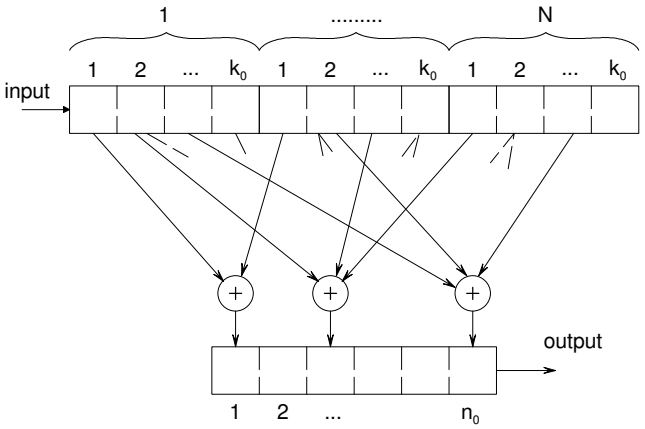
Convolutional code

- A binary convolutional encoder is a finite memory system that outputs n_0 binary digits for every k_0 information digits presented at its input.
- The code rate is defined as $R_c = k_0 / n_0$.



- The message digits are introduced k_0 at the time into the input shift register, which has Nk_0 positions.
- The modulo-q adders feed the output register with the n_0 digits and these are shifted out.
- The output depends not only on the k_0 message digits but also on $(N-1)k_0$ previous message digits.
- The parameter N is called the constraint length of the code.
- The code is called as $\left(n_0, k_0, N\right)$ convolutional code.

Representations of a convolutional code

1. Shift register - (Implementation).
2. State diagram - (Distance properties).
3. Tree - (Sequential decoding).
4. Trellis - (Viterbi algorithm).
5. Matrix - (Syndrome decoding).
6. Polynomial - (Catasthrophic/noncathastrophic).

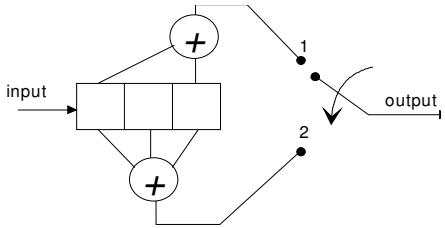
Notation on this slides

u input information bits

s coded bits

c codeword

Example



$n_0 = 2\ k_0 = 1\ N = 2$

1) Shift register.

- We have two interleaved output sequences

$$s^{(1)} = \left(s_0^{(1)}, s_1^{(1)}, \ldots\right)$$
$$s^{(2)} = \left(s_0^{(2)}, s_1^{(2)}, \ldots\right)$$
$$s = \left(s_0^{(1)}, s_0^{(2)}, s_1^{(1)} s_1^{(2)}, \ldots\right)$$

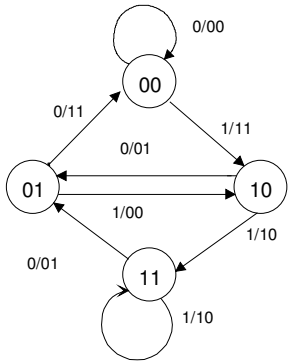
- At time $t = l$

$$s_l^{(1)} = 1 \cdot u_l + 0 \cdot u_{l-1} + 1 \cdot u_{l-2} = \sum_{i=0}^{2} g_i^{(1)} u_{l-i}, \quad g^{(1)} = (101)$$

$$s_l^{(2)} = 1 \cdot u_l + 1 \cdot u_{l-1} + 1 \cdot u_{l-2} = \sum_{i=0}^{2} g_i^{(2)} u_{l-i}, \quad g^{(2)} = (111)$$

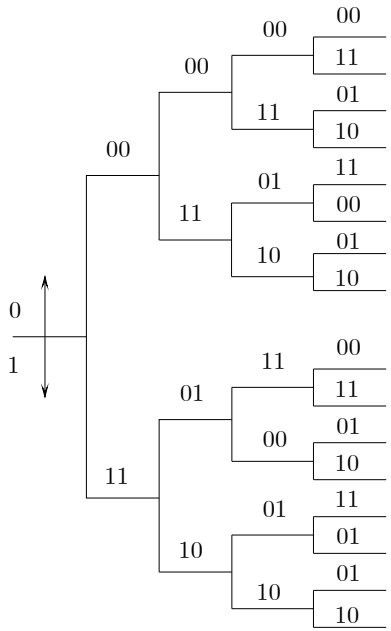
2) State diagram

- Nodes are states, content of the shift register.
- Each state has 2^{k_0} output branches. These are the transition having nonzero probabilities.
- As new input arrives the system moves to new state and generates n_o output bits.
- The example code has $2^N = 4$ states.



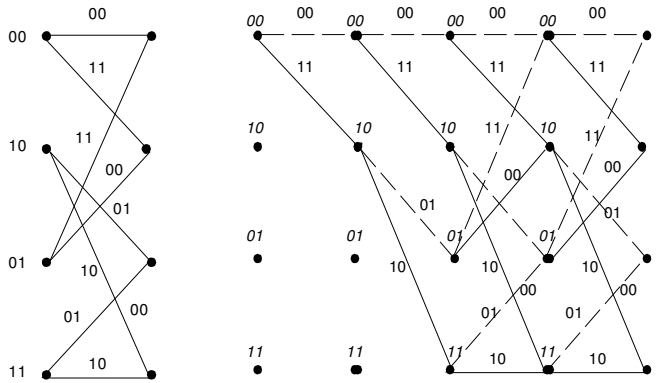
3) Tree.

- Shows the time progression of the state sequences.
- Index the states with the time t .



4) Trellis

- The structure in the tree repeats itself.
- Index the states with both the time t and state index m.
- Shows the time progression of the state sequences.



5) Matrix.

$$\mathbf{s} = \mathbf{uG} = \begin{bmatrix} u_0 & u_1 & u_2 & \dots \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \vdots & \vdots \end{bmatrix}$$

- The columns are shifted versions of the generation matrix.
- If we have a finite amount of input bits i, the matrix has size $(i+1)\times\big(2(i+1)\big)$ and the generated code is a block code.

6) Polynomial.

Output is convolution of input and convolution of $g^{(i)}$.

Take transform so that output will be product of these terms.

Appropriate transform is Z-transform.

$u(z) = u_0 + u_1 z^1 + u_2 z^2 \dots$

$g^{(1)}(z) = 1 + z^2$

$g^{(2)}(z) = 1 + z^1 + z^2$

$g = \begin{bmatrix} 5 & 7 \end{bmatrix}$

$s(z) = \begin{bmatrix} u_0 + u_1 z^1 + u_2 z^2 \end{bmatrix} \begin{bmatrix} 1 + z^2 & 1 + z^1 + z^2 \end{bmatrix}$

assume $u = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$

$s(z) = \begin{bmatrix} 1 + z^2 \end{bmatrix} \begin{bmatrix} 1 + z^2 & 1 + z^1 + z^2 \end{bmatrix}$

$\quad = \begin{bmatrix} 1 + z^4 & 1 + z^1 + z^3 + z^4 \end{bmatrix}$

By taking the inverse transform we get:

$s^{(1)} = \begin{pmatrix} 1\,0\,0\,0\,1 \end{pmatrix}$

$s^{(2)} = \begin{pmatrix} 1\,1\,0\,1\,1 \end{pmatrix}$

$s = \begin{pmatrix} 11\,01\,00\,01\,11 \end{pmatrix}$

A list of good polynomials for half rate code (from Proakis "Digital Communications".)

Constraint lenght	generator		d_{free}	Best possible d_{free}
3	5	7	5	5
4	15	17	6	6
5	23	35	7	8
6	53	75	8	8
7	133	171	10	10
8	247	341	10	11

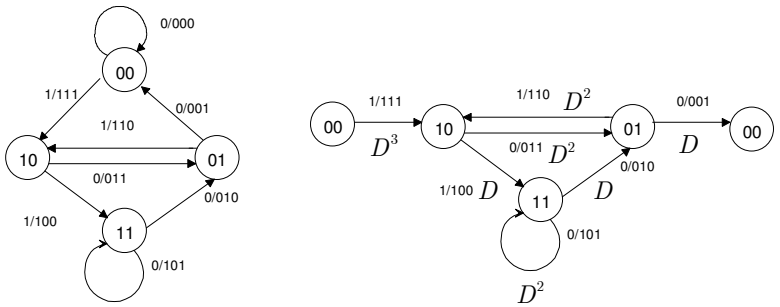
Distance properties of convolutional codes.

- Error detection and error correction properties directly related to the distance of the encoded sequence.
- Due to the uniform error property of linear codes we assume that all zero sequence is transmitted in order to determine the performance of the convolutional code.
- Common performance measures of a convolutional code are: column distance function, minimum distance, and minimum free distance.
- Column distance d_i is the minimum hamming distance between all pairs of output sequences truncated at length i given that the input sequences corresponding to the pair of outputs differ in the first k bit blocks. For linear convolutional code

  $d_i = \min\left\{w\left([y]_i\right) \mid [x]_i = 0\right\}$.
- The minimum distance $d_{\min}$ of an (n,k) convolutional code with constraint length N is the column distance function evaluated at $i = N$.
- The minimum free distance, d_{free}, is the minimal Hamming distance between all pairs of complete convolutional code words.

Weight enumeration function.

- Weight enumerator function, $T(D)$, of the code output sequence weights is a sequence that gives all the information about the weights of the paths starting in the state diagram at the state S_1 and merging again in S_1.
- For calculating the $T(D)$ we modified state diagram
  - The edges are labeled with an indeterminant D raised to the exponent of the weight of the encoded sequence of that state transition.
  - The self loop S_1 is eliminated by splitting the state into two states.

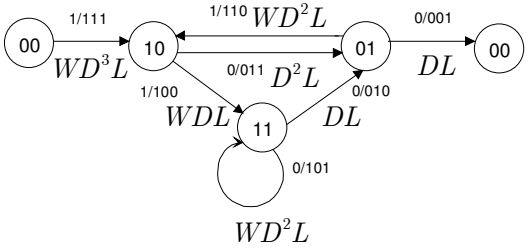


$$T(D) = \frac{2D^6 - D^8}{1 - \left(D^2 + 2D^4 - D^6\right)} = 2D^6 + D^8 + 5D^{10} + \ldots = \sum_{d=d_f}^{\infty} A_d D^d$$

A_d is number of paths with weight d diverging from state S_1.

Input-output weight enumeration function.

- Weight function $T_3(W,D,L)$ that also considers the exponent of input data frame W and the parameter L that counts length of the



paths.

$$T_3(W,D,L) = WD^6L(1+WL) + W^3D^8L^5 + \ldots$$

$$= \sum_{w=1}^{\infty}\sum_{d=d_f}^{\infty}\sum_{l=1}^{\infty} C_{w,d,l} W^w D^d L^l$$

$C_{w,d,l}$ is the number of paths diverging from state S_1 and remerging into it later generated by an information sequence of weight w, having weight d, and with length l.

Systematic recursive convolutional codes.

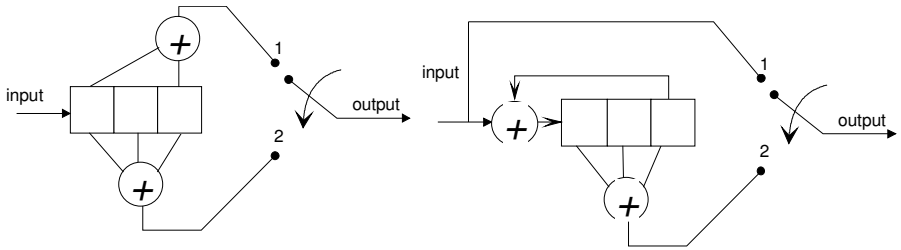
$s^{(1)}(z) = u(z)\,g^{(1)}(z)$

$s^{(2)}(z) = u(z)\,g^{(2)}(z)$

By dividing both sequences with $g^{(1)}(z)$ we get a new sequences where one contains the systematic bit and the other is generated by a recursive digital filter.

$s^{(1)}(z) = u(z)$

$$s^{(2)}(z) = u(z)\frac{g^{(2)}(z)}{g^{(1)}(z)}.$$



Bit distance spectrum of $\left(A_d^{(b)}\right)$ of both codes.

d	5	6	7	8	9	10	11	12	13	14	15
$A_d^{(b)}, NS$	1	4	12	32	80	192	448	1024	2304	5120	11264
$A_d^{(b)}, RS$	2	6	14	32	72	160	352	768	1664	3584	7680

In low SNR the recursive coder will perform better.

Importance of a code spectrum

A linear code

A linear code with length n and rank k is a linear subspace C with dimension k of the vector space F_n^k

For a linear code the minimum weight describes also the minimum distance between the code words.

The error probability can be described by the code words distance to all zero code word.

The minimum distance is described by Hamming weight – minimum weight over all the code word.

Probability of error for an error path

ML probability for all zero code word is multiplication of the probabilities at each trellis section

$$p_0 = \begin{array}{c} p(y_1 \mid s_0 = S_0, s_1 = S_0)p(y_2 \mid s_1 = S_0, s_2 = S_0) \\ \ldots p(y_k \mid s_{k-1} = S_0, s_k = S_0)\ldots p(y_N \mid s_{N-1} = S_0, s_N = S_0) \end{array}$$

If an error path has distance d from all zero code word it differs from all zero code word at d positions.

ML probability for erroneous codeword for example

$$p_1 = \begin{array}{l} p(y_1 \mid s_0 = S_0, s_1 = S_0)p(y_2 \mid s_1 = S_0, s_2 = S_0) \\ \ldots p(y_k \mid s_{k-1} = S_0, s_k \neq S_0)\left(\prod_{i=1}^{d-2} p(y_{k+i} \mid s_{k+i-1} \neq S_0, s_{k+i} \triangleq S_0)\right) \\ p(y_{k+d-1} \mid s_{k+d-1-1} \neq S_0, s_{k+d-1} = S_0) \\ \ldots p(y_N \mid s_{N-1} = S_0, s_N = S_0) \end{array}$$

Maximum likelihood decision error when $p_0 > p_1$

The ML error is when the multiplication of probabilities is higher for "wrong" codeword. Since they are different only at some positions we can compare multiplications at these positions where they differ

$$p_0 \lessgtr p_1$$

$$\prod_{c_k^0} p(y_k \mid s_{k-1} = S_0, s_k = S_0) \lessgtr \prod_{c_k^1} p(y_k \mid s_{k-1} \neq S_0, s_k \neq S_0)$$

Probability of error for an error path in Gaussian channel with BPSK signalling

$$p\left\{c_i \rightarrow c_j\right\} = \frac{1}{2} erfc\left(\sqrt{\frac{4Ed_{ij}}{4N_0}}\right) = Q\left(\sqrt{\frac{4Ed_{ij}}{2N_0}}\right)$$

where c describes the path in trellis and d_{ij} describes hamming distance between the paths.

In each trellis section we could prefer either transition that belongs into correct codeword or transition that belongs into an erroneous path.
If we prefer an erroneous path we do $c_{j,d}$ errors. (some information bits in the erroneous path are the same as bits in the correct path).
The average bit errors we do over all the possible erroneous paths is

$$e = \sum_{j=1}^{\infty} c_{j,d} P\left(c_i \to c_j\right)$$

Now we have to average this result over all the trellis sections.
In a code with rate k/n each trellis section corresponds to k information bits.
For information block length N there are total N/k sections.

In each section we could do an error. There are total N sections
N/k * e
This results should be averaged over the probability of information bit error 1/N. (N information bits and error in each of them is equally possible).

The total error probability

$$\begin{aligned} p(e) &\le \sum_{j=1}^{\infty} c_{j,d} P\left(c_i \to c_j\right) \\ &= \sum_{j=1}^{k} c_{j,d} P\left(c_i \to c_j\right) + \sum_{j=k+1}^{\infty} c_{j,d} P\left(c_i \to c_j\right) \\ &\le \sum_{j=1}^{k} c_{j,d} P\left(c_i \to c_j\right) + C \end{aligned}$$

The union Bhattacharyya bound

- A simpler form of the union bound can be obtained by using a bound to the pair wise error probability exceeds the exact value.

$$p\left\{s_i \to s_j\right\} = \frac{1}{2} erfc\left(\sqrt{\frac{d_{ij}}{4N_0}}\right) \le \exp\left\{-\frac{d_{ij}}{4N_0}\right\}$$

$$p(e) \le \frac{1}{M} \sum_{i=1}^{M} \sum_{j \ne i} \exp\left\{-\frac{d_{ij}}{4N_0}\right\}$$

Some other useful bounds

$$\frac{1}{\sqrt{2\pi}X}\left(1 - \frac{1}{X}\right)\exp\left(-\frac{X}{2}\right) < \frac{1}{2} erfc\left(\sqrt{\frac{X}{4N_0}}\right) \le \frac{1}{\sqrt{2\pi}X} \exp\left\{-\frac{X}{4N_0}\right\}$$