

3F4: Data Transmission

Handout 2: Convolutional Codes

Jossy Sayir

Probability, Systems, Information and Inference Lab Ψ^2
Department of Engineering
js851@cam.ac.uk

Lent Term 2026

1 / 17

Outline

- Convolutional codes and octal notation
- Multiple input bits and puncturing
- D -transform
- Systematic and recursive encoders
- State diagram and trellis, linearity and distance

2 / 17

Convolutional Codes

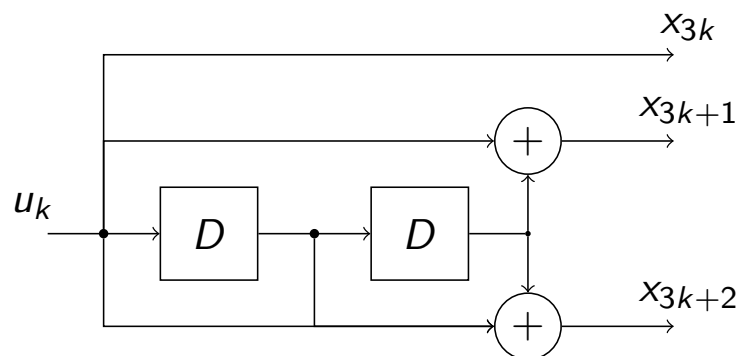
- (N, K) Linear block codes: $\underline{x} = \underline{u}G$ or $\underline{x}H = \underline{0}$ map K to N binary digits, rate $R = K/N < 1$.
- Peter Elias in 1955 proposed an alternative approach inspired by FIR/IIR filtering
- “Streaming” encoder: generates n bits for every input bit, but this is not an $(n, 1)$ block code: the outputs depend on the current input bit **and on a number of past input bits**.
- Rate: $R = 1/n$



Peter Elias, from csail.mit.edu

3 / 17

Convolutional Codes: an example

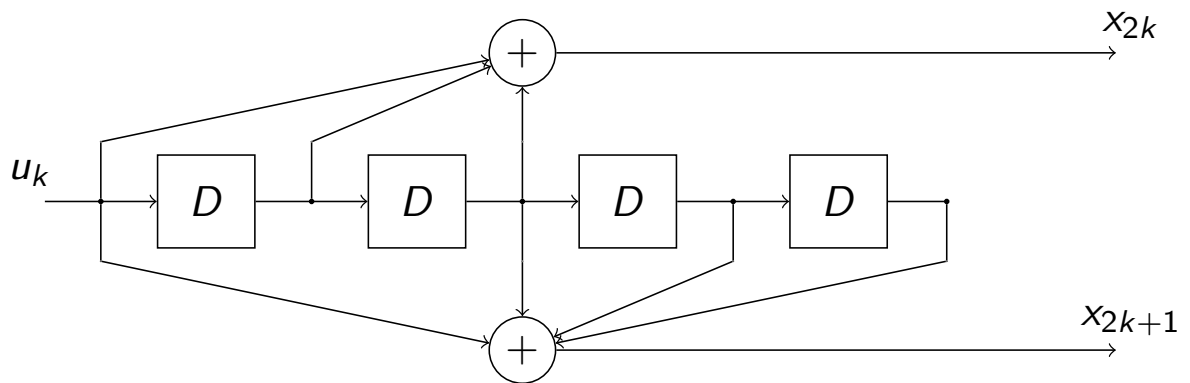


- for $k = 0, 1, 2, 3, \dots$
- For every input bit, $n = 3$ output bits \implies Rate: $R = 1/3$
- D is a delay operator
- The output bits depend on the current and 2 past inputs, u_k, u_{k-1}, u_{k-2}
- What is the output sequence for $\underline{u} = (1, 0, 1, 0, 0)$?

4 / 17

Octal Notation

- Octal numbers are groups of 3 bits, $n_8 = (b_2, b_1, b_0)$
- The bits denote the connections from the inputs $u_k, u_{k-1}, u_{k-2}, \dots$ to the pluses
- The code $(34, 27)_8$ corresponds to connections $(011'100, 010'111)$ and hence to the convolutional encoder

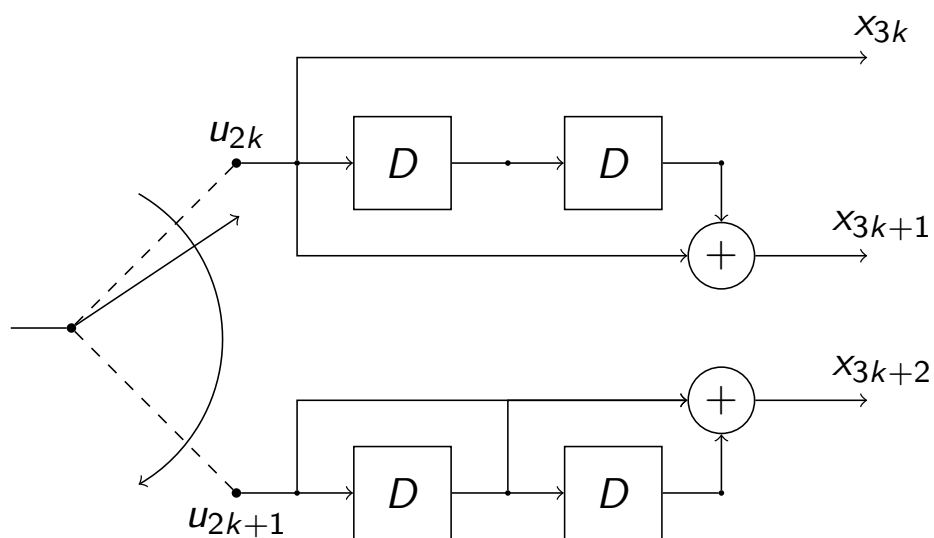


- What is the octal description of the encoder on the previous slide?

5 / 17

(n, k) encoders

- Many textbooks introduce generalised (n, k) convolutional codes



- This is a $(3, 2)$ encoder of rate $R = 2/3$
- I have never heard of anyone using such encoders in practice
- There is an easier way to build convolutional encoders with rates other than $R = 1/n$

6 / 17

Puncturing

- Define a “puncturing pattern”, e.g., $\underline{p} = (1, 1, 1, 0)$
- Parse the output code sequence into blocks of the length of \underline{p}
- Drop the bits corresponding to 0's in the puncturing pattern
- For example, if using the $(2, 1)$, $R = 1/2$ convolutional encoder in our previous example, and the puncturing pattern $\underline{p} = (1, 1, 1, 0)$, we drop one out of every 4 code digits and hence keep 3 out of 4 code digits, resulting in a code of rate

$$R = \frac{1}{2} \times \frac{4}{3} = \frac{2}{3}$$

- All known decoders can easily deal with puncturing and consider them erased bits in the original non-punctured code
- This is a much easier way to achieve rates other than $R = 1/n$ because decoder complexity is barely affected (whereas it becomes prohibitive with structures such as the one on the previous slide)

7 / 17

The D -transform

- $u_0, u_1, u_2, \dots \longrightarrow U(D) = u_0 + u_1D + u_2D^2 + \dots$
- $(c_0, c_1, c_2) \longrightarrow C(D) = c_0 + c_1D + c_2D^2$
- Sequences map to power series and vectors to polynomials
- The D transform is the z transform with $D = z^{-1}$
- It conveniently maps convolution to multiplication:
 - Let us denote as $x^{(2)}$ the output x_2, x_5, x_8, \dots of the bottom adder in our $R = 1/3$ convolutional encoder, i.e., $x_k^{(2)} = x_{3k+2}$.
 - We have

$$x_k^{(2)} = u_k + u_{k-1} + u_{k-2},$$

i.e., the output sequence $x^{(2)}$ is the convolution of the input sequence u_0, u_1, u_2, \dots with the connection vector $(1, 1, 1)$.

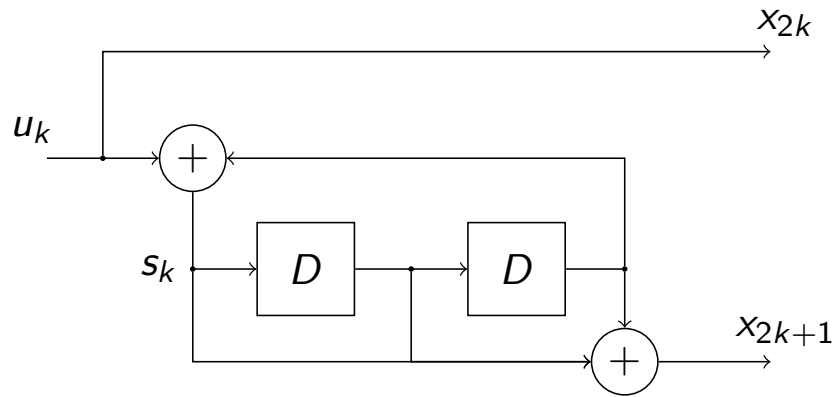
- In D transform terms, this becomes

$$X^{(2)}(D) = U(D)C(D) \text{ where } C(D) = 1 + D + D^2$$

- Unlike the z transform where we are concerned with the value of a power series for complex variables $z \in \mathbb{C}$, we will *never* attribute a value to D . The D transform only uses the formal variable D to harness multiplication and division rules for

8 / 17

Recursive Systematic Encoders



- $s_k = u_k + s_{k-2} \rightarrow u_k = s_k + s_{k-2} \rightarrow U(D) = (1 + D^2)S(D)$
and hence

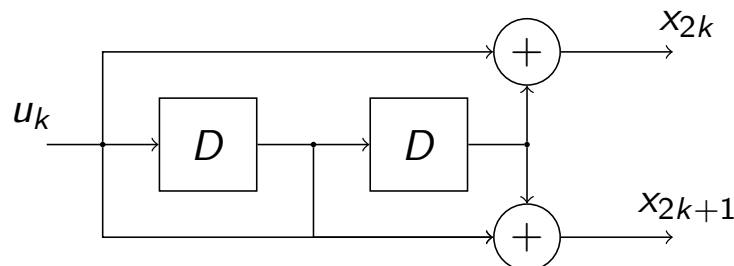
$$S(D) = \frac{U(D)}{1 + D^2} \rightarrow X^{(1)}(D) = \frac{1 + D + D^2}{1 + D^2} U(D)$$

- The encoder is called *systematic* because $x_k^{(0)} = x_{2k} = u_k$

9 / 17

Code vs. Encoder

- Consider the $(5, 7)_8$ encoder:



- $X^{(0)}(D) = (1 + D^2)U(D), X^{(1)}(D) = (1 + D + D^2)U(D)$
- Now consider

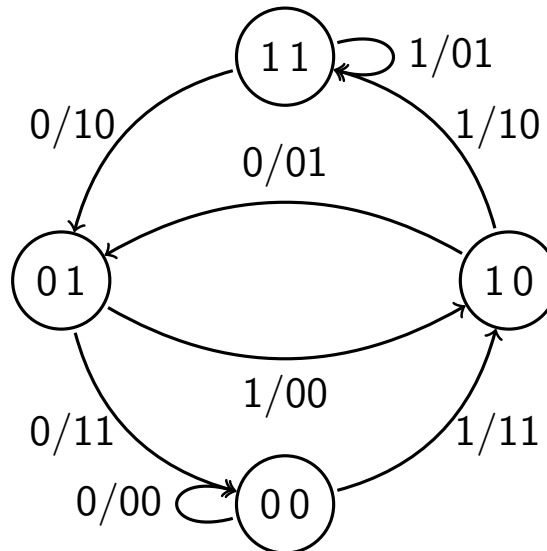
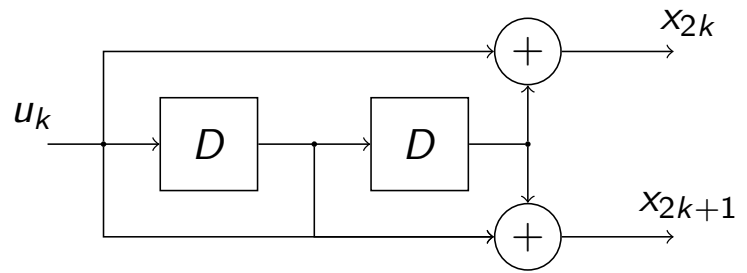
$$Y^{(0)}(D) = \frac{X^{(0)}(D)}{1 + D^2} \text{ and } Y^{(1)}(D) = \frac{X^{(1)}(D)}{1 + D^2}$$

- $Y^{(0)}(D)$ and $Y^{(1)}(D)$ are the output sequences of the recursive systematic encoder on the previous slide
- By dividing all output sequences by $1 + D^2$, we are *not* changing the set of possible sequences (rate 1 operation)
- The *codes* are the same but the *encoders* are different

10 / 17

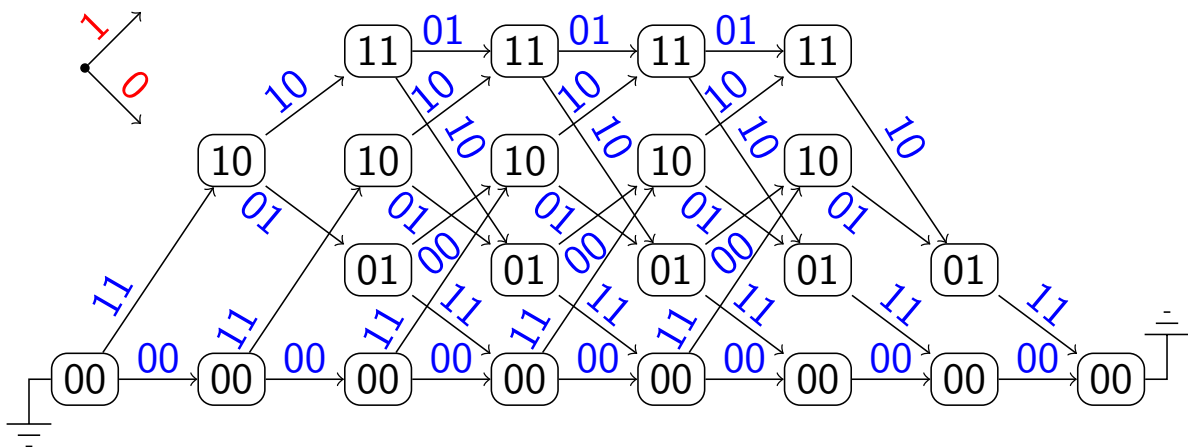
State diagram

- A convolutional encoder is a finite state machine
- Consider again the $(5, 7)_8$ encoder
- Its state diagram is



11 / 17

Trellis Diagram



- The trellis diagram is an “unwound” state diagram.
- States are in the y direction. Time (input binary digit index) is in the x direction
- Branches are labeled with the corresponding output code digits.
- No need to include the inputs digits in the labels: for every state, the outgoing branch going up corresponds to an input 1 and the outgoing branch going down corresponds to an input 0

12 / 17

A garden trellis...

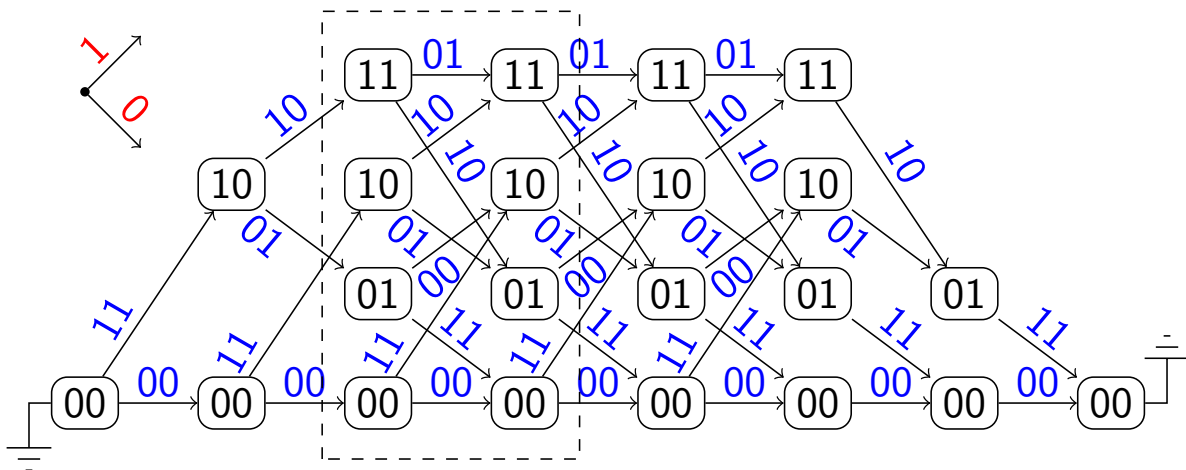
The trellis diagram is not named after a Misses or a Mister Trellis as I thought when I first learned about them but after the shape of a wooden structure used by gardeners.



From the website of AWBS Ltd. Building Supplies, Oxford

13 / 17

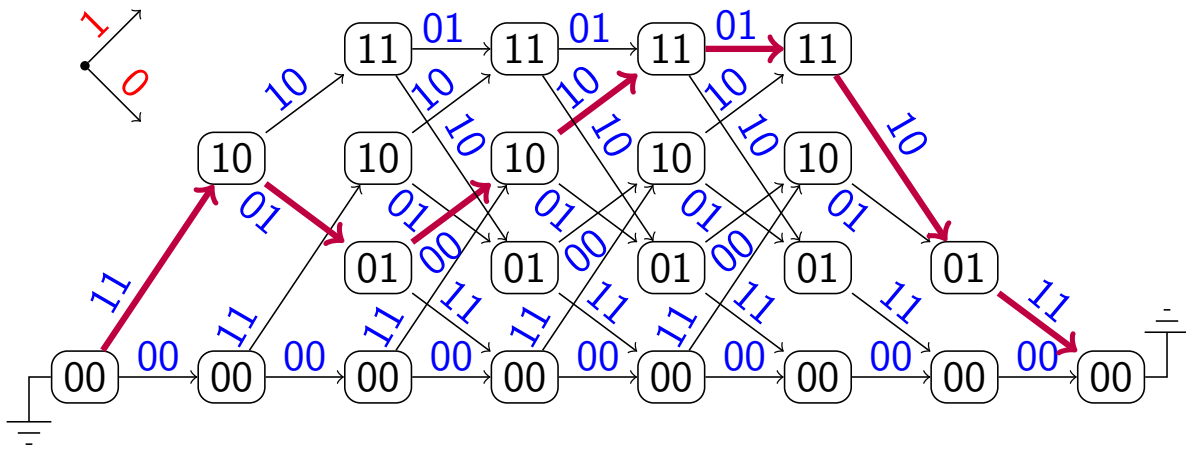
Trellis Diagram (continued)



- The trellis diagram of a convolutional encoder consists of an initial build-up until all the states are attained, followed by the repetition of a regular “trellis module”
- The start node/state is called the “root” (ground symbol)
- This trellis is terminated: 2 symbols are inserted to return the encoder to the zero state
- The final node/state is called the “toor” (root read backwards, upside down ground symbol)

14 / 17

Trellis Diagram (continued)



- Every path in the trellis corresponds to a codeword
- The highlighted path gives the codeword $\underline{x} = (1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1)$ corresponding to the information word $\underline{u} = (1, 0, 1, 1, 1)$ followed by two zeros that don't carry any information since they are needed to terminate the trellis
- We will use trellises extensively when studying decoding algorithms in the next lecture

15 / 17

Termination

- Termination affects the rate of the code since we need to insert dummy termination symbols.
- The rate of the code on the previous slides is $R = 5/14$ since we encode only 5 info symbols and obtain a codeword of length 14. This is lower than the rate $R = 1/2$ of the convolutional code.
- We terminated after 3 trellis modules for illustration purposes. In practice, termination would only occur after 100+ trellis modules so it barely affects the overall rate.
- Termination is not always used. It helps the decoder but in some applications it is impractical or unnecessary.
- For forward encoders, termination is always obtained by inserting zeros. For recursive encoders, the symbols needed to return to the zero state are state dependent and not zero in general.

16 / 17

Linear block vs. convolutional code

- A terminated convolutional code is a block code
- All operations in a convolutional code are linear so the code is linear: the sum of input sequences results in the sum of code sequences
- Everything we learned about distance and symmetry for linear block codes applies to convolutional codes.
- In a trellis, we can read out the weights (= Hamming distance from the all-zero codeword) of code sequences by considering **deviations from the all-zero path at the bottom of the trellis**