Week 4 : Session 1.

CONVOLUTIONAL CODES : AN INTRODUCTION

Outline :

- Introduction
- Encoding for convolutional code

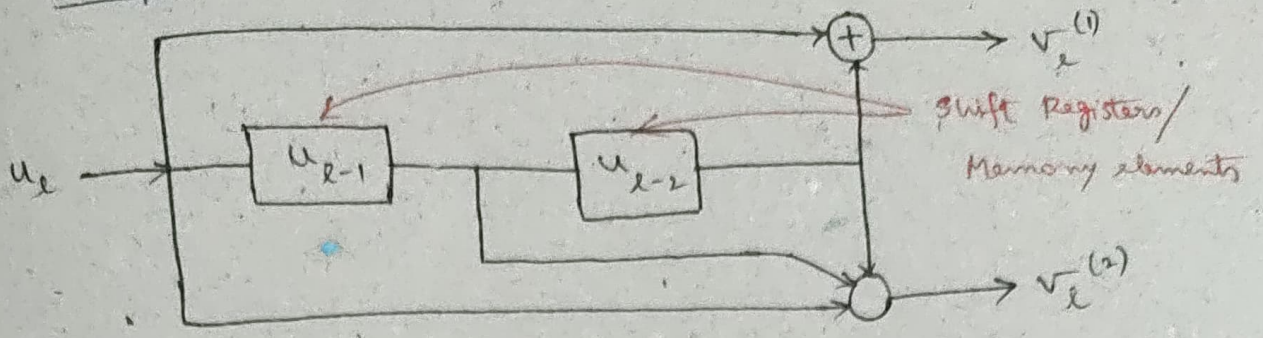
In the previous section, we talked about Block codes, properties of Block codes, how the error correcting and error detecting capability of the code is dependent on the minimum distance of the code, and how we can encode and decode codes using some simple block codes. Now, lets shift our attention to convolutional codes.

Introduction :

- ⊙ Unlike a Block code where we partition data into blocks of k-bit, a convolutional encoder processes the information sequence continuously.
- ⊙ Unlike a Block code, which is memoryless, where the output only depends on the current k bits, here in case of convolutional codes, the n-bit encoder output at a particular time depends not only on the k-bit information sequence, but also on m previous input blocks. (ie) a convolutional encoder has a memory order of m.
- ⊙ The set of sequences produced by a k-input, n-output encoder of memory order m is called an (n, k, m) convolutional code.

The values of n and k are much smaller for convolutional codes compared to the block codes.

Example: Rate $= \frac{1}{2}$ convolutional encoder.



Let $k = 1$ (No. of input)
$n = 2$ (No. of output) and
$m = 2$ (No. of Memory elements).

This circuit generates a (2, 1, 2) convolutional code.

Input : u_l

outputs : $v_l^{(1)} = u_l + u_{l-2}$

$v_l^{(2)} = u_l + u_{l-1} + u_{l-2}$

If ν_i is the length of the i^{th} shift register in a convolutional encoder with k input sequences, $i = 1, 2, \ldots, k$, then

$$m = \max_{1 \le i \le k} \nu_i$$

- The parameter m is known as Memory order of the code.
- The ratio $R = \frac{k}{n}$ is known as the code Rate.
- The overall constraint length ν of the encoder is defined as

$$\nu = \sum_{1 \le i \le k} \nu_i$$

Encoding of $(n, 1, m)$ convolutional code.

In this case, a singel information sequence

$$u = (u_0, u_1, \ldots, u_l, \ldots)$$

is encoded into n output sequences.

$$v^{(1)} = (v_0^{(1)}, v_1^{(1)}, \ldots, v_l^{(1)}, \ldots)$$
$$v^{(2)} = (v_0^{(2)}, v_1^{(2)}, \ldots, v_l^{(2)}, \ldots)$$
$$\vdots$$
$$v^{(n)} = (v_0^{(n)}, v_1^{(n)}, \ldots, v_l^{(n)}, \ldots)$$

The n output sequences are interleaved to form a single code sequence.

$$v = (v_0, v_1, \ldots, v_l, \ldots) \text{ , where } v_l = (v_l^{(1)}, v_l^{(2)}, \ldots, v_l^{(n)})$$

The code is specified by a set of n generator sequences of length $m+1$.

$$g^{(1)} = (g_0^{(1)}, g_1^{(1)}, \ldots, g_m^{(1)})$$

$$g^{(2)} = (g_0^{(2)}, g_1^{(2)}, \ldots, g_m^{(2)})$$

$$\vdots$$

$$g^{(n)} = (g_0^{(n)}, g_1^{(n)}, \ldots, g_m^{(n)})$$

The output sequence is the discrete convolution of the information sequence u and the generator sequence $g^{(i)}$. (ii)

$$v^{(i)} = u * g^{(i)}, \quad 1 \le i \le n.$$

and $$v_l^{(i)} = u_l g_0^{(i)} + u_{l-1} g_1^{(i)} + \ldots + u_{l-m} g_m^{(i)}$$

$$= \sum_{j=0}^{m} u_{l-j} g_j^{(i)}$$

Example:

Consider a Rate $= \frac{1}{2}$, (2, 1, 2) Convolutional code specified by the following generator sequences

$g^{(1)} = (1\ 0\ 1)$ ⟶ Generator for $v_l^{(1)}$

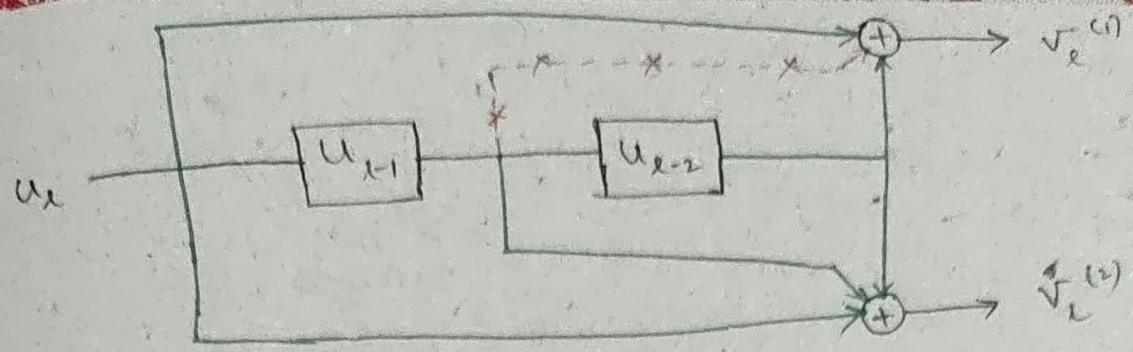
$g^{(2)} = (1\ 1\ 1)$ ⟶ Generator for $v_l^{(2)}$

Let the information sequence $u = (1\ 0\ 1\ 1\ 1\ 0\ 0\ \ldots)$.

The output sequences are given by

$v^{(1)} = (1\ 0\ 0\ 1\ 0\ 1\ 1\ \ldots)$

$v^{(2)} = (1\ 1\ 0\ 0\ 1\ 0\ 1\ \ldots)$



The code sequence can be written as (interleaved)

$$v = (11, 01, 00, 10, 01, 10, 11, \ldots)$$

Representation of encoding matrix

⊙ Matrix form.

$$v = u\,G$$

$$G = \begin{bmatrix} g_0 & g_1 & \cdots & \cdots & g_m & & \\ & g_0 & g_1 & \cdots & \cdots & g_m & \\ & & \ddots & & & & \ddots \end{bmatrix}$$

where, $g_i = \left(g_i^{(1)} \; g_i^{(1)} \cdots g_i^{(n)}\right), \quad 0 \le i \le m.$

Example: For $(2,1,2)$ convolutional code with

$$g^{(1)} = \left(g_0^{(1)} \; g_1^{(1)} \; g_2^{(1)}\right) = (1\,0\,1)$$

$$g^{(2)} = \left(g_0^{(2)} \; g_1^{(2)} \; g_2^{(2)}\right) = (1\,1\,1),$$

the generator matrix is given by

$$G = \begin{bmatrix} 11 & 01 & 11 & & & \\ & 11 & 01 & 11 & & \\ & & 11 & 01 & 11 & \\ & & & 11 & 01 & 11 \\ & & & & \ddots & \ddots \end{bmatrix}$$

For $u = (10111000\ldots)$,

$$v = uG = (11, 01, 00, 10, 01, 10, 11, 00, 00, \ldots)$$

Another way of generating the same output sequence using the delay operator D (Transform Domain).

⊙ Polynomial representation:

$$v^{(i)}(D) = u(D)\, g^{(i)}(D), \quad 1 \le i \le n$$

$$v(D) = v^{(1)}(D^n) + D\, v^{(2)}(D^n) + \dots + D^{n-1} v^{(n)}(D^n)$$

↑ output coded sequence in the Transform domain.

Time Domain	Transform Domain
Generator Sequences: $g^{(1)} = (1\ 0\ 1)$, $g^{(2)} = (1\ 1\ 1)$	$g^{(1)}(D) = 1 + 0D^1 + D^2 = 1 + D^2$
	$g^{(2)}(D) = 1 + D + D^2$
Information sequence → $u = (1011100\dots)$	$u(D) = 1 + D^2 + D^3 + D^4$
Output sequences: $v^{(1)} = u * g^{(1)}$	$v^{(1)}(D) = u(D)\, g^{(1)}(D)$
$= (10010110\dots)$	$= (1+D^2+D^3+D^4)(1+D^2)$
	$= 1 + D^3 + D^5 + D^6$
$v^{(2)} = u * g^{(2)}$	$v^{(2)}(D) = u(D) \cdot g^{(2)}(D)$
$= (11\ 0\ 0\ 1\ 0\ 1\ 0\ 0\dots)$	$= 1 + D + D^4 + D^6$

$$1 + D^2 + D^3 + D^4 + D^2 + D^4 + D^5 + D^6$$

$\because D^2 + D^2 = 0$, $D^4 + D^4 = 0$ } Mod-2 Addition

$$\Rightarrow 1 + D^3 + D^5 + D^6$$

The encoding equations can alternatively written as,

$$v(D) = u(D^n)\, g(D)$$

where,

$$g(D) \triangleq g^{(1)}(D^n) + D\, g^{(2)}(D^n) + D^2 g^{(3)}(D^n) + \dots$$

$$\dots + D^{n-1} g^{(n)}(D^n).$$

Example : Time Domain

$$v = uG = (11, 01, 00, 10, 01, 10, 11, 00, 00, \ldots)$$

Example : Transform Domain

$$v(D) = u(D^2)\, g(D) = (1 + D^4 + D^6 + D^8)(1 + D + D^3 + D^4 + D^5)$$
$$= 1 + D + D^3 + D^6 + D^9 + D^{10} + D^{12} + D^{13}$$
$$= 11\,01\,00\,10\,01\,10\,11\,00\ldots$$

$u(D) = 1 + D^2 + D^3 + D^5$
$\Rightarrow u(D^2) = 1 + D^4 + D^6 + D^{10}$

$g^{(1)}(D) = 1 + D^2 \Rightarrow g^{(1)}(D^2) = 1 + D^4$
$g^{(2)}(D) = 1 + D + D^2 \Rightarrow g^{(2)}(D^2) = 1 + D^2 + D^4$
$\therefore g(D) = g^{(1)}(D^2) + D\, g^{(2)}(D^2)$
$= (1 + D^4) + D(1 + D^2 + D^4)$
$= 1 + D + D^3 + D^4 + D^5$

So, we can see that, whether we find output using Time Domain operation (or) Transform Domain operation, we get the same output. And in case of convolutional code, it is easier to represent it in the Transform Domain.

Week 4 : Session 2:

STATE DIAGRAM & TRELLIS DIAGRAM

In the last section we talked about convolutional code, and how to encode the convolutional code. A convolutional code has Memory, so it is represented using a sequential circuit. And we showed how we can encode a sequence using convolutional code, and we represented convolutional code using a shift register. So, any system which has memory can be represented using its corresponding State diagram and Trellis diagram.

So, in this section, we're going to talk about how to draw the State diagram of a convolutional code, and from the State diagram how we can draw the Trellis diagram of a convolutional code.

Since a convolutional encoder is a sequential circuit, it can be described by a state diagram.

The state of the encoder is defined as its shift register contents. For a $(n, 1, m)$ code, at time instant l, the state is defined by the m-tuple.

$$S_l = (u_{l-1}, u_{l-2}, \ldots, u_{l-m})$$

where, $(u_{l-1}, u_{l-2}, \ldots, u_{l-m})$ are the m message bits stored in the shift register.

There are 2^m number of possible states for a $(n, 1, m)$ convolutional code.

The output of a convolutional code at each time instant l, depends on the input and the current state.

$$V_l = f(u_l, S_l).$$

The convolutional encoder undergoes a state transition whenever a new information bit is input to the encoder.

Time unit	Message bit	State
l	u_l	$S_l = (u_{l-1}, u_{l-2}, \ldots, u_{l-m})$
$l+1$	u_{l+1}	$S_{l+1} = (u_l, u_{l-1}, \ldots, u_{l-m+1})$

A state transition is represented by a directed edge connecting two states, S_l and S_{l+1}.

The state transitions are labelled with the information and coded bits corresponding to that transition.

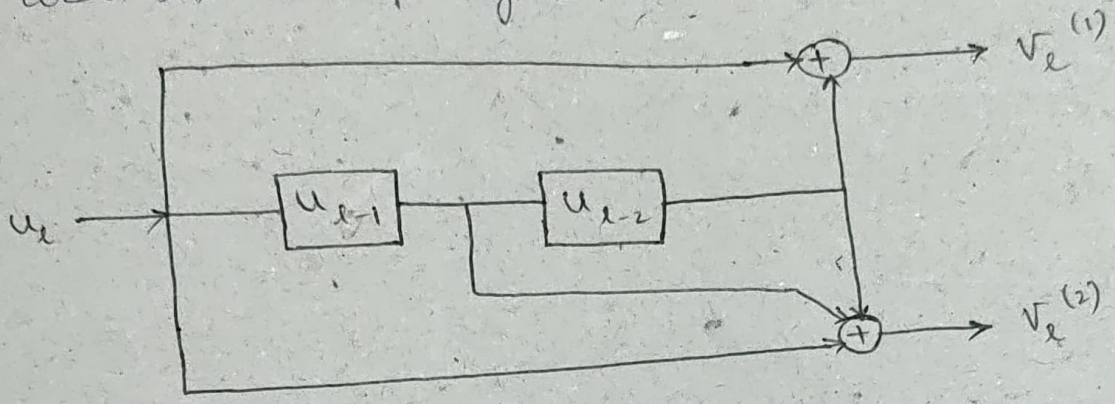
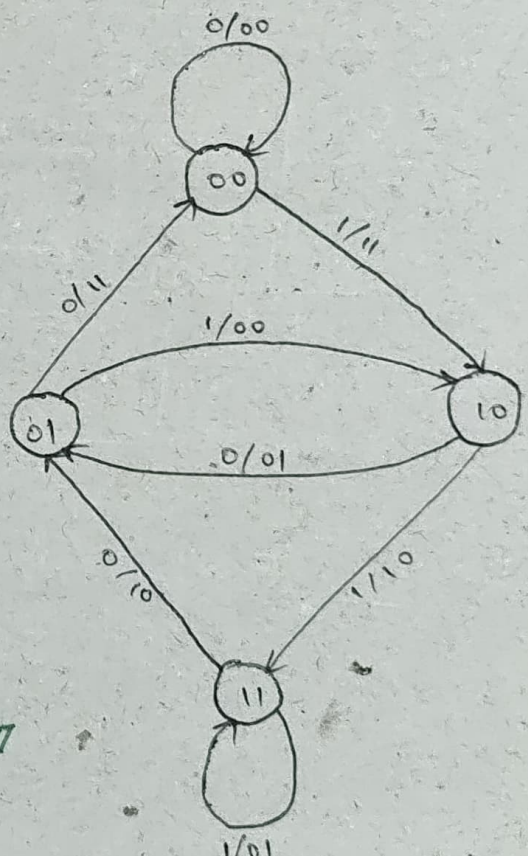


Fig: (2, 1, 2) Convolutional code

Input	Initial state	Next state	Output
0	0 0	0 0	0 0
1	0 0	1 0	1 1
0	0 1	0 0	1 1
1	0 1	1 0	0 0
0	1 0	0 1	0 1
1	1 0	1 1	1 0
0	1 1	0 1	1 0
1	1 1	1 1	0 1

State Table



State Diagram

There are four states : (0,0), (0,1), (1,1) and (1,0) for the (2,1,2) convolutional code.

The Trellis diagram of a convolutional code includes a time-dimension.

In each trellis section, the states are represented twice, once at time l, and another at time $l+1$. There is a branch connecting the state S_l at time l to the state S_{l+1} at time $l+1$ if there exists an input u_l at time l that drives the encoder to state S_{l+1} from state S_l.

A trellis diagram for a convolutional code is obtained by joining the trellis sections at different time units.

Each codeword is made up of the labels on the trellis transitions that correspond to a specific path through the trellis.

Example :

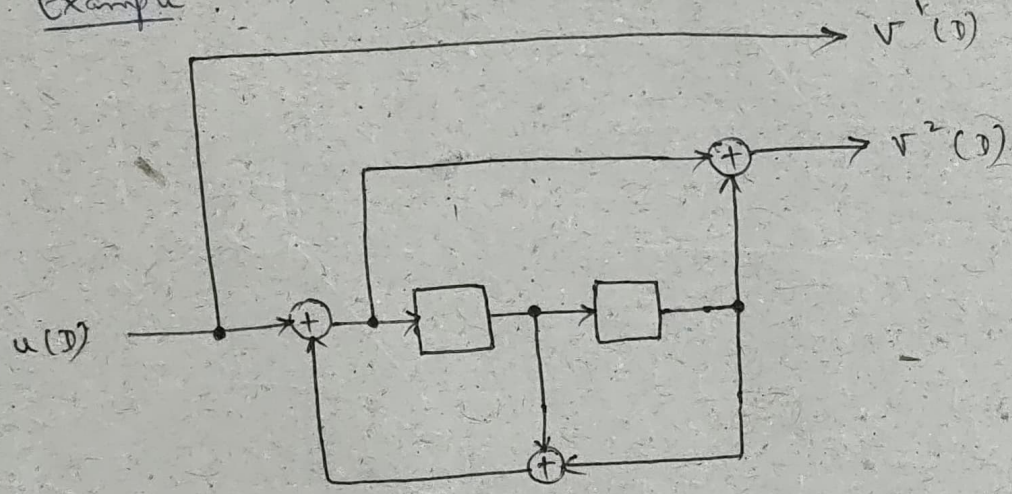


Fig : Rate, $R = \frac{1}{2}$ Systematic Feedback Convolutional Encoder.

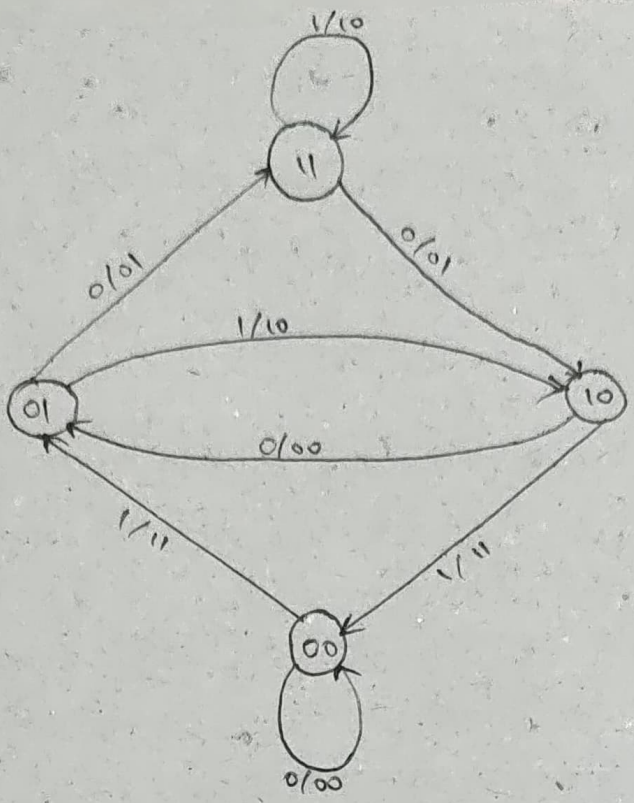


Fig: State Diagram.

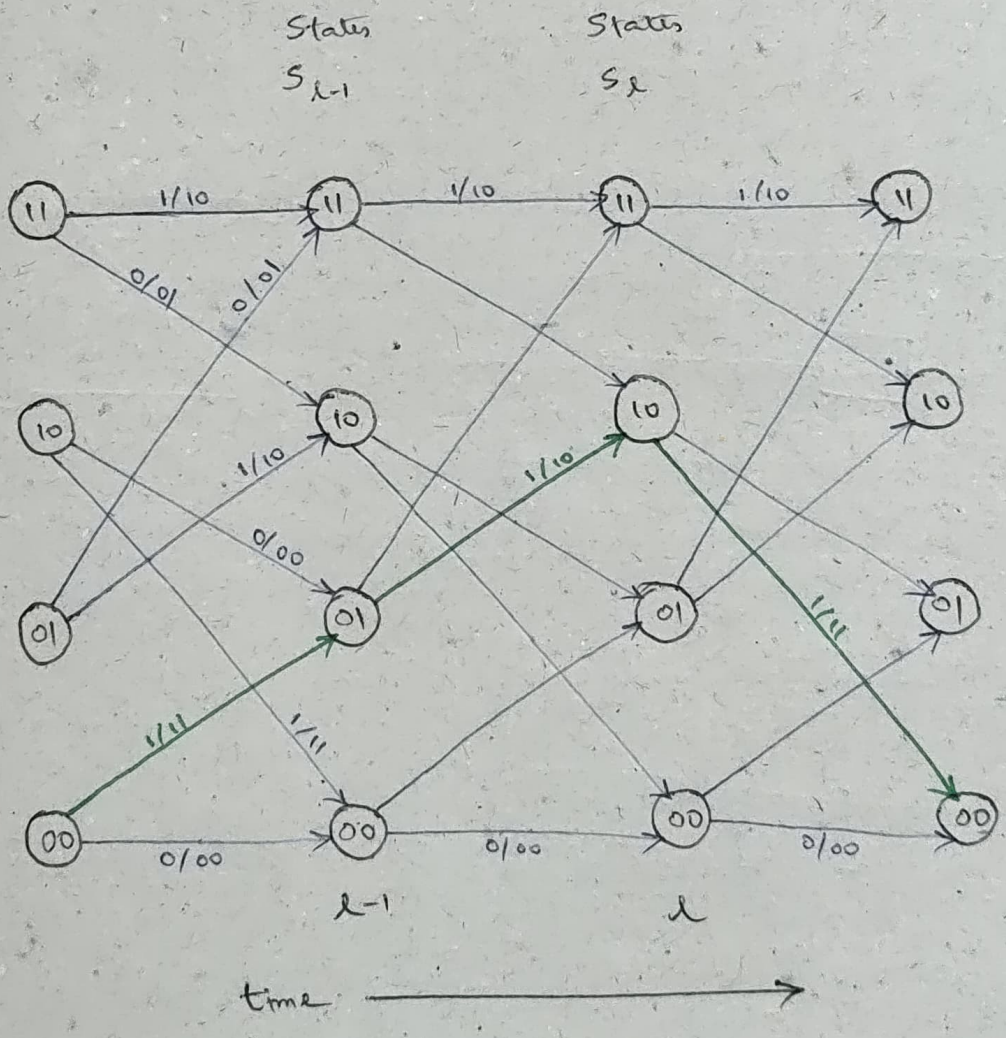


Fig: Trellis Diagram.

Week 4 : Session 3.

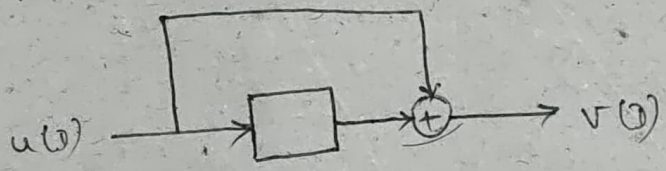
CLASSIFICATION OF CONVOLUTIONAL ENCODER

① FEED FORWARD ENCODER

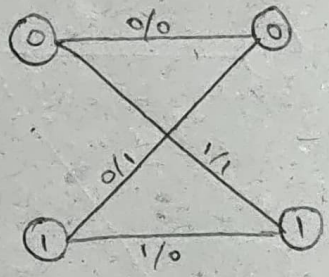
⊙ The Encoder corresponding to a polynomial generator matrix does not contain any feedback path from the output to the input, and hence it is known as a feed forward encoder.

⊙ The output of a feed forward encoder can be represented as a linear combination of the current input and a finite number of past inputs. This is also referred as non-recursive encoder.

⊙ In figure, the encoder diagram of a rate $R = 1$, 2-state feed forward encoder with generator matrix $G(D) = [1 + D]$ is shown using a shift register implementation.



(a) : R=1, 2-state Feedforward Encoder



(b) State diagram.

$$V(D) = u(D) + D\,u(D)$$

$$= (1 + D)\,u(D)$$

$$\therefore G(D) = [1 + D] \longrightarrow \text{Polynomial}$$

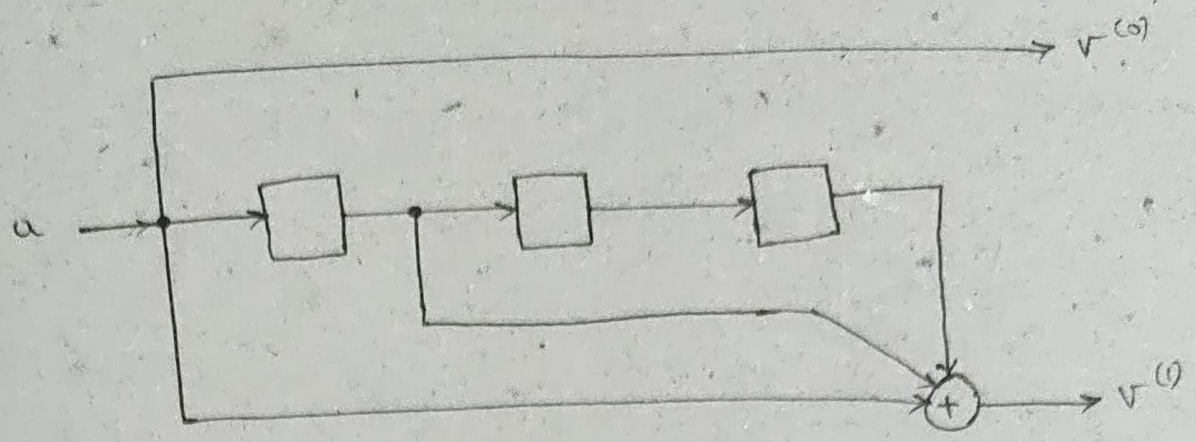


Fig: Another Example for Feed forward Encoder

$$V^{(0)}(D) = u(D)$$

$$V^{(1)}(D) = u(D) + D\,u(D) + D^3\,u(D)$$

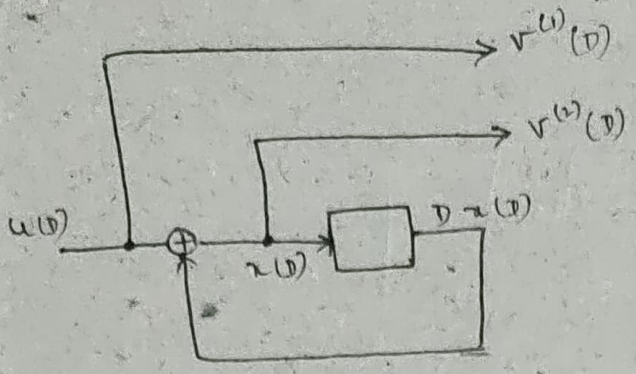
$$= u(D)\,(1 + D + D^3)$$

$$\therefore G(D) = \begin{bmatrix} 1 & 1+D+D^3 \end{bmatrix} \rightarrow \text{Polynomial}$$

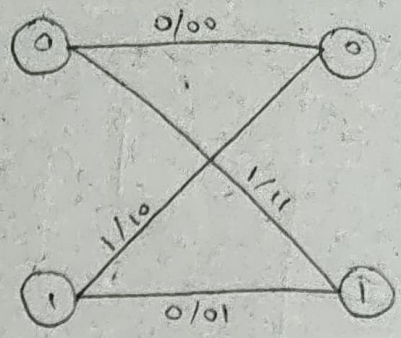
---

(2) FEEDBACK ENCODER.

- The encoder corresponding to a rational generator matrix with atleast one non-polynomial transfer function contains a feedback path, and is known as a feedback encoder.
- The output of a feedback encoder can be represented as a linear combination of past inputs as well as past outputs. Hence the output depends on infinite number of past inputs. This is also sometimes refered as recursive encoder.
- In figure, the encoder diagram of a rate $R = \frac{1}{2}$, 2-state feedback encoder with generator matrix $G(D) = \begin{bmatrix} 1 & \frac{1}{1+D} \end{bmatrix}$ is shown.



(a) $R = \frac{1}{2}$, 2-state Feedback Encoder



(b) State diagram

$$V^{(1)}(D) = u(D)$$

$$V^{(2)}(D) = x(D) = u(D) + D\,x(D)$$

Adding $D\,x(D)$ on both sides,

$$x(D) + D\,x(D) = u(D) + \underbrace{D\,x(D) + D\,x(D)}_{0} = u(D)$$

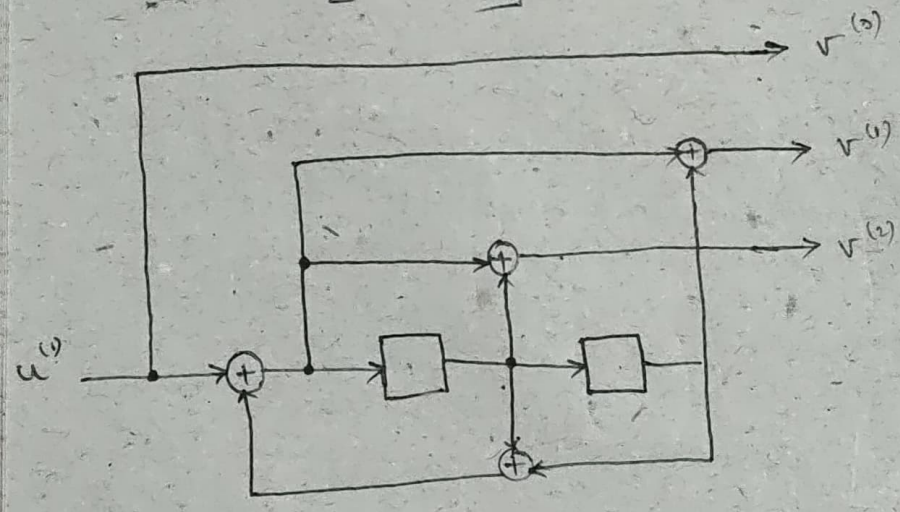
$$\Rightarrow (1+D)\,x(D) = u(D)$$

$$\Rightarrow x(D) = \frac{1}{1+D}\,u(D)$$

Therefore,

$$V^{(2)}(D) = \frac{1}{1+D}\,u(D)$$

$$\therefore G_1(D) = \left[1 \quad \frac{1}{1+D}\right] \longrightarrow \text{Rational}$$



$$G_1(D) = \left[1 \quad \frac{1+D^2}{1+D+D^2} \quad \frac{1+D}{1+D+D^2}\right]$$

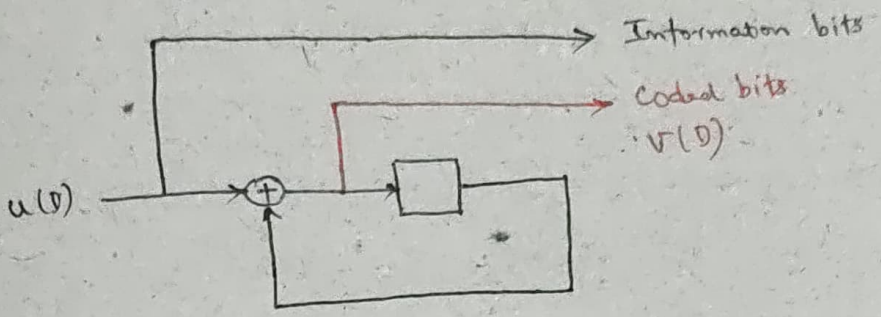
Fig: Another example for Feedback encoder

③ SYSTEMATIC ENCODER

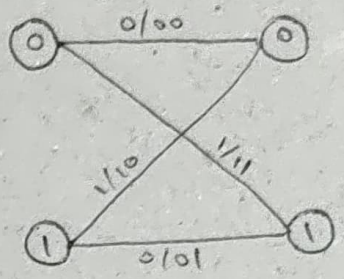
⊙ A rate $R = \frac{k}{n}$ convolutional encoder whose k information sequences appear unchanged among the n code sequences is called a Systematic encoder, and its generator matrix is called a Systematic Generator Matrix.

$G = [P : I]$ (or) $G = [I : P]$.

⊙ In figure, a systematic rate $R = \frac{1}{2}$ feedback convolutional encoder is shown.



(a) Systematic $R = \frac{1}{2}$ Feedback convolutional encoder



(b) State diagram

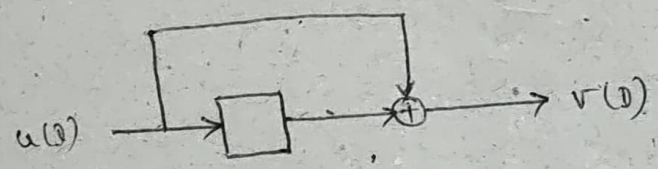
$$G(D) = \left[1 \quad \frac{1}{1+D}\right]$$

unchanged

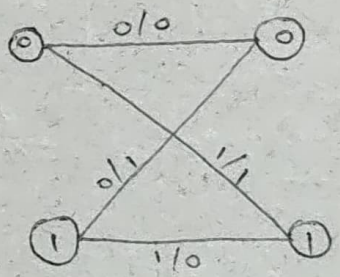
④ NON SYSTEMATIC ENCODER

⊙ In a non systematic convolutional encoder, the k information sequences do not appear unchanged in the n code sequences.

⊙ In figure, a non systematic rate $R = \frac{1}{2}$ feed forward convolutional encoder is shown.



(a) $R = \frac{1}{2}$ Feed forward convolutional Encoder



(b) State diagram

(5) EQUIVALENT ENCODER

⊙ Two convolutional generator matrices $G(D)$ and $G'(D)$ are equivalent if they encode the same code.

Codeword generated by $G(D)$

$$V(D) = u(D) \cdot G(D)$$

Codeword generated by $G'(D)$

$$V'(D) = u'(D) \cdot G'(D)$$

Here $V(D) = V'(D)$ which implies $G(D)$ and $G'(D)$ are equivalent.

⊙ Two convolutional encoders are equivalent if their generator matrices are equivalent.

⊙ Two generator matrices $G(D)$ and $G'(D)$ are equivalent if and only if there exists a rational invertible matrix $T(D)$ such that

$$G'(D) = T(D) \cdot G(D).$$

Example:

The generator matrix $G(D) = \begin{bmatrix} 1 & \frac{1}{1+D} \end{bmatrix}$ and $G'(D) = \begin{bmatrix} 1+D & 1 \end{bmatrix}$ are equivalent.

$$\Rightarrow T(D) = 1+D.$$

⊙ Consider the non-systematic encoder, $R = \frac{2}{3}$

$$G(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}.$$

Step 1: Row 1 $\Rightarrow \left(\frac{1}{1+D}\right)$ Row 1.

$$G(D) = \begin{bmatrix} 1 & \frac{D}{1+D} & 1 \\ D & 1 & 1 \end{bmatrix}$$

Step 2: Row 2 $\Rightarrow$ Row 2 + (D) Row 1

$$G(D) = \begin{bmatrix} 1 & \frac{D}{1+D} & 1 \\ D+D(1) & 1+D\left(\frac{D}{1+D}\right) & 1+D(1) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \frac{D}{1+D} & 1 \\ 0 & \frac{1+D+D^2}{1+D} & 1+D \end{bmatrix}$$

Step 3 : Row 2 $\Rightarrow \left(\frac{1+D}{1+D+D^2}\right)$ Row 2

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

$(1+D)^2$
$= 1 + D^2 + 2D$
$= 1 + D^2 + D + D$
$= 1 + D^2 + 0$
$= 1 + D^2$.

$$= \begin{bmatrix} 1 & \frac{D}{1+D} & 1 \\ 0 & 1 & \frac{1+D^2}{1+D+D^2} \end{bmatrix}$$

Step 4 : Row 1 $\Rightarrow$ Row 1 + $\left(\frac{D}{1+D}\right)$ Row 2

$$G(D) = \begin{bmatrix} 1+\frac{D}{1+D}(0) & \frac{D}{1+D} + \frac{D}{1+D}(1) & 1+\frac{D}{1+D}\left(\frac{1+D^2}{1+D+D^2}\right) \\ 0 & 1 & \frac{1+D^2}{1+D+D^2} \end{bmatrix}$$

$1+\frac{D}{1+D}\left(\frac{1+D^2}{1+D+D^2}\right)$

$= 1+\frac{D}{1+D}\left(\frac{(1+D)^2}{1+D+D^2}\right)$

$= 1+\frac{D+D^2}{1+D+D^2}$

$= \frac{1+D+D^2+D+D^2}{1+D+D^2}$

$= \frac{1}{1+D+D^2}$.

$$= \begin{bmatrix} 1 & 0 & \frac{1}{1+D+D^2} \\ 0 & 1 & \frac{1+D^2}{1+D+D^2} \end{bmatrix}$$

$(a+b)^2 = a^2+b^2+2ab$

$a^2+b^2 = (a+b)^2 - 2ab$

$1^2+D^2 = (1+D)^2 - 2(1)(D)$

$= (1+D)^2 - 2D$

$= (1+D)^2 - D - D$

$= (1+D)^2$

Therefore, the modified Systematic generator matrix

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

So, by doing elementary row operations, we are able to convert a non-systematic encoder into an equivalent Systematic encoder.

---

(6) CATASTROPHIC ENCODER

⊙ A convolutional encoder is Catastrophic if it encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols.

Clearly, a Systematic encoder cannot be a Catastrophic encoder because in Systematic encoder, the output codeword will atleast have the weight of the input codeword. So, catastrophic encoders are non-systematic encoders where the input has a much larger weight, but output has a finite weight.

⊙ This means that a finite number of channel errors may result in infinitely many errors in the receiver.

Example: $G(D) = \begin{bmatrix} 1+D & 1+D^2 \end{bmatrix}$

If the input sequence $u(D) = \begin{bmatrix} \frac{1}{1+D} \end{bmatrix} = 1 + D + D^2 + \dots$, then the output sequence $v(D) = u(D) \cdot G(D) = \begin{bmatrix} 1 & 1+D \end{bmatrix}$ has only weight 3, even though the information sequence has infinite weight.

Example: Rate $\frac{2}{3}$ Non systematic Feed forward Encoder.

⊙ The information sequence $u = (u_0, u_1, \ldots)$

$$= (u_0^{(1)} u_0^{(2)}, u_1^{(1)} u_1^{(2)}, \ldots)$$

can be written as two input sequences

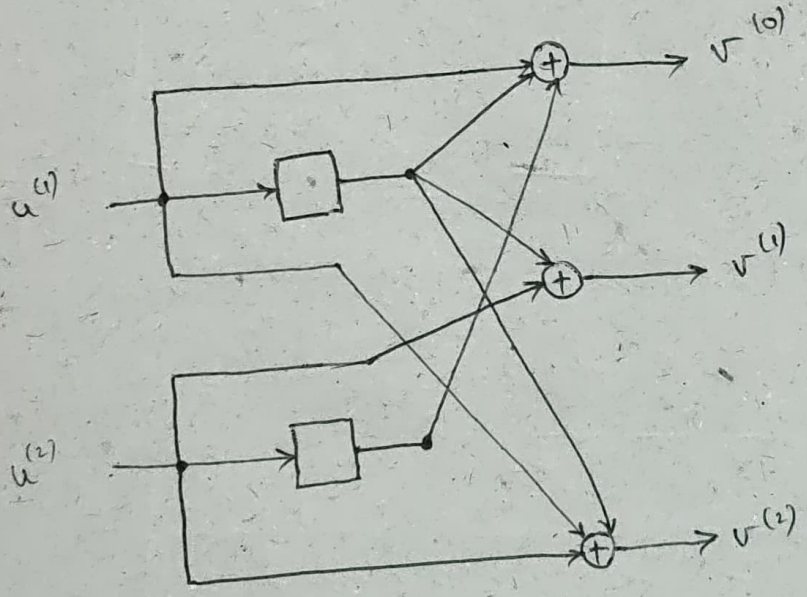
$$u^{(1)} = (u_0^{(1)}, u_1^{(1)}, u_2^{(1)}, \ldots) \text{ and}$$

$$u^{(2)} = (u_0^{(2)}, u_1^{(2)}, u_2^{(2)}, \ldots).$$

⊙ Let $g_i^{(j)} = (g_{i,0}^{(j)}, g_{i,1}^{(j)}, \ldots, g_{i,m}^{(j)})$ represent the generator sequence corresponding to input i and output j.

⊙ The generator matrix of a rate $R = \frac{2}{3}$ encoder is given by

$$G = \begin{bmatrix} g_{1,0}^{(0)} g_{1,0}^{(1)} g_{1,0}^{(2)} & g_{1,1}^{(0)} g_{1,1}^{(1)} g_{1,1}^{(2)} & \cdots & g_{1,m}^{(0)} g_{1,m}^{(1)} g_{1,m}^{(2)} & & \\ g_{2,0}^{(0)} g_{2,0}^{(1)} g_{2,0}^{(2)} & g_{2,1}^{(0)} g_{2,1}^{(1)} g_{2,1}^{(2)} & \cdots & g_{2,m}^{(0)} g_{2,m}^{(1)} g_{2,m}^{(2)} & & \\ & g_{1,0}^{(0)} g_{1,0}^{(1)} g_{1,0}^{(2)} & \cdots & g_{1,m-1}^{(0)} g_{1,m-1}^{(1)} g_{1,m-1}^{(2)} & g_{1,m}^{(0)} g_{1,m}^{(1)} g_{1,m}^{(2)} & \\ & g_{2,0}^{(0)} g_{2,0}^{(1)} g_{2,0}^{(2)} & \cdots & g_{2,m-1}^{(0)} g_{2,m-1}^{(1)} g_{2,m-1}^{(2)} & g_{2,m}^{(0)} g_{2,m}^{(1)} g_{2,m}^{(2)} & \\ & & & & & \ddots \end{bmatrix}$$



$k = 2, n = 3$

Fig: Memory $m = 1$, Constraint length $\nu = 2$.

⊙ The generator sequences for the encoder above are

$g_1^{(0)} = (1\ 1)$ $\quad g_1^{(1)} = (0\ 1)$ $\quad g_1^{(2)} = (1\ 1)$

$g_2^{(0)} = (0\ 1)$ $\quad g_2^{(1)} = (1\ 0)$ $\quad g_2^{(2)} = (1\ 0)$

⊙ The encoding equations can be written as

$$v^{(0)} = u^{(1)} * g_1^{(0)} + u^{(2)} * g_2^{(0)}$$

$$v^{(1)} = u^{(1)} * g_1^{(1)} + u^{(2)} * g_2^{(1)}$$

$$v^{(2)} = u^{(1)} * g_1^{(2)} + u^{(2)} * g_2^{(2)}$$

⊙ If $u^{(1)} = (1\ 0\ 1)$ and $u^{(2)} = (1\ 1\ 0)$, then

$v^{(0)} = (1\ 0\ 1) * (1\ 1) + (1\ 1\ 0) * (0\ 1) = (1\ 0\ 0\ 1)$

$v^{(2)} = (1\ 0\ 1) * (0\ 1) + (1\ 1\ 0) * (1\ 0) = (1\ 0\ 0\ 1)$

$v^{(2)} = (1\ 0\ 1) * (1\ 1) + (1\ 1\ 0) * (1\ 0) = (0\ 0\ 1\ 1)$

⊙ The output code sequence $(v = 110, 000, 001, 111)$.

$$v = uG = (1\ 1,\ 0\ 1,\ 1\ 0) \begin{bmatrix} 101 & 111 & & \\ 011 & 100 & & \\ & 101 & 111 & \\ & 011 & 100 & \\ & & 101 & 111 \\ & & 011 & 100 \end{bmatrix}$$

$$= (110,\ 000,\ 001,\ 111).$$

Week 4 : Session 4.

REALIZATION OF CONVOLUTIONAL ENCODER

In this section, we are going to talk about how we can realize a convolutional code using shift registers. We'll be talking about different realization of codes, which includes

- Controller Canonical form realization
- Observer canonical form realization
- Minimal encoder.

(1) Controller Canonical Form Realization

- In Controller Canonical Form realization, to realize a rate $R = \frac{k}{n}$ convolutional encoder, k shift register are used for input sequences, and n adders are used to form the output sequences.

  (a) No. of Shift Registers = No. of Input sequences.

- The k input sequences enter the shift registers at the left end of each shift register.
- The n adders used to obtain output sequences are external to the shift register.
- In the figure below, a rate $R=1$ non-systematic convolutional encoder with following generator function $G(D)$ is implemented in Controller canonical form realization.

$$G(D) = \left[\frac{f_0 + f_1 D + \cdots + f_{m-1} D^{m-1} + f_m D^m}{1 + g_1 D + g_2 D^2 + \cdots + g_m D^m} \right]$$

$(k=1, n=1)$

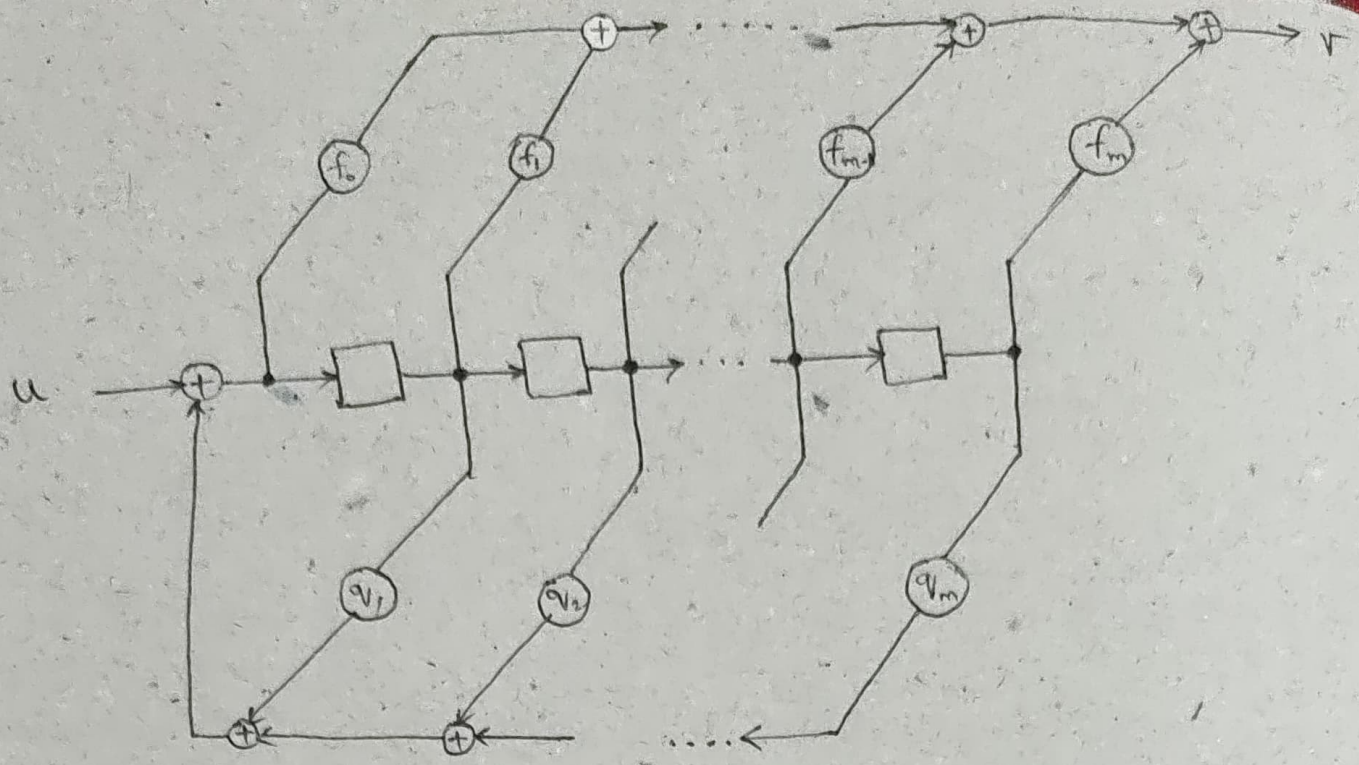


Fig: Controller Canonical Form Realization.

---

(2) Observer Canonical Form Realization

- ⊙ In Observer canonical form realization, to realize a Rate $R = \frac{k}{n}$ convolutional encoder, n shift register are used for output sequences.

  (ii) No. of Shift Registers = No. of coded sequence.

- ⊙ The k input sequences enter the adders internal to the shift registers.
- ⊙ The lowest degree term in the generator polynomial represent the connections to the right hand side of the shift registers.
- ⊙ In the figure below, a rate $R=1$, non systematic convolutional encoder with following generator function $G(D)$ is implemented in observer canonical form realization.

$$G_r(D) = \left[\frac{f_0 + f_1 D + \cdots + f_{m-1} D^{m-1} + f_m D^m}{1 + g_1 D + g_2 D^2 + \cdots + g_m D^m} \right]$$

$(k=1 \; ; \; n=1)$

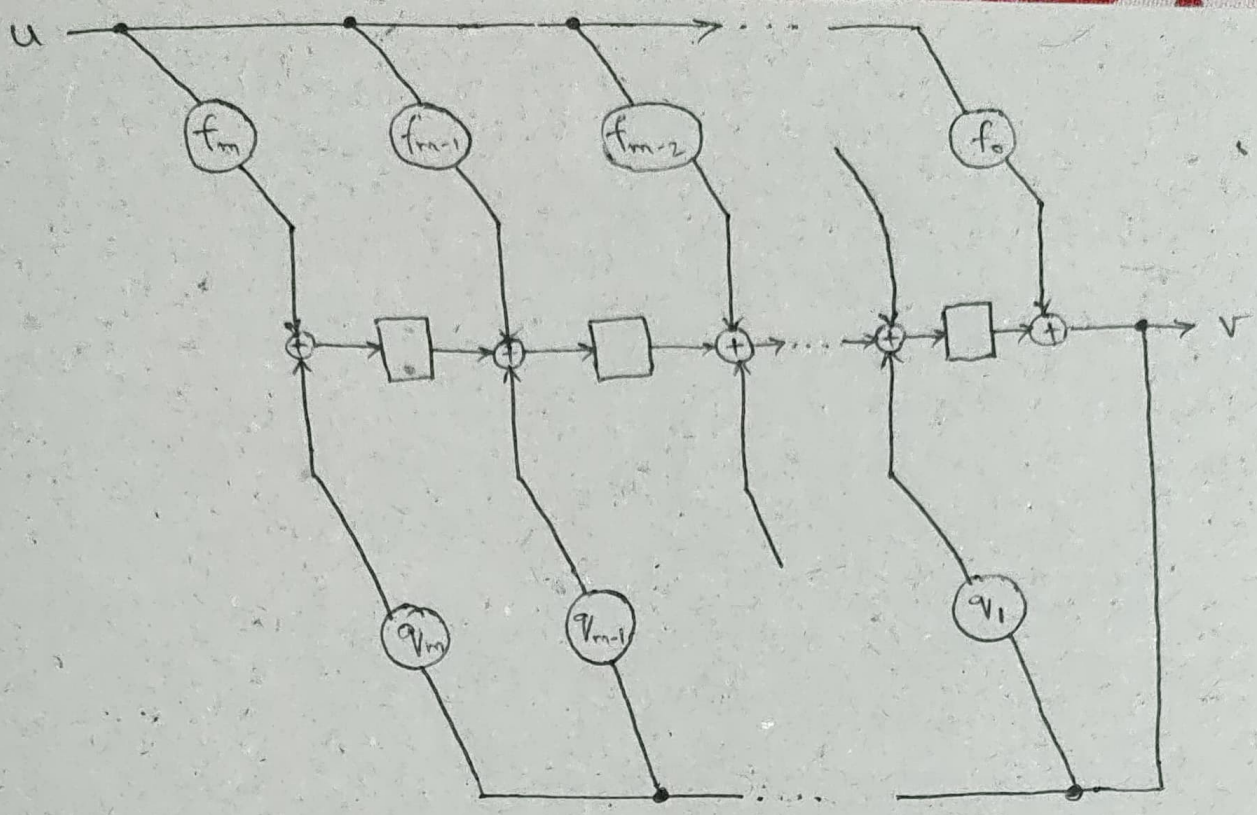


Fig: Observer Canonical Form Realization.

Now, what is interesting is, the same Generator Matrix can be implemented using different number of memory elements in one realization, compared to other. Lets illustrate this using an example.

We'll take a Generator Matrix, $G(D)$, and try to implement it using Observer Canonical form and Controller Canonical form. realization. And then we can see that, we require different number of memory elements in both the realization.

Example:

Let us consider a rate $R = \frac{2}{3}$ systematic feed forward encoder with generator matrix

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

WKT, $G = [I : P] \Rightarrow H = [P^T : I]$

The parity check matrix can be written as

$$H(D) = \begin{bmatrix} h^{(0)}(D) & h^{(1)}(D) & 1 \end{bmatrix} = \begin{bmatrix} 1+D+D^2 & 1+D & 1 \end{bmatrix}$$

⊙ The controller canonical form realization results in $(3, 2, 3)$ encoder. (ii) 3 memory elements.

⊙ The observer canonical form realization results in $(3, 2, 2)$ encoder. (ii) 2 memory elements.

(i) Controller Canonical form realization

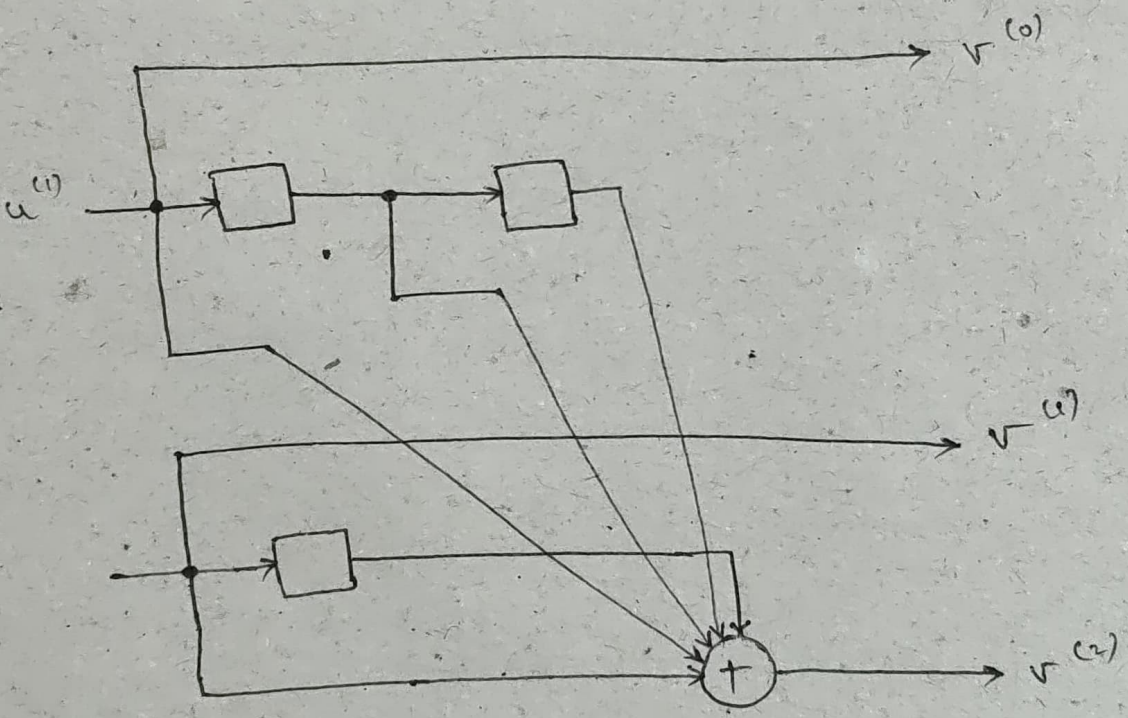
- No. of shift registers = No. of inputs.

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

Max. degree = 2.
∴ No. of Memory elements reqd = 2

Max. degree = 1
∴ No. of Memory Elements reqd. = 1

- There are 2 inputs.
So, we use 1 shift register to implement each input.



(ii) Observer canonical form realization.

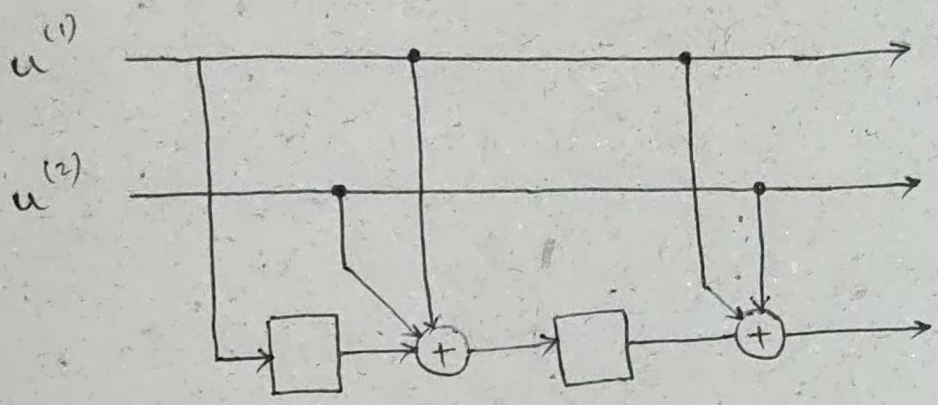
- No. of shift registers = No. of outputs.

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

→ Max. degree = 2
∴ No. of memory elements reqd. = 2

Max. degree = 0
∴ No. of memory elements reqd = 0.

- There are 3 outputs.
So, we use 1 shift register to implement each output.



---

(3) Minimal Encoder

⊙ A generator matrix of a convolutional code is minimal if its number of states is minimal over all equivalent generator matrices.

Example: R = 1 code

$$G(D) = \left[\frac{1+D^2}{1+D+D^2} \right] \longrightarrow 2^2 = 4 \text{ states}$$

and $$G'(D) = \left[\frac{(1+D^2)(1+D)}{(1+D+D^2)(1+D)} \right] \longrightarrow 2^3 = 8 \text{ states}$$

We can see that, both $G(D)$ and $G'(D)$ are same.
(i) Encode the same code, but the memory required for $G(D)$ is 2 and the memory required for $G'(D)$ is 3.
So, clearly $G(D)$ requires less number of states.

So, among all equivalent generator matrix, we say a Generator matrix is minimal if that Generator matrix requires least number of memory elements / states to represent it.

⊙ A minimal encoder is a realization of a minimal encoding matrix $G(D)$ with the minimal number of memory elements over all realizations of $G(D)$
