Convolutional Codes

Telecommunications Laboratory
Alex Balatsoukas-Stimming

Technical University of Crete

November 6th, 2008

Outline

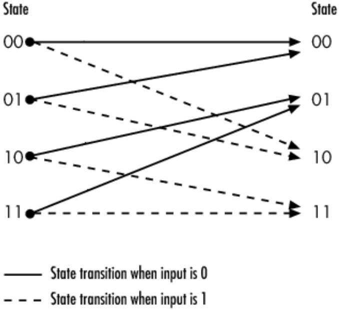
1. Trellis diagrams
2. Convolutional codes
   - A first look
   - Theoretical foundations
   - Defining convolutional codes
   - Systematic encoders
   - Polynomial encoders
   - Minimal encoders
   - Punctured convolutional codes
3. Block codes from convolutional codes
   - Direct termination
   - Zero termination
   - Tail-biting
4. Performance evaluation

Coding on a Trellis

- 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-ν binary shift register whose contents at any given time the define the state of the trellis.
- Obviously, the number of states is 2^ν

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.

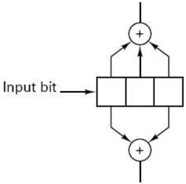
Convolutional codes

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 ν.
- 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:

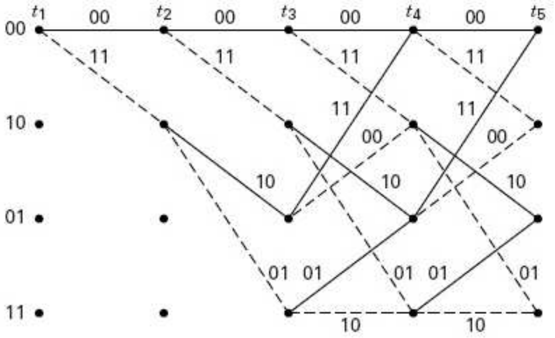


- For each input bit, we have two output bits, so the rate of the encoder is 1/2.
- The output bits are:

$$c_1 = x_1 + x_2 + x_3$$

$$c_2 = x_1 + x_3$$

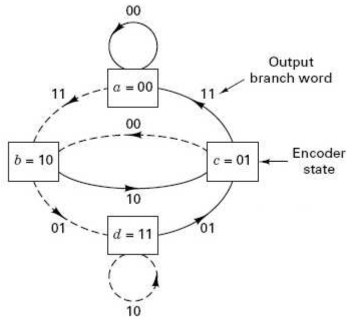
- Conventionally, the initial state is chosen as the all-zero state.



The trellis representing the above code.

State diagram

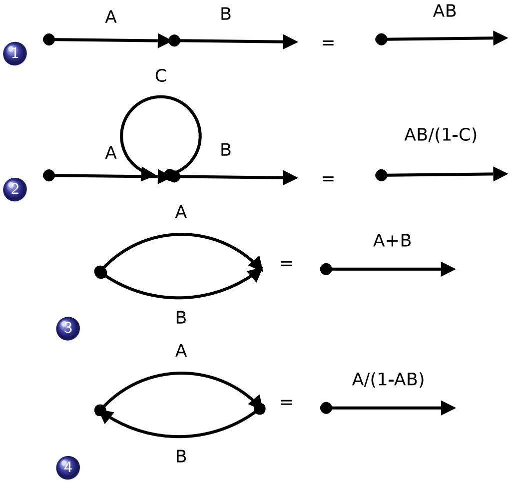
- Another representation of a convolutional code is its state diagram.
- A state diagram describes the transitions between states and the corresponding output symbols without an explicit time axis.



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:



Rate k_0/n_0 convolutional codes

- 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 reponse:

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

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

$$\mathbf{x} = \mathbf{g} * \mathbf{u}$$

Theoretical foundations (2/4)

- 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 $\mathbf{x} = \mathbf{g} * \mathbf{u}$ can be now written as:

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

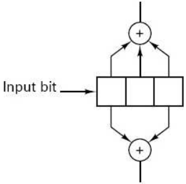
- If $g(0) = 1$ we say that the polynomial g is *delay-free*.
- $g(D)$ may have an infinite number of terms, if for example it has the form of a ratio between polynomials:

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

- 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*.

Theoretical foundations (4/4)

- We can now describe a rate k_0/n_0 convolutional code through a $k_0 \times n_0$ generator matrix $\mathbf{G}$ which contains its $k_0 n_0$ impulse responses.
- Recall the following encoder:



- We have 1 input and 2 outputs, so the generator matrix will have dimensions 1×2 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

Defining convolutional codes (2/2)

- The same convolutional code can be generated by more than one encoder.
- Let $\mathbf{Q}(D)$ denote an invertible matrix, we have:

$$\begin{aligned}\mathbf{x}(D) &= \mathbf{u}(D)\mathbf{G}(D)\\ &= \mathbf{u}(D)\mathbf{Q}(D)\mathbf{Q}^{-1}(D)\mathbf{G}(D)\\ &= \mathbf{u}'(D)\mathbf{G}'(D)\end{aligned}$$

- 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.

Systematic encoders (1/2)

- Consider an encoder with the following transfer function:

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

- Observe that:

$$\begin{bmatrix} 1 & D^2 & D \\ D & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & D^2 \\ D & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \frac{D}{1+D^3} \\ 0 & 1 & \frac{D^2}{1+D^3} \end{bmatrix} = \mathbf{Q}(D)\mathbf{G}'(D)$$

- $\mathbf{Q}(D)$ is full rank, so $\mathbf{u}'(D) = \mathbf{u}(D)\mathbf{Q}(D)$ is a permutation of all possible input sequences.
- We can write:

$$\mathbf{x}(D) = \mathbf{u}(D)'\mathbf{G}'(D)$$

- Recall that:

$$\mathbf{G}'(D) = \begin{bmatrix} 1 & 0 & \frac{D}{1+D^3} \\ 0 & 1 & \frac{D^2}{1+D^3} \end{bmatrix}$$

- This encoder is said to be *systematic*.
- It can be shown that for each code there exists a systematic encoder.

Polynomial encoders

- Let $q(D)$ denote the least common multiple of all the denominators of the entries of the generator matrix.
- Then we have that:

$$\mathbf{G}'(D) = q(D)\mathbf{G}(D)$$

  where $\mathbf{G}'(D)$ is an encoder which is polynomial and equivalent to $\mathbf{G}(D)$.
- Thus, every convolutional code admits a polynomial encoder.

Minimal encoders

- It can be shown that among all equivalent encoder matrices, there eixsts 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.

Punctured convolutional codes

- By puncturing we can obtain a higher rate code from one with a lower rate.
- A fraction of symbols ϵ 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

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:

$$\mathbf{x}_t = u_t \mathbf{g}_1 + u_{t-1} \mathbf{g}_2 + \ldots + u_{t-\nu} \mathbf{g}_{\nu+1}$$

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

$$\mathbf{x} = \mathbf{u} \mathbf{G}_\infty$$

where

$$\mathbf{G}_\infty = \begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} & & & \\ & \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} & & \\ & & \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} & \\ & & & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Direct termination

- Consider an input sequence with finite length N.
- The first $n_0 N$ output bits can be computed as:

$$\mathbf{x} = \mathbf{u}\mathbf{G}_N$$

- 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.

Zero termination

- 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.

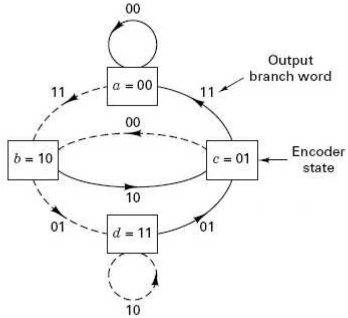
$$\mathbf{G}_N = \begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} & & & \\ & \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} & & \\ & & \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} & \\ & & & \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu+1} \\ \mathbf{g}_{\nu+1} & & & \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{\nu} \\ \mathbf{g}_{\nu} & \mathbf{g}_{\nu+1} & & & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & & & \ddots & \mathbf{g}_2 \\ \mathbf{g}_2 & \mathbf{g}_3 & \cdots & \mathbf{g}_{\nu+1} & & & \mathbf{g}_1 \end{bmatrix}$$

- 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

Performance evaluation (1/2)

- 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.

Performance evaluation (2/2)

- 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_{free}.
- It can be shown that the error probability for the AWGN channel for large SNR can be written as:

$$P(e) \leq \nu_{d_{\text{free}}} Q(\sqrt{2\rho d_{\text{free}} \frac{\mathcal{E}_b}{N_o}})$$