Convolutional Codes

Telecommunications Laboratory
Alex Balatsoukas-Stimming

Technical University of Crete

November 6th, 2008
Outline

1. Trellis diagrams

2. Convolutional codes
   - A first look
   - Theoretical foundations
   - Defining convolutional codes
   - Systematic encoders
   - Polynomial encoders
   - Minimal encoders
   - Punctured convolutional codes

3. Block codes from convolutional codes
   - Direct termination
   - Zero termination
   - Tail-biting

4. Performance evaluation
Two categories:
1. Binary symbols, linear encoders $\rightarrow$ Convolutional codes
2. General set of symbols and encoders $\rightarrow$ Trellis-coded modulation

The trellis will be assumed to have a periodic structure, meaning that the Viterbi decoding algorithm operations will be the same for every state transition interval.

To construct such a trellis, we can use a memory-$\nu$ binary shift register whose contents at any given time define the state of the trellis.

Obviously, the number of states is $2^\nu$
**Trellis example**

- For $\nu = 2$ we have $2^2 = 4$ states: 00, 01, 10 and 11.
- From state $yz$ we can only move to $xy$, where $x$ denotes the input symbol.

A section of the trellis generated by the above shift register.
Convolutional codes
A first look at convolutional codes

- A convolutional code linearly combines the contents of the shift register to create an output.
- Such a code is said to have memory $\nu$.
- If for every input bit the code creates $n_0$ output bits, the code has a rate of $1/n_0$.
- The branches of the corresponding trellis are labeled with the output symbols generated by the state transitions they represent.
Convolutional code example (1/2)

- Consider the following encoder:

  ![Encoder Diagram]

- For each input bit, we have two output bits, so the rate of the encoder is 1/2.
- The output bits are:

  \[ c_1 = x_1 + x_2 + x_3 \]

  \[ c_2 = x_1 + x_3 \]
Convolutional code example (2/2)

- Conventionally, the initial state is chosen as the all-zero state.

The trellis representing the above code.
Another representation of a convolutional code is its state diagram.

A state diagram describes the transitions between states and the corresponding output symbols without an explicit time axis.

The state diagram representing the above code.
Graph reduction rules

We can gradually reduce a graph to a straight line to find its transfer function, using the following rules:

1. \[ A \rightarrow B \]
2. \[ A \circ B \]
3. \[ B \circ A \]
4. \[ B \circ B \]
Having only rate $1/n_0$ codes is obviously not very practical.

We can define rate $k_0/n_0$ codes. These codes create $n_0$ output bits for each $k_0$ input bits.

To achieve this, we need $k_0$ shift registers and $n_0$ binary adders.
In general, a single input, single output causal time-invariant system is characterized by its impulse response:

$$g \triangleq \{g_i\}_{i=0}^\infty$$

The output sequence \(x \triangleq \{x_i\}_{i=-\infty}^\infty\) is related to the input sequence \(u \triangleq \{u_i\}_{i=-\infty}^\infty\) by the convolution:

$$x = g * u$$
We can associate the sequences \( g, x \) and \( u \) with their D-transforms.

The D-transform is a function of the indeterminate \( D \) (the delay operator) and is defined as:

\[
g(D) = \sum_{i=0}^{\infty} g_i D^i
\]

\[
x(D) = \sum_{i=\infty}^{-\infty} x_i D^i
\]

\[
u(D) = \sum_{i=\infty}^{-\infty} u_i D^i
\]
The convolution $x = g * u$ can be now written as:

$$x(D) = u(D)g(D)$$

If $g(0) = 1$ we say that the polynomial $g$ is delay-free.

$g(D)$ may have an infinite number of terms, if for example it has the form of a ratio between polynomials:

$$g(D) = p(D)/q(D)$$

Every rational transfer function with a delay-free $q(D)$ can be realized in the “controller form” (i.e. with feedback).

Each such function is called realizable.
We can now describe a rate $k_0/n_0$ convolutional code through a $k_0 \times n_0$ generator matrix $G$ which contains its $k_0n_0$ impulse responses.

Recall the following encoder:

![Diagram of encoder](image)

We have 1 input and 2 outputs, so the generator matrix will have dimensions $1 \times 2$ with:

$$g_{11} = 1 + D + D^2 \quad g_{12} = 1 + D^2$$
Defining convolutional codes (1/2)

- We can define a rate $k_0/n_0$ convolutional code as the set of all possible sequences one can observe at the output of a convolutional encoder.

- For a convolutional encoder to be useful, we require it to:
  1. be realizable
  2. be delay free
  3. have a rank $k_0$ generator matrix
The same convolutional code can be generated by more than one encoder.

Let $Q(D)$ denote an invertible matrix, we have:

$$x(D) = u(D)G(D) = u(D)Q(D)Q^{-1}(D)G(D) = u'(D)G'(D)$$

All encoders generating the same code are called equivalent.

We look for useful properties, e.g. minimum number of memory elements for a minimum complexity Viterbi decoder.
Consider an encoder with the following transfer function:

\[ G(D) = \begin{bmatrix} 1 & D^2 & D \\ D & 1 & 0 \end{bmatrix} \]

Observe that:

\[ \begin{bmatrix} 1 & D^2 & D \\ D & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & D^2 \\ D & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \frac{D}{1+D^3} \\ 0 & 1 & \frac{D^2}{1+D^3} \end{bmatrix} = Q(D)G'(D) \]
**Systematic encoders (2/2)**

- $Q(D)$ is full rank, so $u'(D) = u(D)Q(D)$ is a permutation of all possible input sequences.
- We can write:
  \[ x(D) = u(D)'G'(D) \]

  Recall that:
  \[ G'(D) = \begin{bmatrix} 1 & 0 & \frac{D}{1+D^3} \\ 0 & 1 & \frac{D^2}{1+D^3} \end{bmatrix} \]

- This encoder is said to be *systematic*.
- It can be shown that for each code there exists a systematic encoder.
Let $q(D)$ denote the least common multiple of all the denominators of the entries of the generator matrix.

Then we have that:

$$G'(D) = q(D)G(D)$$

where $G'(D)$ is an encoder which is polynomial and equivalent to $G(D)$.

Thus, every convolutional code admits a polynomial encoder.
Minimal encoders

- It can be shown that among all equivalent encoder matrices, there exists one corresponding to the minimum number of trellis states.
- The above means that its realization in controller form requires the minimum number of memory elements.
- We have seen that every encoder can be transformed into a systematic rational one.
- It can be shown that systematic encoders are minimal.
By puncturing we can obtain a higher rate code from one with a lower rate.

A fraction of symbols $\epsilon$ is punctured (i.e. not transmitted) from each encoded sequence, resulting in a code with rate $r_0/(1 - \epsilon)$.

For example, if we puncture $1/4$ of the output symbols of a rate $1/2$ code, we will get a rate $(1/2)/(3/4) = 2/3$ code.

Several rates can be obtained from the same “mother code”, making it possible to create a “universal encoder/decoder”.
Block codes from convolutional codes
In practice, a convolutional code is used to transmit a finite sequence of information bits, so its trellis must be terminated at a certain time.

At each time $t > 0$, the $n_0$ output bits of a rate $1/n_0$ polynomial encoder are a linear combination of the contents of the shift register:

$$x_t = u_t g_1 + u_{t-1} g_2 + \ldots + u_{t-\nu} g_{\nu+1}$$

The above equation can be written in a matrix form as follows:

$$x = u G_\infty$$

where

$$G_\infty = \begin{bmatrix} g_1 & g_2 & \cdots & g_{\nu+1} \\ g_1 & g_2 & \cdots & g_{\nu+1} \\ g_1 & g_2 & \cdots & g_{\nu+1} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}$$
Consider an input sequence with finite length $N$.
The first $n_0N$ output bits can be computed as:

$$x = uG_N$$

The downside of this method is that the coded symbols are not equally error protected.
This happens because for the first bits the decoder starts from a known state, thus decreasing their BER.
The exact opposite happens for the last bits in the black, increasing their BER.
To avoid the above problem, we can have the encoder end in a predefined state (usually the all-zero state).

To achieve this, we have to append a deterministic sequence at the end of the input, which forces the decoder to end in the desired state.

This sequence has length $k_0/n_0$, in order to fill the shift register(s).

Obviously, we will have a decrease in rate which may be substantial for short blocks.
Tail-biting

- We can force the encoder to start and end in the same state with a tail-biting trellis.

\[ G_N = \begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} \\ \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} \\ \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} \\ \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu} \\ \mathbf{g}_{\nu} & \mathbf{g}_{\nu+1} & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{g}_2 & \mathbf{g}_3 & \cdots & \mathbf{g}_{\nu+1} & \mathbf{g}_1 \end{bmatrix} \]

- We do not have the rate loss of zero tailing.
- The decoder complexity is increased because the starting and ending states are unknown.
Performance evaluation
We can describe the transfer function for each transition of a graph describing a convolutional code as a function of the indeterminate $X$ raised to the power of the Hamming weight of the corresponding output word.

Recall the following graph:

For example, the transfer function for the transition $\alpha \rightarrow \beta$ would be $X^2$. 
By fully reducing the graph, according to the rules we have seen, we can compute its transfer function.

The transfer function will be a polynomial of $X$:

$$T(X) = \nu_\alpha X^\alpha + \nu_\beta X^\beta + \ldots$$

The minimum exponent of $T(X)$ is called the free distance of the code, denoted $d_{\text{free}}$.

It can be shown that the error probability for the AWGN channel for large SNR can be written as:

$$P(e) \leq \nu_{d_{\text{free}}} Q\left(\sqrt{2\rho d_{\text{free}} \frac{E_b}{N_o}}\right)$$