum_{\ell=0}^{m_{1,2}} g_{\ell}^{(1,2)} u_{i-\ell}^{(1)} + \cdots + \sum_{\ell=0}^{m_{k,2}} g_{\ell}^{(k,2)} u_{i-\ell}^{(k)}$$

$$\vdots$$

$$c_i^{(n)} = \sum_{\ell=0}^{m_{1,n}} g_{\ell}^{(1,n)} u_{i-\ell}^{(1)} + \cdots + \sum_{\ell=0}^{m_{k,n}} g_{\ell}^{(k,n)} u_{i-\ell}^{(k)}$$
• Defining a vector output sequence \( c_0, c_1, c_2, \ldots \) such that \( c_i = (c_i^{(1)}, \ldots, c_i^{(n)}) \) and a vector input sequence \( u_0, u_1, u_2, \ldots \) such that \( u_i = (u_i^{(1)}, \ldots, u_i^{(k)}) \), we can write

\[
c_i = \sum_{\ell=0}^{m} u_{i-\ell} G_\ell
\]

where we let \( m = \max\{m_{i,j}\} \).

• We obtain a block-Toeplitz notation

\[
(c_0, c_1, c_2, c_3, \ldots) = (u_0, u_1, u_2, u_3, \ldots)
\]

\[
\begin{bmatrix}
G_0 & G_1 & \cdots & G_m & 0 & \cdots \\
0 & G_0 & \cdots & G_{m-1} & G_m & \cdots \\
\vdots & 0 & \ddots & \vdots & G_{m-1} & \cdots \\
& G_0 & \cdots & G_0 & \cdots
\end{bmatrix}
\]

• The impulse responses \( g^{(i,j)} \) are called the code generators.
• $D$-transform domain

\[ u_i \rightarrow u(D) = \sum_i u_i D^i \]  (Laurent series)

• Convolutional encoding in the $D$-transform domain:

\[ c(D) = u(D) G(D) \]

or, equivalently,

\[ (c_1(D), \ldots, c_n(D)) = (u_1(D), \ldots, u_k(D)) \begin{bmatrix} g_{1,1}(D) & g_{1,2}(D) & \cdots & g_{1,n}(D) \\ g_{2,1}(D) & g_{2,2}(D) & \cdots & g_{2,n}(D) \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,1}(D) & g_{k,2}(D) & \cdots & g_{k,n}(D) \end{bmatrix} \]
Example: a \((2, 1)\) convolutional code

\[
C_i^{(1)} = u_i + u_{i-2}
\]

\[
C_i^{(2)} = u_i + u_{i-1} + u_{i-2}
\]
Encoder canonical forms

• A code $C$ is defined as the set of all output sequences (code sequences).

• As for block codes, a convolutional code $C$ may have several input-output encoder implementations.

• We seek encoders in **canonical form**: a general problem in system theory is for a given system, defined as the ensemble of all its output sequences, what is the minimal canonical realization?

• State-space representation (ABCD):

\[ s_{i+1} = s_i A + u_i B, \quad c_i = s_i C + u_i D \]

a minimal representation is a representation with the minimum number of state variables.
• In the \((2, 1)\) example of before, the state is defined as the content of the memory elements,

\[
    s_i = (u_{i-1}, u_{i-2})
\]

therefore we have

\[
    s_{i+1} = s_i \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + u_i \begin{bmatrix} 1 & 0 \end{bmatrix}
\]

and

\[
    c_i = s_i \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} + u_i \begin{bmatrix} 1 & 1 \end{bmatrix}
\]
A convolutional code can be seen as a block code defined on the field $\mathbb{F}_q(D)$ of rational functions over $\mathbb{F}_q$.

Roughly speaking: rational functions are to polynomials as rationals $\mathbb{Q}$ to the integers $\mathbb{Z}$.

Generalizing what seen before, we can consider $G(D)$ with rational elements $g_{i,j}(D)$.

In system theory, this corresponds to AR-MA linear systems.

The code is preserved by elementary row operations.

It follows that for any $G(D)$, we can find a systematic generator matrix in the form

$$G(D) = [I|P(D)]$$

where $I$ is the $k \times k$ identity, and $P(D)$ is a $k \times (n - k)$ matrix of rational functions.
State diagram

• A state-space realization with \( m \) binary state variables is a finite-state machine (FSM) with a state space \( \Sigma = \mathbb{F}_2^m \).

• In general, a FSM is described by its state transition diagram, i.e., by a graph with \( |\Sigma| \) vertices, corresponding to all possible state configurations, and edges connecting those states for which a transition is possible.

• Each edge \((s, s') \in \Sigma \times \Sigma\) is labeled by input and output vectors \( b \in \mathbb{F}_2^k \) and \( c \in \mathbb{F}_2^n \), corresponding to the state transition between \( s \) and \( s' \).
Example: the $(2, 1)$ 4-state code
Trellis diagram

- An alternative representation consists of the trellis section, i.e., by a bipartite graph with $|\Sigma|$ state vertices on the left and $|\Sigma|$ state vertices on the right.

- Left vertices represent the possible states at time $i$, and right vertices represent the possible states at time $i+1$. Edges represent the possible state transitions corresponding to input $u_i$ and output $c_i$.

- The trellis representation follows from the state transition diagram by introducing the time axis.

- A trellis diagram for a convolutional code consists of the concatenation of an infinite number of trellis sections.

- Given an initial state at time $i = 0$, an input sequence $u(D)$ determines an output sequence $c(D)$ and a state sequence $s(D)$ that correspond to a path in the trellis.
Example: the $(2, 1)$ 4-state code

![Diagram of a $(2, 1)$ 4-state code](image)
The Factor Graph for a convolutional code

- Three types of variable nodes: information bitnodes $u_i$, coded bitnodes $c_i$ and states nodes $s_i$.

- The function nodes correspond to the state and output mappings

$$s_{i+1} = s_i A + u_i B, \quad c_i = s_i C + u_i D$$
End of Lecture 4