7th Semester (COURSE CODE – BEC1705)

INFORMATION THEORY & CODING (3-1-0)

Module –I (8Hours)

Information Theory and Source Coding

Introduction to Information Theory, Uncertainty and Information, Average Mutual Information and Entropy, Information

Measures for Continuous Random Variables, waveform sources

Amplitude Quantizing: quantizing noise, uniform quantizing, non-uniform quantizing

Differential Pulse Code Modulation: one-tap prediction, N-tap prediction, delta modulation, sigma delta modulation,

sigma delta A-to-D convertor(ADC), sigma delta D-to-A convertor(DAC)

Block coding: vector quantizing, Transform Coding: quantization for transform coding, Sub-band Coding

Source Coding for Digital Data: properties of codes, Huffman codes, Run-length codes

Module –II (12Hours)

Waveform coding: Antipodal and Orthogonal signals, Orthogonal and Biorthogonal codes, waveform coding system

example, Types of error control: Terminal connectivity, automatic repeat request

Structured Sequence: Channel models, Channel capacity, Channel coding, Information Capacity Theorem, The Shannon

Limit, Introduction to Error correcting codes, code rate & redundancy, parity check codes: Single parity check code,

Rectangular code

Linear Block codes: vector spaces, vector subspaces, A(6,3) linear block code example, Generator matrix, systematic

linear block codes, parity-check matrix, syndrome testing, error correction , Decoder implementation

Error Detecting & Correcting Capability: weight & distance of binary vectors, minimum distance of linear code, error

detection & correction, visualization of a 6-tuple space, erasure correction

Usefulness of Standard Array: estimating code capability, an (n, k) example, designing the (8,2) code, error detection vs.

error correction trade-off

Cyclic Codes: algebraic structures of cyclic code, binary cyclic code properties, encoding in systematic form, circuit for

dividing polynomial, systematic encoding with an (n-k)-stage shift register, error detection with an (n-k)-shift register

Well-Known Block Codes: Hamming codes, extended Golay code, BCH codes.

Module –III (12Hours)

Convolutional Encoding, Convolutional Encoder Representation: connection representation, state representation & the

state diagram, the tree diagram, the trellis diagram

Formulation of the Convolutional Decoding Problem: maximum likelihood decoding, channel models: hard versus soft

decisions, Viterbi Convolutional Decoding Algorithm, an example of viterbi convolutional decoding, decoder

implementation, path memory and synchronization

Properties of Convolutional Codes: distance properties of convolutional codes, systematic & non-systematic

convolutional codes, catastrophic error propagation in convolutional codes, performance bounds for convolutional codes,

coding gain, based known convolutional codes, convolutional code rate trade-off, soft-decision viterbi decoding

Other Convolutional Decoding Algorithms: sequential decoding, comparisons & limitations of viterbi & sequential

decoding, feedback decoding.

Module –IV (8Hours)

Reed-Solomon Codes: Reed-Solomon Error Probability, Why R-S codes perform well against burst noise, R-S
performance as a function of size, redundancy, and code rate
Interleaving & Concatenated Codes: Block interleaving, Convolutional interleaving, concatenated codes
Coding & Interleaving Applied to CD Digital Audio System: CIRC encodings, CIRC decoding, interpolation & muting
Turbo Codes: turbo code concepts, log-likelihood algebra

ACKNOWLEDGMENT

Different sources which are used in preparation of this manuscript are:

1. Digital Communications - Fundamentals and Applications - Bernard Sklar, 2nd Edition of Person Education Publication
2. Information Theory, Coding & Cryptography - Ranjan Bose, TMH Publication
3. Digital Communications ï Simon Haykin, Wiley Edition
4. Digital Communications - J.G.Proakis, 3rd Edition, Mc Graw Hill Publications.

MODULE:1

4.1 Source coding

Source coding deals with the task of forming efficient description, of information sources. Efficient descriptions permit a reduction in the memory or bandwidth resources required to store or to transport sample realizations of the source data. For discrete sources, the ability to form reduced data-rate descriptions is related to .their information content and the statistical correlation among the source symbols. For analog sources, the ability to form reduced data rate descriptions, subject to a stated fidelity criterion is related to the amplitude distribution and the temporal correlation of the source waveform. The goal of source coding is to form good fidelity description of the source for a given available bit rate, or to permit low bit rates to obtain a specified fidelity description of the source. To understand where the tools and techniques of source coding are effective, it is important to have Coliimon measures of source parameters. For this reason, in this section, we examine simple models of discrete and analog sources. and then we describe how source coding can be applied to these models.

4.2 Discrete Sources

A discrete source generates (or emits) a sequence of symbols X(k), selected from a source alphabet at discrete time intervals kT, where k = 1, 2, ... is a counting index. If the alphabet contains a finite number of symbols, say N symbols, the source is said to be a finite discrete source. An example of such a source is the strategy to initialize the game of Hangman. (In this game, a player must guess the letters, but not the positions. of a hidden word of known length. Penalties accurate to false guesses, and the letters of the entire word must be determined prior to the occurrence of six false guesses.) A discrete source is said to be memoryless if the symbols emitted by the source are statistically independent. In particular, this means that for the symbols taken two at a time. the joint probability of the two elements is simply the product

of their respective probabilities: P(X,, X,) = P(X,I X,) P(X,) = P(X,) P(X,)

A result of statistical independence is that the information required to transmit a sequence of M symbols (called an M-tuple) from a given alphabet is precisely **M** times the average information required to transmit a single symbol. This happens because the probability of a statistically independent M-tuple is given by

$$P(X_1, X_2, \ldots, X_M) = \prod_{m=1}^{M} P(X_m) \qquad (13.6)$$

so that the average entropy per symbol of a statistically independent M-tuple is

$$\begin{aligned} H_M(X) &= \frac{1}{M} \mathbf{E}\{-\log_2 P(X_1, X_2, \ldots, X_M)\} \\ &= \frac{1}{M} \sum_{X_m} [-P(X_m) \log_2 P(X_m)] \qquad (13.7) \\ &= H(X) \end{aligned}$$

A discrete source is said to have memory if the source element; composing the sequence are not independent. The dependency between symbols means that in a sequence of M symbols. there is reduced uncertainty about the M-th symbol when we know the previous (M-1) symbols. For instance, is there much uncertainty about the next symbol for the 10-tuple CALIFORNI-? The M-tuple with dependent symbols contains less information, or resolves less uncertainty, than does one with independent symbols. The entropy of a source with memory is the limit

$$H(X) = \lim_{M \to \infty} H_M(X) \qquad (13.8)$$

We observe that the entropy of an M-tuple from, a source with memory is always less than the entropy of a source with the same alphabet and symbol probability but without memory

$$H_M(X)_{\text{memory}} < H_M(X)_{\text{no memory}} \qquad (13.9)$$

For example, given a symbol (or letter) '-q" in English text, we know that the next
symbol will probably be a "u'.. Hence. in a communication task, being told that the
letter "u" follows a letter 'q' adds little information to our knowledge of the word being transmitted. As another example. given the letters 'th' the most likely syrnbol to follow is one of the following: a, e**.** i, o, **u,** r. and, space. Thus, adding the nest symbol to the given set resolves some uncertainty, but not much. A formal statement For example, given a symbol (or letter) '-q" in English text, we know that the nest symbol will probably be a "u'.. Hence. in a communication task, being told that the letter "u" follows a letter 'q' adds little information to our knowledge of the word being transmitted. As another example. given the letters 'th' the most likely symbol.-: to follow is one of the following: a,e i, o, **u,** r. and, space. Thus, adding the nest
symbol to the given set resolves some uncertainty, but not much. A formal statement. Statement of this awareness is that the average entropy per symbol of an M-tuple from a source with memory decreases as the length M increases. A consequence is that it is more efficient to encode symbols from a source with memory in groups of several symbols rather than to encode them one symbol at a time. For purposes of source encoding, encoder complexity, memory constraints, and delay considerations limit the sii:, of symbol sequences treated as a group.

;j ***J*** help us understand the gains to be had in coding sources with memory. We Form simple models of these sources. One such model is called a first-order- ***Markov source*** [I]. This model identifies a number of states (or symbols in the contest of information theory) and the conditional probabilities of transitioning to the next state. In the first-order model, the transition probabilities depend only. on the present state. This is. P(Xi+ ,IX,. &- ,, ...) = P(Xi+t|Xi). The model's memory does not extend beyond the present state. In the context of a binary sequence, this expression gives the probability of the next bit conditioned on the value of the current bit.

Example 13.2 Entropy of a Binary Source With Memory

Consider the binary (i.e., two symbol) first-order Markov source described by the state transition diagram shown in Figure 13.1. The source is defined by the state transition probabilities P(0|1) and P(1|0) of 0.45 and 0.05, respectively. The entropy of the source X is the weighted sum of the conditional entropies that correspond to the transition probabilities of the model. That is,

$$H(X) = P(0)H(X|0) + P(1)H(X|1) \tag{13.10}$$

where

$$H(X|0) = -[P(0|0)\log_2 P(0|0) + P(1|0)\log_2 P(1|0)]$$

and

$$H(X|1) = -[P(0|1)\log_2 P(0|1) + P(1|1)\log_2 P(1|1)]$$

The a priori probability of each state is found by the total probability equations

$$P(0) = P(0|0)P(0) + P(0|1)P(1)$$
$$P(1) = P(1|0)P(0) + P(1|1)P(1)$$
$$P(0) + P(1) = 1$$

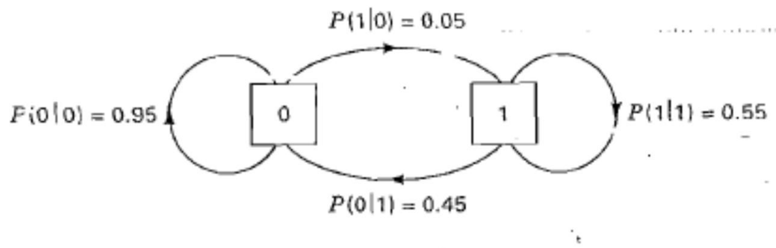


Figure 13.1 State transition diagram for first-order Markov model.

Solving for the a priori probabilities using the transition probabilities, we !lave P(0) = 0.9 and P(1) = 0.1 Solving for the source entropy using Equation (13.10). we have

$$\begin{aligned} H(X) &= P(0)H(X|0) + P(1)H(X|1) \\ &= (0.9)(0.286) + (0.1)(0.993) \\ &= 0.357 \text{ bit/symbol} \end{aligned}$$

Comparing this result with the result of Example 13.1, are see that the source n.ith memory has lower ***entropy*** than the source without memory, even though the a priori symbol probabilities are the same.

4.3 Waveform Sources

A waveform source is a random process **of** some random variable. We classically consider this random variable to be time. so that the waveform of interest is a time varying waveform. Important examples of time-varying waveforms a r e the outputs of transducers used in process control, such as temperature, pressure, velocity, and flow rates. Examples of particularly high interest include speech and music. The waveform can also be a function of one or more spatial variables (e.g., displacement in **r** a n d y). Important examples of spatial waveforms include single images, such as a photograph, or moving images, such as successive images (at 24-frames1 sec) of moving picture film. Spatial waveforms are often converted to time-varying waveforms by a scanning operation. This is done, for example, for facsimile and

Joint Photographic Expert Group (JPEG) transmission, as well as for standard broadcast television transmission.

4.3.1 Amplitude Density Functions

Discrete sources were described by their list of possible elements (called letters of an alphabet) and their multidimensional probability density functions (pdf) of all orders By analogy: waveform sources are similarly described in terms of their probability. density functions as well as by parameters and functions derived from these functions. we model many waveforms as random processes with classical probability density functions and with simple correlation properties. In the modeling process, we distinguish between short-term or local (time) characteristics and long-term or global characteristics. This partition is necessary because many

Waveforms are non stationary. The probability. density function of the actual process may not be available to the system designer. Sample density functions can of course be rapidly formed in

real time during a short preliminary interval and used as reasonable estimates over the subsequent interval. A less ambitious task is simply to form short-term wave to related averages. 'l'hese include Lime sample mean (or time average value), the sample variance (or mean-square value of the zero mean process), and the sample correlation coefficients formed over the prior sample interval. In many applications of waveform analysis, the input waveform is converted into a zero mean process by subtracting an estimate, of its mean value. For instance, this happens, in comparators used in analog-to-digital converters for which auxiliary circuitry measures the internal dc-offset voltages and subtracts them in a process known as *autozero.* Further, the variance estimate is often used to scale the input waveform to match the dynamic amplitude range of subsequent waveform conditioning circuitry. This process, performed in a data collection process, is called ***autoranging*** or ***automatic gain control*** (AGC).

The function of these signal conditioning operations. mean removal. and variance control or gain adjustment (shown in Figure 13.2) is to formalize probability density functions of the input waveforms. This normalization assures optimal utility of the limited dynamic range

of subsequent recording transmission .or processing subsystems. Many waveform sources exhibit significant amplitude correlation in successive time intervals. T. this correlation means that signal levels in successive time intervals are not independent. If the time signal is independent over successive intervals, the autocorrelation function would be an impulse function. Many signals of engineering interest have finite width correlation functions. The effective width of the correlation function (in seconds) is called the correlation time of the process and is akin to the time constant of a low-pass filter. This time interval is an indication of how much shift along the time axis is required to de correlate the data. If the correlation time is large we interpret this to mean that the waveform makes significant amplitude changes slowly. Conversely, if the correlation time is small, we infer that the waveform makes significant changes in amplitude very quickly.

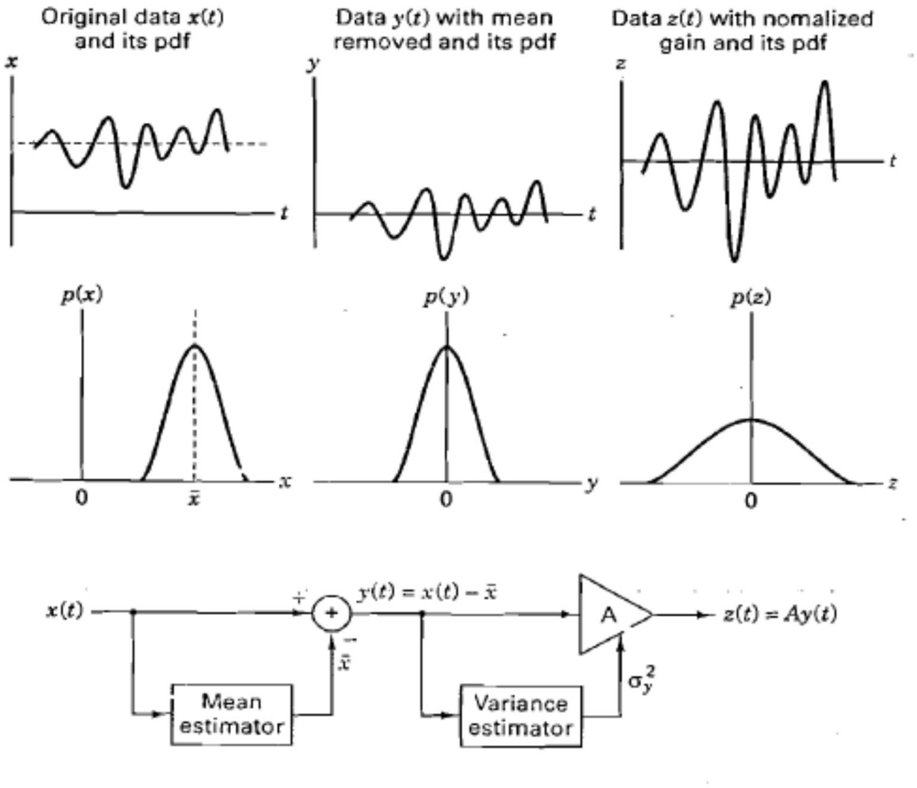


Figure 13.2 Mean removal and variance normalization (gain adjustment) for a data dependent signal conditioning system.

4.4 AMPLITUDE QUANTIZING

Amplitude quantizing is the task of mapping samples of a continuous amplitude waveform to a finite set of amplitudes. The hardware that performs the mapping is the analog-to-digital converter (ADC or A-to-D). The amplitude quantizing occurs after the sample-and-hold operation. The simplest quantizer to visualize performs an instantaneous mapping from each continuous input sample level to one of the pre assigned equally spaced output levels.

Quantizers that exhibit equally spaced increments between possible quantized output levels are called ***uniform quantizers*** or ***linear quantizers.*** Possible instantaneous input-output characteristics are easily visualized by a simple staircase graph consisting of risers and treads of the types shown in Figure 13.3. Figure 13.3a, b, and d show quantizers with uniform quantizing steps, while Figure 13.3 c is a quantizer with non uniform quantizing steps. Figure lj.3a depicts a quantizer with midtread at the origin, while Figure 13.3b and d

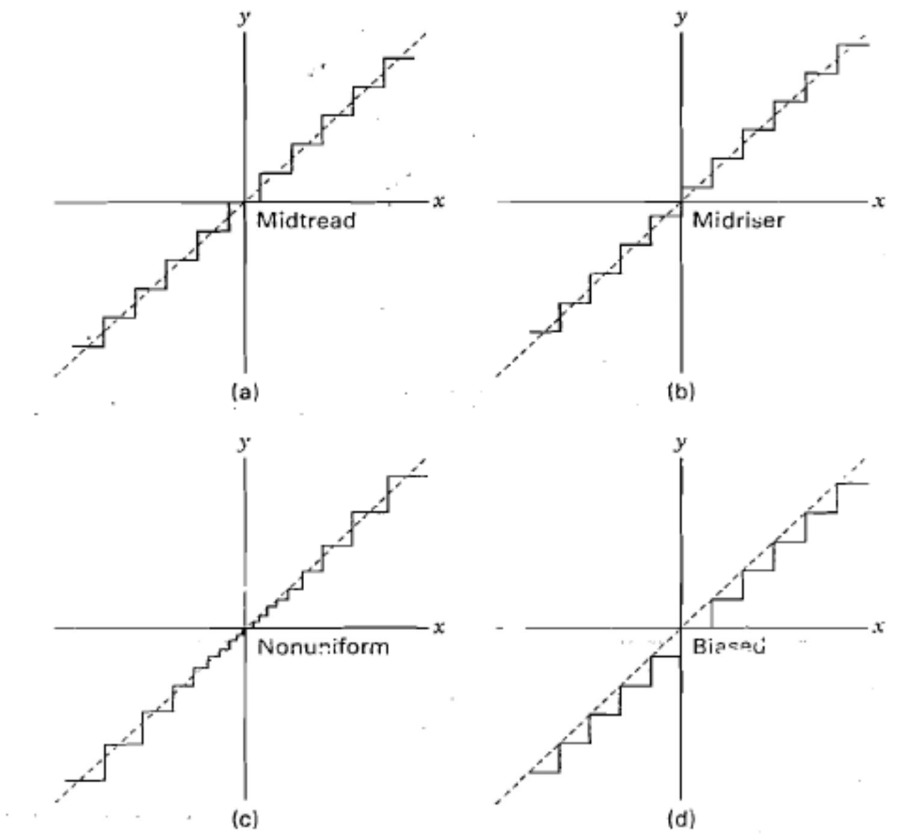


Figure 13.3 Various quantizer transfer functions.

present quantizers with midrisers at the origin. A distinguishing property of midriser and midtread converters is related to the presence or absence, respectively, of output level changes when the input to the converter is low-level idle noise. Further, Figure 13.3d presents a biased (i.e., truncation) quantizer, while the remaining quantizers in the figure are unbiased and are referred to as rounding quantizers. Such unbiased quantizers represent ideal models, but rounding is never implemented in A/D converters. Quantizers are typically implemented as truncation quantizers. terms '-midtread" a-d "midriser" are staircase terms used to describe whether the horizontal or vertical member of the staircase is at the origin. The unity-slope dashed line passing through the origin represents the ideal non quantized input-output characteristic we are trying to approximate with the staircase. The difference between the staircase and the unity-slope line segment

represents the approximation error made by the quantizer at each input level. Figure 13.4 illustrates the approximation error amplitude versus input amplitude function for each quantizer characteristic in Figure 13.3. Parts (a) through (d) of Figure 13.4 correspond to the same parts in Figure 13.3. This error is often modeled as quantizing noise because the error sequence obtained when quantizing a wideband random process is reminiscent of an additive noise sequence. Unlike true additive noise sources, however, the quantizing errors are sienal dependent and are highly structured. It is desirable to break up this structure, which can be accomplished by

introducing an independent noise purturbation, known as dither, prior to the quantization step.

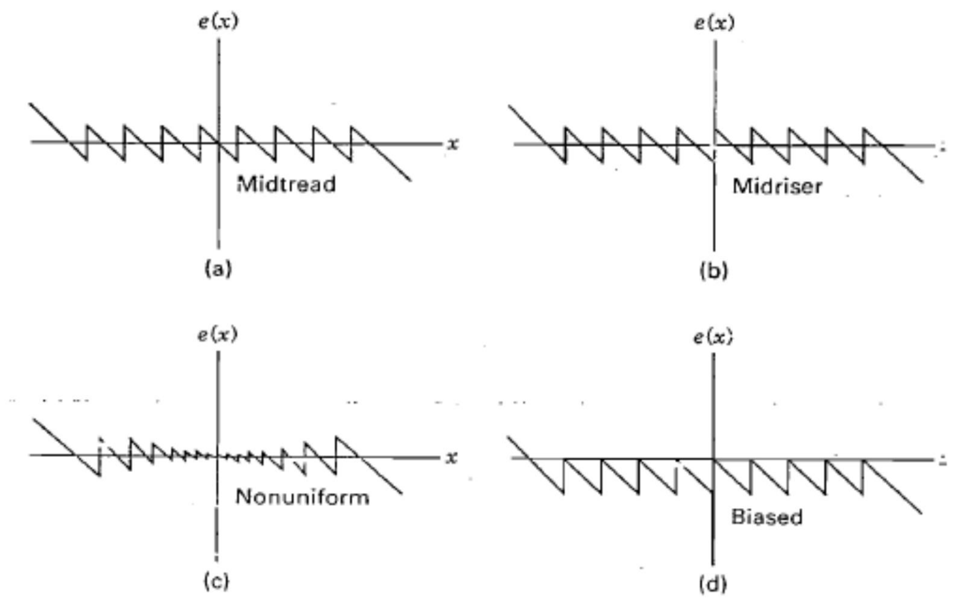


Figure 13.4 Instantaneous error for various quantizer transfer functions.

The linear quantizer is simple to implement and is particularly easy to understand. It is the universal form of the quantizer, in the sense that it makes no assumptions about the amplitude statistics and correlation properties of the input waveform, nor does it take advantage of user-related fidelity specifications. Quantizers that take advantage of these considerations are more efficient as source coders and are more task specific then the general linear quantizer; these quantizers are often more complex and more expensive, but they are justified in terms or improved system performance. There are applications for which the uniform quantizer is the most desirable amplitude quantizer. These include signal processing applications, graphics and display applications, and process control applications. There are other applications for which nonuniform adaptive quantizers are more desirable amplitude quantizers. These include waveform encoders for efficient storage and communication, contour encoders for images. vector encoders for speech. and analysis synthesis encoders (such as the vocoder) for speech.

4.4.1 Quantizing Noise

The difference between the input and output of a quantizer is called the quantizing error. In Figure 13.5 we demonstrate the process of mapping the input sequence s(t) to the quantized output sequence .i-(I). We can visualize forming.it by adding to each ***x(t)*** an error sequence, e(t):

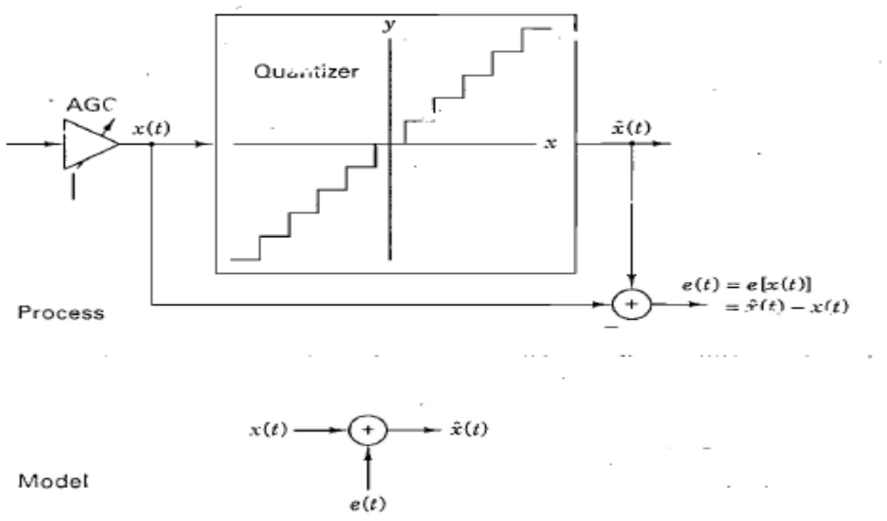


Figure 13.5 Process and model of quantizing noise corruption of input signal.

$$\hat{x}(t) = x(t) + e(t)$$

The error sequence e(t) is deterministically defined by the input amplitude through the instantaneous error versus amplitude characteristic of the form in Figure 133. We note that the error sequence exhibits two distinct characteristics over different input operating regions. The first operating interval is the granular error region corresponding to the input sawtooth-showed error characteristic. Within this interval, the quantizer errors are confined by the size of the nearby staircase risers. The errors that occur in this region are called the granular errors, or sometimes the quantizing errors. The input interval for which the quantizing errors are granular defines the dynamic range of the quantizer. This interval is sometimes called the region of "'tear operation. Proper use of the quantizer requires that the input signal conditioning match the dynamic range of the input signal to the dynamic range of the quantizer. This is the function of the signal-dependent gain control system, called automatic gain control(AGC), indicated in the signal flow path of Figure 13.5. The second operating interval is the nongranular error region corresponding to the linearly increasing (or decreasing) error characteristic. The errors that occur

in this interval are called saturation or overload errors. When the quantizer operates in this region, we say that the quantizer is saturated Saturation errors are larger than the granular errors and may have a more objectionable effect on reconstruction fidelity. The quantization error corresponding to each value of input amplitude represents an error or noise term associated with that input amplitude. Under tile assumptions that the quantization interval is small compared with the dynamic range of the input signal, and that the input signal has a smooth probability density function over the quantization interval, we can assume that the quantization errors arc.

uniformly distributed over that interval, as illustrated in Figure 13.6. The pdf with zero mean corresponds to a rounding quantizer, while the pdf with a mean of -q/2 corresponds to a truncation quantizer. A quantizer or analog-to-digital converter (ADC) is defined by the number.

size. and location of its quantizing levels or step boundaries. and the corresponding step sizes. In a uniform quantizer., the step sizes are equal and are equally spaced. The number of levels N is typically a power of 2 of the form N =2^b, where b is the number of bits used in the conversion process. This number of levels is equally distributed over the dynamic range of the possible input levels. Normally, this range is defined $\pm E_{max}$,such as ± 1.0 V or ± 5.0 V. Thus, accounting for the full range of 2Emax. the size-of a quantization step is .

$$q = \frac{2E_{\max}}{2^{b}} \qquad (13.11)$$

As an example, using Equation (13.1 I), the quantizing step (here after called a quantile.) for a 10-bit converter operating over the +I .0 V range is 1.953 mV. Occasionally the operating range of inverter is altered so that the quantile is a "whole" number. For example, changing the operating range of the converter to f1.024 V results in a quantizing step size of *2.0* mV. A useful figure of merit for the

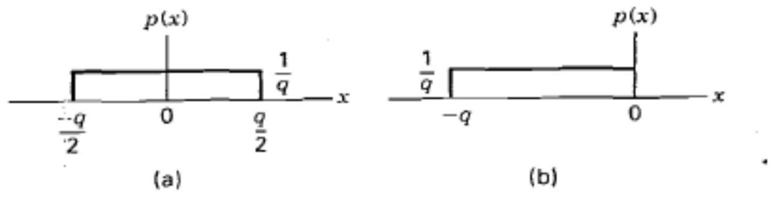


Figure 13.6 Probability density f··nctions for quantizing error uniformly distributed over one quantile, q. (a) Probability density function for a roundi·g quantizer. (b) Probability density function for a truncating quantizer.

uniform quantizer is the quantizer output variance. If we assume that the quantization error is uniformly distributed over a single quantile interval q-wide, the quantizer variance (which represents the quantizer noise or error power) for the zero-mean error is found to be I

$$\sigma^2 = \int_{-q/2}^{q/2} e^2 p(e)de = \int_{-q/2}^{q/2} e^2 \frac{1}{q} de = \frac{q^2}{12} \qquad (13.12)$$

where p(c) = 1/q, over an interval of width *q,* is the probability density function *7* (pdf) of the quantization error e. Thus, the rms quantizer noise in a quantile interval of width q is found to be q fi or 0.29q. Equation (13.12) determined the I quantizing noise pou.er over one quantile, assuming that the errors are equiprobable over. the quantization interval. If we include operation in the saturation interval of a quantizer, or if we include nonuniform quantizers, we find that the quanlization intervals are not of equal width ;b.cr the range of the input variable, and that

the amplitude density is not uniform over the quantization interval. We can account for this amplitude-dependent error power ***are*** by averaging the squared error over the amplitude variable weighting :,y the probability of that amplitude. This is expressed by

$$\sigma_q^2 = \mathbf{E}\{[x - q(x)]^2\} = \int_{-\infty}^{\infty} e^2(x)p(x)dx \qquad (13.13)$$

where **s** is the input variable, q(x) is its quantized version, ***e(x)*** = x - ***q(x)*** is the error. and y(.v) is tile pdf OI the amplitude **;:x.** We can partition the interval of integration in Equation (13:13) into two main intervals-one accounting for errors in the staircase or linear region of the quantizer, and the second accounting Lor errors in the saturation region. We define the saturation ampiitude of the quantizer as *Emax.* Also. we assume an odd symmetric transfer function FG, the quantizer, and a symmetric pdf for the input signal. The error power defined in Equation

(13.13). is the total error power, which can be partitioned 2s

$$\sigma_q^2 = 2\int_0^\infty e^2(x)p(x)dx \qquad (13.14a)$$

$$= 2\int_0^{E_{\max}} e^2(x)p(x)dx + 2\int_{E_{\max}}^\infty e^2(x)p(x)dx \qquad (13.14b)$$
$$= \sigma_{\text{Lin}}^2 + \sigma_{\text{Sat}}^2$$

where **a:;,** is the error power in the linear region and **a$,,** is the error power in the saturation region. The error power **a&,** can be further divided into subintervals corresponding to the :I received discrete quantizer output levels (i.e., quantiles). If we assume that there are N such quantile levels, the integral becomes

$$\sigma_{\text{Lin}}^2 = 2\sum_{n=0}^{N/2-1} \int_{x_n}^{x_{n+1}} e^2(x)p(x)dx \qquad (13.15)$$

where x,, is a quantizer level and an interval or step size between two such levels is called a quantile interval. Recall that N is typically a power of 2. Thus, there are ***N/2*** - 1 positive levels, N/2 - 1 negative levels, and a zero level, making a total of N - 1 levels and N - 2 intervals. If we now approximate the density function by a constant qn = (xn+ -.r,), in each quantile interval, Equation (13.15) simplifies to

$$\sigma^2_{\text{Lin}} = 2 \sum_{n=0}^{N/2-1} \frac{x^3}{3} \Bigg|_{x=-q_n/2}^{x=+q_n/2} p(x_n)$$

$$= 2 \sum_{n=0}^{N/2-1} \frac{q_n^2}{12} \; p(x_n)q_n \qquad (13.16)$$

wl~cree cv) in Equation (13.15) has been replaced by **.v** in Equation (13.16), since **e(t)** 1s a linear function of **.t-** with unity slope and passes through zero at the midpoint of each interval. Also, the limits 01 integration in equation (13.15) have been replaced by the change in **x** over a quantile inter Val. Since the change has been denoted as q,. the lower and upper limits can be designated as ***x*** = -q,,/2 and *x* = *+q,,/2,* respectively. Equation (13.16) describes the error power In the linear region as a summation of error power q;J12 in each quantile interval weighted by :he probability p(xn)qn of that error PO\: :r.

4.4.2 Uniform Quantizing

If the quantizer has uniform quantiles equal to q and all intervals are equally likely,

Equation (13.16) simplifies further to

$$\sigma^2_{\text{Lin}} = \frac{2}{12} \sum_{n=0}^{N/2-1} q_n^2 \; p(x_n)q_n = \frac{2}{12} \sum_{n=0}^{N/2-1} q^2 \frac{1}{q(N-2)} q = \frac{q^2}{12} \qquad (13.17)$$

If the quantizer does not operate in the saturation region (the quantization noise power), then **a:** = u:,,; and these terms are often used interchangeably. Noise power alone will not fully describe the noise performance of the quantizer. A more - meaningful measure of quality is the ratio of the second central moment (variance) , of the quantizing noise and the input signal. Assuming that the input signal has zero mean, the signal variance **I;**

$$\sigma_x^2 = \int_{-\infty}^{\infty} x^2 p(x)dx \qquad (13.18)$$

4.4.3 Dithering .

Dithering is one of the cleverest applications of noise as a useful engineering tool.

A dither signal is a small perturbation or disturbance added to ,a measurement process to reduce the effect of small local nonlinearities. The most familiar form of dither is the slight tapping we apply to the side of a d'Arsonval meter movement prior to taking the reading (before the days of digital meters). The tapping is a sequence of little impulses for displacing the needle movement beyond the local region, which exhibits a nonlinear coefficient of friction at low velocities. A more sophisticated example of this same effect is the mechanical dither applied to the counter-rotating laser beams of a laser beam gyro to break up low-level frequency entrapment, known as a dead band [3]. In the analog-to-digital converter application, the effect of the dither is to reduce

or eliminate the local discontinuities (i.e., the risers and treads) of the instantaneous input-output transfer function. We can best visualize the effect of these discontinuities by listing the desired properties of the error sequence formed by the quantizer process and then examining the actual properties of the same sequence. The quantizer error sequence is modeled as additive noise. The desired properties of such a noise sequence e(tz) are as follows

1. Zero mean: $\mathbf{E}\{e(n)\} = 0$
2. White: $\mathbf{E}\{e(n)e(n+m)\} = \sigma^2 \delta(m)$
3. Uncorrelated with data $x(n)$: $\mathbf{E}\{e(n)\, x(n+m)\} = 0$

where ti and m are indices, and S(m) is a Dirac delta function. In Figure 13.10, we examine a sequence of samples formed by a truncating ADC and make the following observations: **1.** The error sequence is all of the same polarity; therefore, it is not zero near.

2. The error sequence is not independent, sample-to-sample; therefore, it is not white.

3. The error sequence is correlated with the input; therefore, it is not independent.

4.4.4 Non-uniform Quantizing

Uniform quantizers are the most common type of analog-to-digital converters because they are the most robust. By "robust" we mean that they are relatively insensitive to small changes in the input statistics. They achieve this robustness by not being finely tuned to one specific set of input parameters. This allows them to perform well even in the face of uncertain input parameters, and it means that small changes in input statistics will result in only small changes in output statistics. When there is small uncertainty in the input signal statistics, it is possible to design a nonuniform quantizer that exhibits a smaller quantizer NSR than a uniform

quantizer using the same number of bits. This is accomplished by partitioning the input dynamic range into nonuniform intervals such that the noise power, weighted by the probability of occurrence in each interval, is the same. Iterative solutions for the decision boundaries and step sizes for an optimal quantizer can be found for specific density functions aod for a small number of bits. This task is simplified by modeling the nonuniform quantizer as a sequence of operators, as depicted in Figure 13.12. The input signal is first mapped, via a nonlinear function called a compressor, to an alternative range of levels. These levels are uniformly quantized and the quantized signal levels are then mapped, via a complementary nonlinear function called an explorer; to the output range of levels. Borrowing part of the name from each of the operations Compress and expand, we form the acronym by which this process is commonly identified: companding.

4.4.5 (Near) Optimal Non-uniform Quantizing

Examining the compressor characteristics **j.** = *C(s)* of Figure 13.12 we note that the quantizing step sizes for the output variable). are related to the quantizer step sizes for the input variable **.v** through the slope C(s) [e.g.. ***Ay*** = A s C(x)]. For an ...-.r y pdf and arbitrary compressor characteristics. we can arrive at the output quantizing noise variance [7]

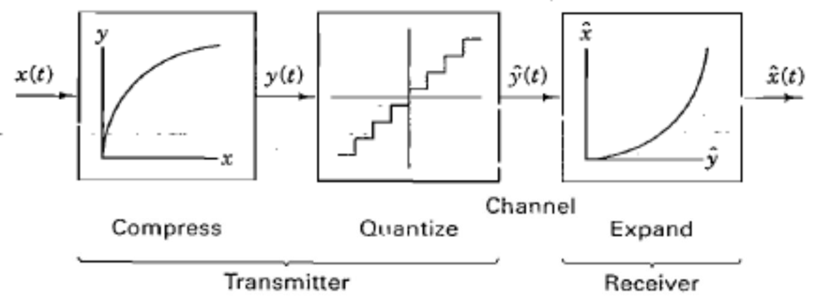


Figure 13.12 Nonuniform quantizer as a sequence of operators: compression, uniform quantization, and expansion.

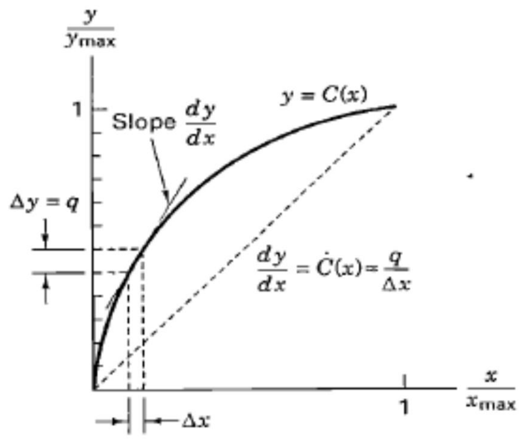


Figure 13.13 Compressor characteristics $C(x)$ and estimate to local slope $\dot{C}(x)$.

$$\sigma_q^2 = \frac{q^2}{12}\int_{-x_{max}}^{x_{max}} \frac{p(x)}{|\dot{C}(x)|^2}\,dx \tag{13.25}$$

for a specific pdf, the compression characteristics C(s) call be found which minimize **a:.** The optimal compressor law for a given pdf is

$$C(x) = \int_0^x \sqrt[3]{Kp(z)}\,dz \tag{13.26}$$

We find that thc optimal compressor characteristic is proportional to the integration of the cube root of the input probability density function. This is called ***fine tuning.*** If the compressor is designed to operate with one density function and it is used with some other density function (including scaled versions), the quantizer is said to be mismatched and there may be severe performance degradation due to the mismatch

4.4.6 **Logarithmic compression**

In thc preceding section, we presented the compression law for the case in which the input signal's pdf is well defined. We now address the case for which little is known about the signal's pdf. This case occurs. for instance, when the average power of the input signal is a random variable As ax-example, the voice level of a randomly choose;; telephone user may vary from one extreme of a barely audible to the other extreme of a bellowing shout. Fo, ;he case of an unknown pdf, the compressor characteristics of the nonuniforrn quantize1 must be selected such that the resultant noise performance is independent of the specific density function. Although this is a worthy undertaking, it may not be possible to achieve>\this independence. We are willing to compromise, however. and we will settle for' virtual independence over a large range of input variance and input density functions. An example of a quantizer that ,exhibits a

SNR independent of the input signal's pdf can be visualized with the aid of Figure **2.18.** There we can see a very large difference in NSR ratio for different amplitude input signals when quantized with a uniform quantizer. By comparison, we can see that the nonuniform quantizer only permits large errors for large signals. This makes intuitive sense. If the SNR is to be independent of the amplitude distribution, the quantizing noise must be proportional to the input level. Equation (13.25) resented the quantizer noise variance for an arbitrary pdf and arbitrary compressor characteristics. The signal variance for r/; pdf is

$$\sigma_x^2 = \int_{-\infty}^{\infty} x^2 p(x)\,dx \qquad (13.27)$$

In the absence of saturation, the quantizer SNR is of the form

$$\frac{\sigma_x^2}{\sigma_q^2} = \frac{\displaystyle\int_{-x_{\max}}^{x_{\max}} x^2 p(x)\,dx}{(q^2/12)\displaystyle\int_{-x_{\max}}^{x_{\max}} [p(x)/\dot{C}^2(x)]\,dx} \qquad (13.28)$$

To have the SNR be independent of the specific density function, we require that the numerator be a scaled version of the denominator. This happens if the following is true:

$$[\dot{C}(x)]^2 = \left(\frac{K}{x}\right)^2 \qquad (13.29)$$

or

$$\dot{C}(x) = \frac{K}{x} \qquad (13.30)$$

from which we obtain by integration,

$$C(x) = \int_0^x \frac{K}{z}\,dz \qquad (13.31)$$

or

$$C(x) = \log_e(x) + \text{constant} \qquad (13.32)$$

This result is intuitively appealing. A *logarithmic compressor* allows for a *constant S*QNR output, because with a logarithmic scale equal distances (or errors) are, infact , equal ratios, which is what we require in order for the SNR to remain fixed over the input signal range. In Equation (13.32), the constant is present to match the boundary conditions between **xmax** and ymax**.** Accounting for this boundary condition, we have' the logarithmic converter of the form

$$\frac{y}{y_{\max}} = \frac{C(x)}{y_{\max}} = \log_e\left(\frac{x}{x_{\max}}\right) \qquad (13.33)$$

The fo1.m of the compression suggested by the logarithm functions shown in Fig l: e 13.11a. The first difficulty with this function is that it does not map the negative input signals. We account for the negative signals by adding a reflected version of the log to **[he** negative axis. This modification results in Figure 13.14 yielding

$$\frac{y}{y_{\max}} = \log_e\left(\frac{|x|}{x_{\max}}\right)\operatorname{sgn}(x) \qquad (13.34)$$

where

$$\operatorname{sgn} x = \begin{cases} +1 \text{ for } x \geq 0 \\ -1 \text{ for } x < 0 \end{cases}$$

The remaining difficulty we face is that the compression described by Equation (13.34) is not continuous through the origin; in fact, if completely misses the origin. We need to make a smooth transition between the logarithmic function and a linear segment passing through the origin. There are two standard compression functions that perform this transition-the p-law and A-law companders.***p-Law. Compander.*** The p-law compander, introduced by the Bell System for use in South America, is of the form Figure

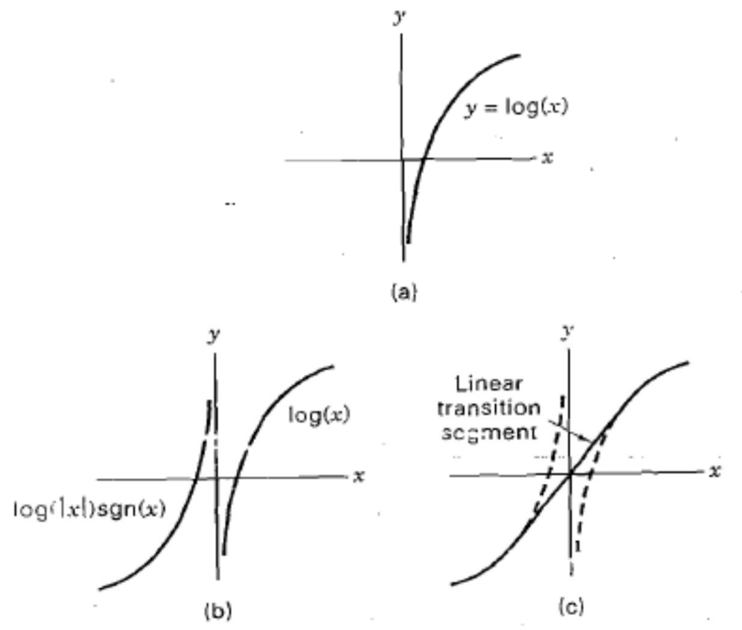


Figure 13.14 (a) Log function prototype for compression law. (b) Log | x | sgn x function prototype for compression law. (c) Log | x | sgn x function with a smooth transition between segments.

$$y = C(x) = y_{\max}\frac{\log_e[1 + \mu(|x|/x_{\max})]}{\log_e(1 + \mu)}\operatorname{sgn} x \qquad (13.35)$$

The approximate behavior of this compressor in the regions corresponding to small and large values of the argument are

$$y = C(x) = \begin{cases} y_{\max}\dfrac{\mu(|x|/x_{\max})}{\log_e(\mu)} & \text{for } \mu\dfrac{|x|}{x_{\max}} \ll 1 \\ y_{\max}\dfrac{\log_e[\mu(|x|/x_{\max})]}{\log_e(\mu)} & \text{for } \mu\dfrac{|x|}{x_{\max}} \gg 1 \end{cases} \qquad (13.36)$$

The parameter μ in the μ-law compander had originally been set to 100 for use with a 7-bit converter. It was later changed to 255 for use with an 8-bit converter. The 8-bit $\mu = 255$ μ-law converter has become the standard North American conversion law.

A-Law ***Compander.*** The A-law compander is the CCIR (hence the European) standard approximation to the logarithmic compression. The form of the compressor is

$$y = C(x) = \begin{cases} y_{\max} \dfrac{A(|x|/x_{\max})}{1 + \log_e (A)} \operatorname{sgn} x & \text{for} \quad 0 < \dfrac{|x|}{x_{\max}} < \dfrac{1}{A} \\ y_{\max} \dfrac{1 + \log_e[A(|x|/x_{\max})]}{1 + \log_e A} \operatorname{sgn} x & \text{for} \quad \dfrac{1}{A} < \dfrac{|x|}{x_{\max}} < 1 \end{cases} \qquad (13.43)$$

The standard value of the parameter A is 87.56, and for this value, using an S-bit converter, the SNR is 3S.0 dB. The A-law compression characteristic is approximated. in a manner similar to the p-IP compressor,b y a sequence of 16 lilLtar chords spanning the output range. The lower two chords in each quandrant are in fact a signal chord corresponding to the linear segment of the A-law compressor. One important difference between the A-law and rile p-law compression characteristics is that the A-law standard has a mid-riser at the origin, while the p-law standard

has a mid-tread at the origin. Thus, the A-law compressor has no zero value, and hence it exhibits no interval for which data are not king transmitted for zero input. There are direct mappings from the A-law 8-bit compressed ADC format to a 12-bit linear binary code. and fiam the p-law d-bit compressed format to a 13-bit hear code [S]. This operation permits the AID conversion to be performed a uniform quantizer and then to be mapped to the smaller number of bits in a code converter This also permits the inverse mapping at the receiver (i.e., the expansion) to be informed on the digital sample.

4.5 DIFFERENTIAL PULSE-CODE MODULATION

By the use of past data assist in measuring (i.e., quantizing) new data, we leave ordinary. PCEvI and enter the realm of differential(DPCIVI).I n DPCM. a prediction of the nest sample value is formed from past \.ides. This prediction can be thought of as instruction for the quantizer to conduct its search for the next Sample \.slue in a particular interval. By using the redundancy in the signal to form a prediction .The region of uncertainty is reduced and the quantization can be performed with a reduced number of decisions (or bits) for a given quantization level or \\-ith reduced quantization levels for a given number of decisions. The reduction in redundant is realized by subtracting the prediction from the next sample. This difference is called the ***prediction error.*** The quantizing methods described in Section 13.2 are called ***instantaneous error*** or ***true error.*** In Section 13.1 we identified the properties of sources' that permitted SOLIrCr {ate reductions. These properties where non-equiprobable source levels and non independent sample values. Instantaneous quantizers achieve

source-coding gain? by taking into account the probability density assignment each sample. The quantizing methods that take account of sample-to-sample correlation are noninstantaneous quantizers. These quantizers reduce source redundant bit First converting the correlated input sequence into a related sequence with reduced correlation. reduced variance, or reduced bandwidth. This new sequence is then quantized with fewer bits. The correlation characteristics of a source can be visualized in the time domain by samples of its autocorrelation function and in the frequency domain by its power spectrum. If we examine a power spectrum G,(n of a short-term speech signal, as shown in Figure 13.18, we find that spectrum has a global maxima in the neighborhood of 300 to 800 Hz and falls off at a rate of 6 to 12 dB octave. By interpreting

this power spectrum, we can infer certain properties of the time function from which it was derived. We observe that large changes in the signal occur slowly (low frequency) and that rapid changes in the signal (high frequency) must be of low amplitude. An equivalent interpretation can be found in the autocorrelation function ***R,(T)*** of the signal, as shown in Figure 13.19. Here a broad, slowly changing autocorrelation function suggests that there will be only slight change on a sample-to-sample basis, and that a time interval exceeding the correlation distance

is required for a full amplitude change. The correlation distance seen in Figure 13.19 is the time difference between the peak correlation and the first zero correlation. In particular, correlation values for typical single-sample delay is on the order of 0.79 to 0.87, and the correlation distance is on the order of 4 to 6 sample intervals of Tseconds per interval. Since the difference between adjacent time samples for speech is small, coding techniques have evolved based on transmitting sample-to-sample differences rather than actual sample values. Successive differences are in fact a specla1 case of a class of non-instantaneous converters called N-tap linear predictive scders. These coders, sometimes called predictor-corrector coders, predict the next input sample

value based on the previous input sample values. This structure is shown in Figure **11** 20. !n this type of converter, the transmitter and the receiver have the same prediction level, which is derived from the signal's correlation characteristics. The code forms the prediction error (or the residue) as the difference between the next measured sample value 2nd the predicted sample value. The equation for the prediction loop is

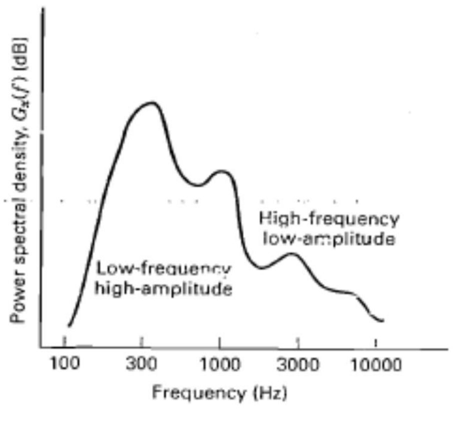


Figure 13.18 Typical power spectrum for speech signals.

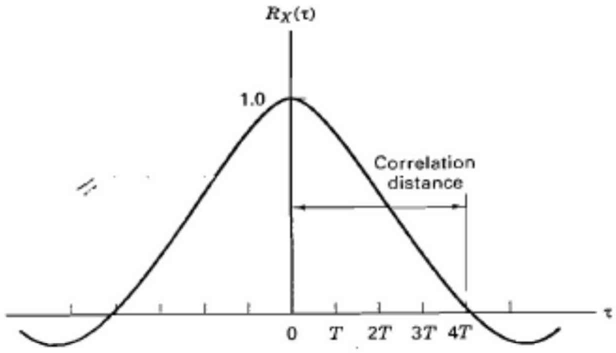


Figure 13.19 Autocorrelation function for typical speech signals.

$$d(n) = x(n) - \hat{x}(n)$$

where $x(n)$ is the nth input sample, $\hat{x}(n)$ is the predicted value of that sample, and $d(n)$ is the associated prediction error. This is performed in the predict-and-compare loop, the upper loop of the encoder shown in Figure 13.20. The encoder corrects its prediction by forming the sum of its prediction and the prediction error. The equations for the correction loop are

$$\tilde{d}(n) = \text{quant}\,[d(n)]$$
$$\tilde{x}(n) = \hat{x}(n) + \tilde{d}(n)$$

where qusnt (-) represents the quantization operation, ***c?(h)*** is the quantized version of the prediction error. and i(n) is the corrected and quantized version of the input sample. Tlii; is performed in the predict-and-correct loop, the lower loop of the encode,, and the only loop of the decoder in Figure 13.20. The decoder must also be informed of the prediction error so that it can use its correction loop to correct its prediction. The decoder "mimics" the feedback loop of the encoder. The communication task is that of transmittiria the difference (the error signal) between the predicted and the actual data sample. For this reason, this class. of coder-is often

called a differential pulse cod^ modulator (3PCM). If the prediction model forms predictions that are close to the actual sample values, the residues will exhibit reduced variance (relative to the original signal). From Section 13.2, we know that the number of bits required to move data through the channel with a given fidelity is related to the signal variance. Hence, the reduced variance sequence of residues can be moved through the channel with a reduced data rate.

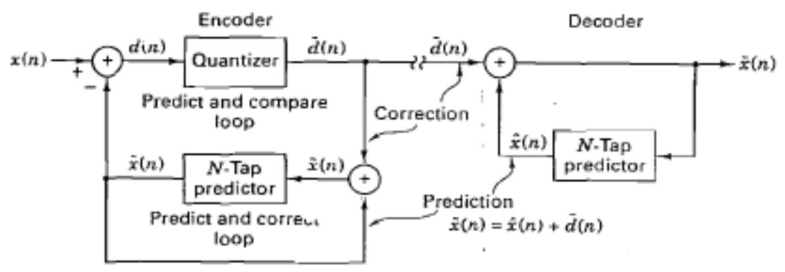


Figure 13.20 *N*-tap predictive differential pulse code modulator (DPCM).

4.5.1 One-Tap Prediction

The one-tap linear prediction coding (LPC) filter in the DPCM process predicts the next input sample value based on the previous input sample value. The prediction equation is of the form

$$x(n|n-1) = a\,x(n-1)|n-1) \qquad (13.44)$$

where x(n1nz) is the estimate of **.r** at time tl given all the samples collected up through time *nz,* and where -0" is a Gitrameter used to minimize the prediction error. The prediction error available after the measurement is of the form

$$d(n) = [x(n) - x(n|n-1)] \qquad (13.45a)$$

$$= [x(n) - a\,x(n-1|n-1)] \qquad (13.45b)$$

The mean-squared error is of the form

$$\mathbf{E}\{d^2(n)\} = \mathbf{E}\{x(n)\,x(n) - 2\,a\,x(n)\,x(n-1\,|\,n-1) + a^2\,x(n-1\,|\,n-1)\,x(n-1\,|\,n-1)\} \qquad (13.46)$$

If $x(n-1\,|\,n-1)$ is an unbiased estimate of $x(n-1)$, Equation (13.46) can be written as

$$R_d(0) = R_x(0) - 2\,a\,R_x(1) + a^2\,R_x(0) \qquad (13.47a)$$

$$= R_x(0)\,[1 + a^2 - 2\,a\,C_x(1)] \qquad (13.47b)$$

where R,t(n) and R,(tz) are the autocorrelation functions of the prediction error and the input signal, respectively. R,,(O) is the power in the error, R,(O) is the power in the signal, and C,(IZ) = R,(II)IR,(O) is the normalized autocorrelation function. We can select the parameter *o* to minimize the Prediction error power of Equation (13.47) by setting **a** zero the partial derivative of R./(O) with respect to n:

$$\frac{\partial R_d(0)}{\partial a} = R_x(0)\,[2a - 2C_x(1)] \tag{13.48}$$

Setting to zero and solving for a^{opt}, the optimal solution, we have

$$a^{\text{opt}} = C_x(1) \tag{13.49}$$

Substituting a^{opt} back into Equation (13.47), we obtain

$$R_d^{\text{opt}}(0) = R_x(0)[1 + a^{\text{opt}}C_x(1) - 2a^{\text{opt}}\,C_x(1)] \tag{13.50a}$$
$$= R_x(0)\,[1 - a^{\text{opt}}\,C_x(1)] \tag{13.50b}$$
$$= R_x(0)\,[1 - C_x^2(1)] \tag{13.50c}$$

we can define the prediction of the encoder as the ratio of input to output variances. R,(O)IR,(O), For a fixed bit rate this gain represents an increase in output SNR, while for a fixed output SNR this gain represents a reduced bit-rate description. We note that the prediction gain for the optimal predictor is always greater than any value of signal. correlation R,(O) -%- used in Equation (13.50b). On the other hand. the prediction gain is greater than one for the nonoptimum unity gain. one-tap predictor, only if the signal correlation exceeds 0.5 as used in Equation (13.47b).

4.5.2 NTap Prediction

The N-tap LPC filter predicts the next sample value based on a linear combination of the previous *N* sample \.slues. We will assume that the quantized estimates used by the prediction filters are unbiased and error free. With-this assumption; *we* can drop the double indices (used in Section 13.3.1) from the data in the tllter but still use them for the predictions. Then the N-tap prediction equation lakes the form

$$x(n \mid n-1) = a_1 x(n-1) + a_2 x(n-2) + \cdots + a_N x(n-N) \tag{13.52}$$

The prediction error takes the form

$$d(n) = x(n) - x(n \mid n-1) \tag{13.53a}$$
$$= x(n) - a_1 x(n-1) - a_2 x(n-2) - \cdots - a_N x(n-N) \tag{13.53b}$$

The mean-square prediction error is of the form

$$\mathbf{E}\{d(n)d(n)\} = \mathbf{E}\{[x(n) - x(n \mid n-1)]^2\} \tag{13.54}$$

Clearly. the mean-square prediction error is quadratic in the filter coefficients n, As we did in Section 13.3.1. we can form the partial derivative of the mean squared error with respect to each coefficient and solve for the coefficients that set the partials to zero.'Formally, taking the partial derivative with respect to the jth coefficient prior to expanding .r(nln - I), we have

$$\frac{\partial R_d(0)}{\partial a_j} = \mathbf{E}\left\{2[x(n) - x(n|n-1)]\frac{\partial x(n|n-1)}{\partial a_j}x(n|n-1)\right\} \tag{13.55a}$$

$$= \mathbf{E}\{2[x(n) - x(n|n-1)][-x(n-j)]\} \tag{13.55b}$$

$$= 2\mathbf{E}\{[x(n) - a_1x(n-1) - a_2x(n-2) - \cdots - a_Nx(n-N)][-x(n-j)]\} \tag{13.55c}$$

$$= 2[R_x(j) - a_1R_x(j-1) - a_2R_x(j-2) - \cdots - a_NR_x(j-N)] \tag{13.55d}$$

This collection of equations (one for each j) can be arranged in matrix form known as the normal equations. This form is

$$\begin{bmatrix} R_x(1) \\ R_x(2) \\ R_x(3) \\ \vdots \\ R_x(N) \end{bmatrix} = \begin{bmatrix} R_x(0) & R_x(-1) & R_x(-2) & \cdots & R_x(-N+1) \\ R_x(1) & R_x(0) & R_x(-1) & \cdots & R_x(-N+2) \\ R_x(2) & R_x(1) & R_x(0) & \cdots & R_x(-N+3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_x(N-1) & R_x(N-2) & R_x(N-3) & \cdots & R_x(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_N \end{bmatrix}^{\text{opt}} \tag{13.56a}$$

The normal equations can be written more compactly as

$$\mathbf{r}_x(1,N) = \mathbf{R}_{xx}\,\mathbf{a}^{\text{opt}} \tag{13.56b}$$

where $\mathbf{r}_x(1,N)$ is the correlation vector of delays from 1 through N, $\mathbf{R}_{xx}$ is the correlation matrix (assuming a zero-mean process), and $\mathbf{a}^{\text{opt}}$ is the optimum filter weight vector.

To gain insignificant o the solution of the normal equations, we now rec2-t the mean-squrred erroer q uation (13.54) in matrix :arm. We have

$$R_d(0) = \mathbf{E}\{[x(n) - \mathbf{a}^T\mathbf{x}(n-1)][x(n) - \mathbf{x}^T(n-1)\mathbf{a}]\} \tag{13.57a}$$

$$= R_x(0) - \mathbf{r}_x^T(1,N)\,\mathbf{a} - \mathbf{a}^T\mathbf{r}_x(-1,-N) + \mathbf{a}^T\mathbf{R}_{xx}\mathbf{a} \tag{13.57b}$$

where $\mathbf{r}^T$ is the transpose of $\mathbf{r}$. Substituting $\mathbf{a}^{\text{opt}}$ for $\mathbf{a}$ in Equation (13.57b), and then substituting $\mathbf{r}_x(1,N)$ for $\mathbf{R}_{xx}\mathbf{a}^{\text{opt}}$ in the resulting equation, yields

$$R_d(0) = R_x(0) - \mathbf{r}_x^T(1,N)\,\mathbf{a}^{\text{opt}} - \mathbf{a}^{\text{opt}\,T}\mathbf{r}_x(-1,-N) + \mathbf{a}^{\text{opt}\,T}\mathbf{r}_x(1,N) \tag{13.58a}$$

$$= R_x(0) - \mathbf{r}_x^T(-1,-N)\,\mathbf{a}^{\text{opt}} \tag{13.58b}$$

We can now bring the right-hand side of Equation (13.56) over to the left-hand side, and use Equation (13.58b) to augment the top row of the matrix to obtain the whitening form of the optimal predictor:

$$\begin{bmatrix} R_x(0) & R_x(-1) & R_x(-2) & R_x(-3) & \cdots & R_x(-N) \\ R_x(1) & R_x(0) & R_x(-1) & R_x(-2) & \cdots & R_x(-N+1) \\ R_x(2) & R_x(1) & R_x(0) & R_x(-1) & \cdots & R_x(-N+2) \\ R_x(3) & R_x(2) & R_x(1) & R_x(0) & \cdots & R_x(-N+3) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ R_x(N) & R_x(N-1) & R_x(N-2) & R_x(N-3) & \cdots & R_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ -a_1 \\ -a_2 \\ -a_3 \\ \vdots \\ -a_N \end{bmatrix}^{\text{opt}} = \begin{bmatrix} R_d(0) \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{13.59}$$

In this form, the only nonzero output of the matrix product occurs at time zero, which is akin to an output impulse.

The top row of Equation (13.59) states that the power in the prediction error is of the form

$$R_d(0) = R_x(0)[1 - a_1C_x(1) - a_2C_x(2) - \cdots - a_NC_x(N)] \tag{13.60}$$

4.5.3 Delta Modulation

The delta modulator, often denoted **A** modulator, is a process that embeds a low resolution A-to-D converter in a sampled data feedback loop that operates at rates far in excess . The activation for this technique is our awareness that in the conversion process, speed is less expensive than precision. and that by being clever one can use faster signal processing to obtain higher precision. Equation (13.50~d) demonstrated that the prediction gain for a one-tap ;rt;&ctor could be large if the normalized correlation coefficient, C,(l), is close to unity. Working toward the goal of high sample-to-sample correlation, the predictive filter in generally operated at a rate that far exceeds the Nyquist rate. For example, the

sample rate might be chosen to be 64 times the Nyquist rate. This for a 20 khz bandwidth with a nominal sample rate of 45 kHz, the high correlation prediction filter would operate at a 3.072 MHz sample rate. The justification for the high sample rate is to insure that the sampled data is highly correlated so that a sim1,leonetap predictor will exhibit a small prediction error. which in turn permits the quantizer operating in the error loop **LJ** operate with a very small number of bits. The simplest form of the quantizer is a one-bit quantizer, which is, in fact, only a

comparator that detects and reports the sign of the difference signal. As a result, the prediction error signal is a 1 -bit word that has the interesting advantage of not requiring word framing in subsequent processing. The block diagram of the one-tap linear predictor, illustrated in Figure 13.20,is shown in Figure 13.21, with slight modifications. Note that the one-tap predict

and correct loop is now a simple integrator and that a low-pass reconstruction filter follows the predict and correct loop at the decoder. This filter removes the out-of band quantizing noise that is generated by the two-level coding and that extends beyond the information bandwidth of this coding process. The coder is completely characterized by the sampling frequency. the quantizing step size to resolve the prediction error or ***delta*** of the loop. and the reconstruction filter. The equations for re diction and for the residual error of the modulator are of the form

$$x(n \mid n-1) = x(n-1 \mid n-1) \qquad (13.61a)$$

$$d(n) = x(n) - x(n \mid n-1) \qquad (13.61b)$$

where n is a sample index. This structure, sometimes called the *deltamodulator*, is a DPCM process for which the predict-and-correct loop consists of a digital accumulator.

4.5.4 Sigma-Delta Modulation

The structure of the **X-A** modulator can be examined from a number of perspectives, the most appealing being that of a modified one-tap DPCM converter, and that of an error-feedback converter. Let us start with the modified one tap DPCM

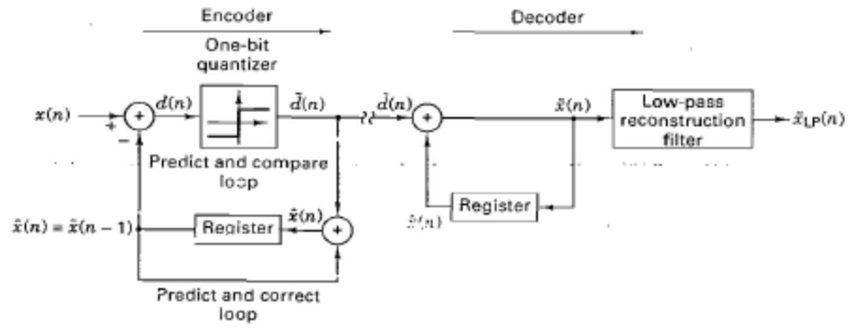


Figure 13.21 One-tap, one-bit DPCM coder (delta-modulator).

converter. As indicated the loop relies on high correlation of successive samples. a condition **wc** assume by significant oversampling. We can enhance the correlation of the sampled data presented to the modulator by pre-filtering the data with an integrator and then compensating for the prefilter with a post-filtering differentiator. This structure is shown in Figure 13.22, where the integrators. differentiator, and delay functions are expressed in terms of the *z* transform. (See Appendix E.) \ire can then rearrange the signal flow blocks to realize an economy of irnplementaion. .At the input to the encoder, there are the outputs of two digital integrators. which are summed and presented to the loop quantizer. Our first modification is that \\e can share a single digital integrator by sliding the two integrators r!~rough the summing junction in the encoder. Our second modification to the encoder is that the post filter differentiator can be moved **tn** the decoder, which then cancels the digital integrator at the input to the decoder. All their remains of the decoder 1s the Ion'-pass reconstruction filter. This simplified form of the modified DPCM system is shown in Figure 13.23. This form, called a ***sigma-delta*** modulator contains an integrator (the ***sigma)*** and a DPCM modulator (the ***delta)*** [ll]. The second perspective useful for understanding the H-A modulator 1s that of noise feedback loop. We understand that a quantizer adds an error to its input to form its output. \\'hen the signal is highly oversampled. not only are the samples highly correlated. the errors are as well. When errors are highly correlated. They are predictable. and thus they can be subtracted from the signal presented to the quantizer prior to the quantization process. When the signal and error are highly oversampled. the previous quantization error can be used as a good estimate of the current error. The previous error, formed as the difference between the inpi,; and output of the quantizer. is stored in a delay register for rise as the estimate of the next quantization error. This structure is shown in Figure 13.24. The signal flow graph of Figure 13.24 can be redrawn to emphasize the two inputs, signal and quantization noise. as n ell as the two loops, one including the quantizer and one not including it. This form is shown in Figure 13.25 and except for explicitly showing the feedback leg of the digital integrator, this has the structure as presented in Fiqure 13.23. From F~a,ure 13.25, if follows that the output of the Z-\ modulator and its :-transform (see

Appendix E) can be written as

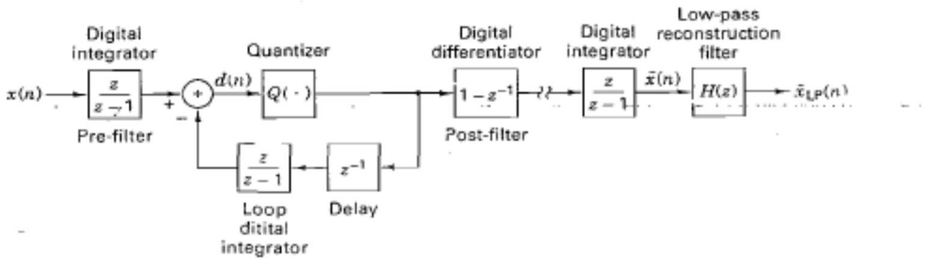


Figure 13.22 One-bit delta modulator, pre and post filtered form.

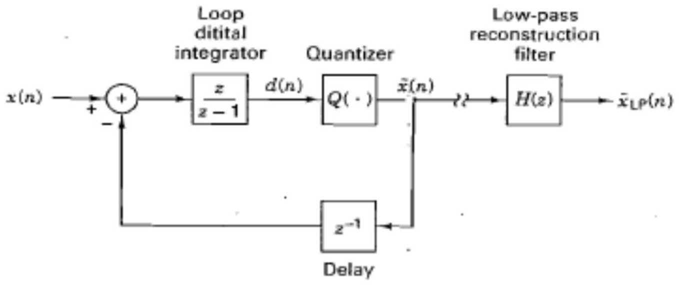


Figure 13.23 Σ-Δ modulator as a rearranged pre and post filtered Δ-modulator.

$$\begin{aligned} y(n) &= \bar{x}(n) = x(n) - q(n-1) + q(n) \qquad (13.62)\\ &= x(n) + [q(n) - q(n-1)] \end{aligned}$$

$$\begin{aligned} Y(Z) &= X(Z) - Z^{-1}Q(Z) + Q(Z)\\ &= X(Z) + Q(Z)[1 - Z^{-1}] \qquad (13.63)\\ &= X(Z) + Q(Z)\,\frac{Z-1}{Z} \end{aligned}$$

Intuitively, Equation (13.63) is pleasing, since it shows that the loop does not affect the input signal since only the noise is being circulated by the loop, and only the noise experiences the effect of the loop. The integrator in the feedback path of the

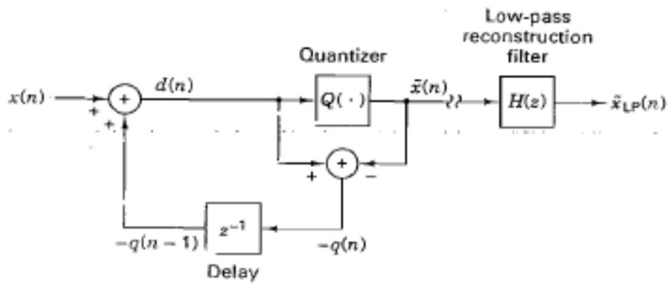


Figure 13.24 Σ-Δ modulator as a noise feedback process.

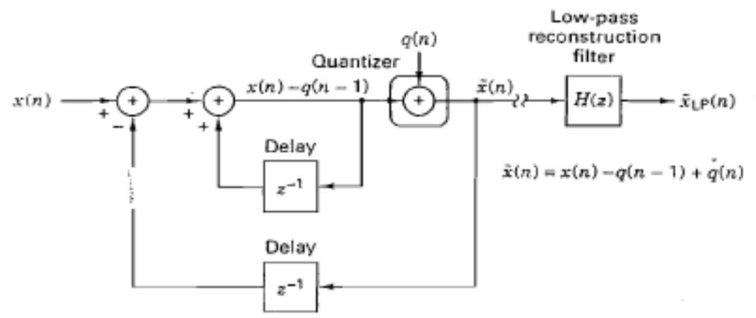


Figure 13.25 Noise feedback quantizer redrawn as Σ-Δ modulator.

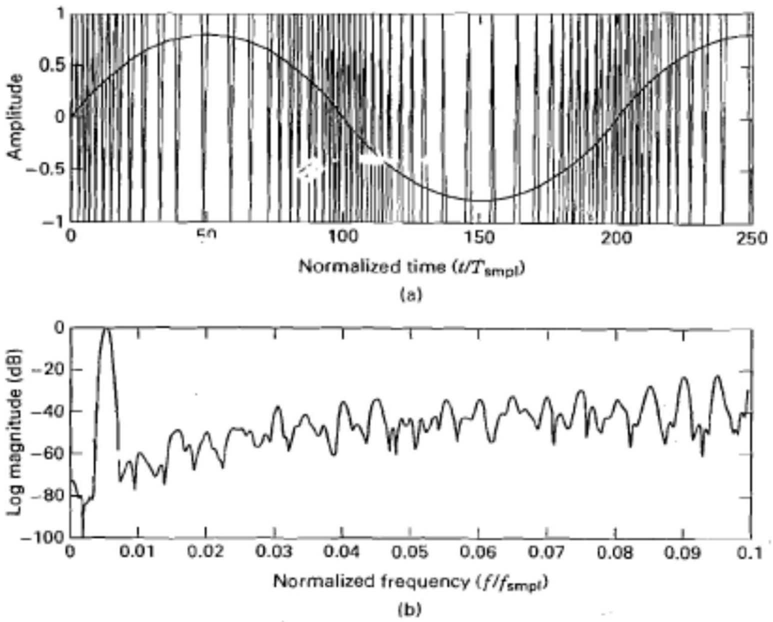


Figure 13.27 One bit Σ-Δ modulator: (a) input and output time series (b) spectral response.

Figure 13.27a presents an oversampled input sinusoidal signal and the corresponding output signal of a one-bit, dual-zero, **Z-A** modulator. Figure 13.27b shows the spectral response of the output series. Note that the shaped noise spectrum in the neighborhood of the signal is approximately 50 dB 5elow the peak spectrum of the input signal. Note also that the output signal is restricted to 21, and that the loop is esssentially duty cycle modulating the square wave is proportion to the amplitude of the input signal. figure 13.28 presents the time series and the spectrum obtained from the output of the down-sampling filter following the modulator.

4.5.6 Sigma-Delta D-to-A Converter (DAC)

In an ADC, has become an essential building block of the reverse task-the digital-to-analog converter (DAC). Nearly all high-end audio equipment and most communication system

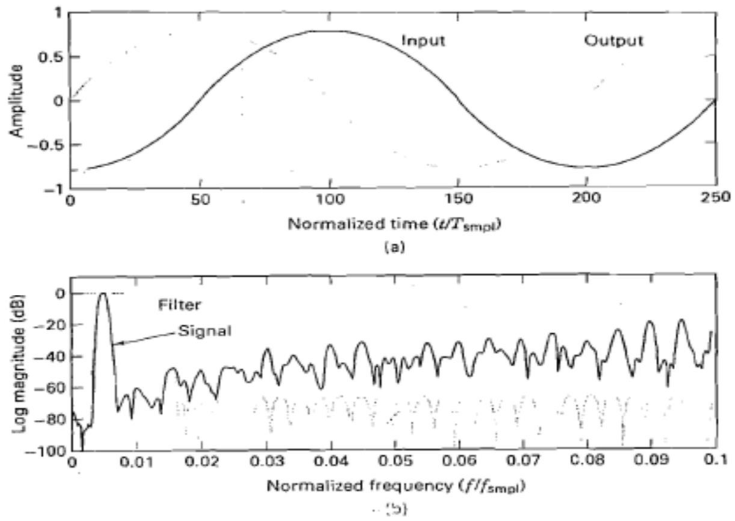


Figure 13.28 Post processing filter following Σ-Δ modulator: (a) input and output time series, (b) spectral response.

DACs are implemented with ***C-h*** converters. The process uses the *X-h* modulator as a digital-to-digital (D-to-D) transformation, which converts a high-precision (say, 16-bit) representation of oversampled digital data to a low-precision (say, 1-bit) representation of the same data. The oversampled one-bit data stream is then deliver a 1-bit DAC with two analog output levels defined with the same precision as a 16-hit converter. Here, the advantage of a one bit DAC operating at high speed but with only two levels is that speed is less expensive than precision.

The 2-lcvcl. high-speed DAC replaces a low-speed DAC that would have to resolve 65.55; distinct levels Very simple analog low-pass filtering following the 1-bit DAC suppr2sse.s the out-of-band noise spectrum and delivers the reduced bandwidth. high-fidelity version of the original digital data. The requantizing of the oversampled data is a straightforward signal processing task, using an all-digital X-A modulator. The only additional task to be performed when using a 2-1 DAC is the requirement to raise [he sample rate of the data to **64** times its Nyquist rate. This task is performed by a DSP-based interpolating filter. which is a standard

signal-processing block found in most systems that use a DAC to transition between a digital signal source and an analog output [12].As a standard illustration of th.: process, a CD player uses an interpolating filter to realize a l-to-l up-sampling, resulting in a separation of the periodic spectra associated with sampled data. This increased separation permits the smoothing filter

following the DAC to have a wider transition bandwidth, and hence, a reduced component count and reduced implementation cost. The CD specification uses terms such as "four-to-one resampled "to reflect the presence of the interpolating filters. Starting with the CD 1-to-4 interpolator, it is a simple task to again raise the sample rate by another factor of 1-to-16. with a second inexpensive interpolating filter. The data. now 64 times oversampled. is presented to the all-digital **P-A** modulator and one-bit DAC to complete the analog conversion process. This structure is shown in Figure 13.29. There are many signals that are significantly oversampled with respect to the signal's bandwidth. These signals can easily be converted to analog representations by the use of a **2-A** modulator and a 1-bit DAC. Examlples are control signals used in circuits such as AGC, carrier VCO. and timing VCO. Many systems employ the

Z-A modulator and I-bit DAC to generate and deliver analog control signals throughout the system. 13.5 BLOCK CODES The quantizers we have examined up to now have been *scalar* in nature, sinre they form a *single scalar sample* based on the present input sample and (possibly) the N preciovs output samples. Block coders, on the other hand, form a *vector of orcrplcr*

sctttzpies based on the present and the N previous input samples. The *coding gain* of a waveform coder is the ratio ol 1r.5 input SNR to the-output SNR. Where the noise variances of t!le input and output are equal, this gain is simply the ratio of input-to-output signal variances. The ratio convert. directly to 6 dB per bit for the difference between the number of input bits per sample and the average number of output bits per sample. Block coders can achieve impressive coding gains. On average, they can represent sequences quantized to S bits wit:; only **1-2:** 2 bits per sample [a]. Block-coding techniques are varied. but a control thread that runs

through block-coding techniques is the mapping of an input sequence *of* alternative coordinate system. This mapping may be to a subspace of a larger space, so that the mapping may not be reversible [S]. Alternatively, a data-dependent editing scheme may be used to identify the subspace of the mapping from which the quantized data are extracted. Block-coding techniques are often classified by their mapping techniques. which include, for example, vector quantizers, various orthogonal transform coders, and channelized coders, such as subband coders. Block coders are further described by their algorithmic structures, such as codebook, tree, trellis,

discrete Fourier transform, discrete cosine transform, discrete Walsh-Hadanlard transform, discrete Karhuy<-1;Loeve transform, and quadrature mirror filter-bank coders. We now examine examples of the various block-coding schemes. **13.5.1 Vector quantizing** Vector quantizers represent an extension of conventional scalar quantization. In scalar quantization, a scalar value is selected from a finite list of possible values to represent an input sample. The value is selected to be close (in some sense) to the sample it is representing. The fidelity measures are various weighted mean-square measures that preserve our intuitive concept of distance in terms of ordinary vector lengths. By extension, in vector quantization, a vector is selected from a finite list of possible vectors to represent an input vector of samples. The selected vector is chose^, in be close (in some sense) to the vector it is representing. Each input vector can be visualized as a point in an N-dimensional space. The quantizer is defined by a partition of this space into a set of non overlapping volumes **[14].** These volumes are called intervals. polygons, and polytopes, respectively, for one-. two-, and I\'-dimensional vector spaces. The task of the vector quantizer is to determine the \solumien which an input vector is located; The output

of the optimal quantizer is the vector identifying the centroid of that volume. As in the one-dimension quantizer. the mean-square error is a function of the boundary locations for the partition and the multidimensional pdf of the input vector. The description of a vector quintizer can be cast as two distinct tasks. The first is the code-design task, which deals with the problem of performing the multidimensional volume qantization partition) and selecting the allowable output sequences. The second task is that of using the code, and deals with searching for the particular volume with this partition that corresponds (according to some fidelity criterion) to the best description of the source. The form of the algorithm selected to control the complexity of encoding and decoding may couple the two tasks-the partition and the search. The standard vector coding methods are **Trellis algorithm**

4.6 Codebook, Tree, and Trellis Coders

The codebook coders are essentially table look-up algorithms; A list of candidate- patterns (code words) is stored in the codebook menial-y. Each pattern is identified by an address 01. pointel: indes The coding routine searches through the list of patterns fol- [he one that is closest to the input pattern and transmits to the receiver the address where that pattern can be fou:.ci in its codebook. The tree and **854** Source Coding Chap. **13** trellis coders are sequential codcrs. As such, the allowable code words **of** the code cannot be selected independently but must exhibit a node-steering structure. This is similar to the structure of the sequence tiaelrr or-detection-and-correction algorithms which traverse the branches of a graph while forming the branch weight approximati011 to the input sequence. A tree graph suffers from exponential memory growth as the dimension or depth of the tree increases. The trellis graph reduces the dimensionality problem by the simultaneous tracking of contender paths with an associated path-weight metric called ***inlmsit)***~w**,** ith the use of a finite state trellis Code Population The code vectors stored in the codebook, tree, or trellis are the likely or typical vectors. The first task. that of code design, in which the likely code vectors are

identified is called poprllrrring the code. The methods of determining the code population

are classically ***deternministic, stochastic,*** and ***iterative.*** The deterministic population is a list of pre assigned possible outputs based on a simple suboptinla1 or user-perception fidelity criterion or based on a simple decoding algorithm. An example of the former is the coding of the samples in 3-space of the red, green. And blue (RGB) components of a color TV signal. The eye does not have the same resolution to each color and it would appear that the coding could be applied independently to each color to reflect this different sensitivity. The resulting quantizing

volumes would be rectangular parallelepipeds. The problem with independent quantizing is that we do not see images in this coordinate system; rather: \tre see im- I ayps in the coordinates of luminance, hue, and saturation. A black-and-white photo. for example, uses only the luminance coordinate. Thus, quantizillg RGB co- I ordinates independently does not result in the smallest amount of user-perceived distortion for a given number of bits. To obtain improved distortion performance, the RGB quantizer should partition its space into regions that reflect the partitions

in the alternate space. Alternatively, the quantization could be performed independently in the alternative space by the use of transform coding, treated in Section 13.6. Deterministic coding is the easiest to implement but leads to the smallest coding gain (smallest reduction in bit rate for a given SNR). ' The stochastic population would be chosen based on an assumed underlying

pdf of the input samples. Iterative solutions to the optimal partitions euist and can be determined for any assumed pdf. The overall samples are modeled by the assumed pdf. In the absence of an underlying pdf, iterative techniques based on a large population of training sequences can be used to form the partition and the output population. Training sequences involve tens of thousands of representative input samples.

4.6.1 Searching

Given a11 input vector and a codebook, tree, or trellis, the coder algorithm must conduct a search to determine the best matching contender vector. An exhaustive search over all possible contenders will assure the best match. Coder performance improves for larger dimensional spices, but so does complexity. **AP** exhaustive search over a large dimension may be prohibitively time consuming. An alternative is to conduct a non exhaustive, suboptimal search scheme, with acceptably small degradations form the optimal path. Memory requirements and computational complexity often are driving consideration in the selection of search algorithms. Examples of search algorithms include single-path (best leaving branch) algorithms, multiple-path algorithms, and binary (successive approximation) codebook algorithms. Most of the search algorithms attempt to identify and discard unlikely patterns ; without having to test the entire pattern.

4.7 TRANSFORM CODING

In Section L3.5.1 we examined vector quantizers in terms of a set of likely patterns and techniques to determine the one pattern in the set closest to the input pattern. One measure of goodness of approximation is the weighted mean-square error of the form

$$d(\mathbf{X}, \hat{\mathbf{X}}) = (\mathbf{X} - \hat{\mathbf{X}})\mathbf{B}(\mathbf{X})(\mathbf{X} - \hat{\mathbf{X}})^T \quad (13.76)$$

where B(S) is a weight matrix and **X'** is the transpose of the vector **X.** The minimization may be computationally simpler if the weighting matrix is a diagonal matrix. A diagonal weighting matrix implies a decoupled (or uncorrelated) coordinate set so that the error minimization due to quantization can be performed independently over each coordinate. Thus. transform coding entails the following set of operations, which are given in Figure 13.32:

1. An invertible transform is applied to the input vector.

2. The coefficients of the transform are quantized.

3. The quantized coefficients are transmitted and received.

4. The transform is inverted with quantized coefficients.

Note that the transform does not perform any source encoding, it merely allows for a more convenient description of the signal vector to permit ease of source encoding. The task of the transform is to map a correlated input sequence into a different coordinate system in bb;~icil the coordinates have reduced correlation. Recall that . . this is precisely the task performed by predictive codes. The source encoding occurs with the bit assignment to the various coefficients of the transform. As part of t!?is assignment, the coefficients may be partitioned into subsets that are quantized with a rliff?r~nntu mber of bits but ***not*** with different quantizing step sizes. This assignme~ tr eflects the dy~i~nlriacn ge (variance) of each coefficient and may be

weighted by a measure that reflects the importance, rehtive to the human perception [17], of the basis element carried by each coefficient. A subset of the coefficients, for instance, may be set to zero amplitude, or may be quantized wit11 1 or 2 bits.

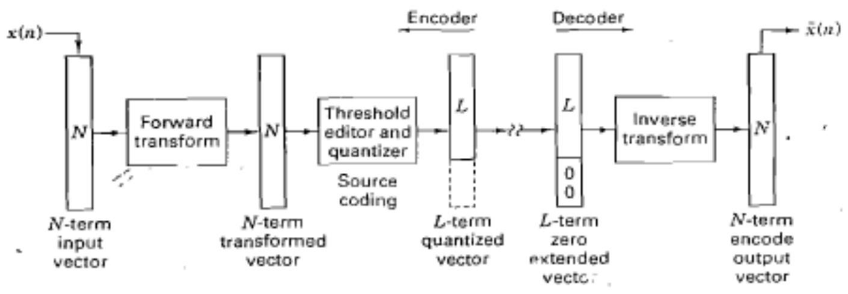


Figure 13.32 Block diagram: Transform coding.

4.7.1 Quantization for Transform Coding

Transform coders are called spectral encoders because the signal is described in terms of a spect,il decomposition (in a selected basis set). The spectral terms a1.p computedfo r non overlapped successive blocks of input data. Thus, the output of a transform coder can be viewed as a set of tin12 series, one series for each spectral term. The variance of each series can be determined and each can be quantized ,'with a different number of bits. By pel-;;;ittin: indepeni;;;;: quantization af each transform coefficient, \\,e have the option of allocating a fixed number of ï bits among the iLiu~sformco efficients to obtain a minimum quantizing error.

4.7.2 Subband Coding

The transform coders of Section 13.6 wei-e described as a partition of an input signal int,? ? --l!ect;nn nf ~!nwlvv arying time series, each of which is associated with a particular basis vector of the transform. The specil.al rerms. the inner product of the data with the basis vectors, are ijmputed by a set of inner products. The set of inner prod- 13.6 Transform Coding 857 ucts can be computed by a **set** offitrile irtrpulse r-esl>orzsc (FIR) filters [19]. With this perspective, the transform coder can be considered to be performing a channelization of the input data. By extension, a subband coder, which performs a spectral channelization by a bank of contiguous narrowband filters, can be considered a special case of a transfomal coder. (A typical subband coder is shown in Figure 13.33.) Casting the spectral decomposition of the data as a filtering task affords us the option of forming a class of custom basis sets (i.e.. spectral filters)-in particular,

basis sets that reflect out user-perception preferences and our source models. For example, the quantizino, noise generated in a band with large variance will be confined to that band, not spilling into a nearby band with low variance and hence susceptible to low-level signals being masked by noise. We also have the option of forming filters with equal or unequal bandwidths. (See Figure 13.33.) Thus. we can independently assign to each subband the sample rate appropriate to its bandwidth and a number of quantizing bits appropriate to its variance. By comparison in conventional transform coding, each basis vector amplitude is sampled at the same rate. The subband coder can be designed as a transmultiplexer. Here the input signal

is considered to be composed of a number of basis functions modeled as independent narrow-bandwidth subchannels. The encoder separates or channelizes the input signal into a set of low-data-rate, time-division multiplexed (TDM) channels.

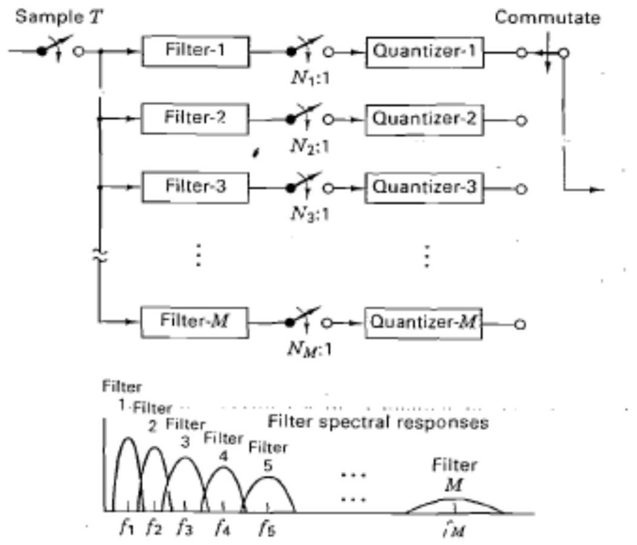


Figure 13.33 Subband coding performed by channelized spectral decomposition.

After quantization and transmission. the decoder reverses the filtering and resampling process, converting the TDM channels back to the original signal. In the classic approach to this process, one can implement a bank of narrow-band filters with the steps 01 heterodyning. low-pass filtering, and down sampling (often called deciniariot~). T his filtering operation reduces the input bandwidth to the selected channel bandwidth and re samples the signal to the lowest rate that avoids aliasing of the red13 led bandwidth channelized data. At the receiver, the reverse process is performed. The channelized signals are passed through interpolating filters to increase

their sample rate to the desired output sample rate are heterodyned back to their proper spectral position. and they are combined to form the original composite signal. For speech encoding-or, more generally, for signals that are re1atc.i to mechanical resonance-filter banks with non equal center frequencies and non equal bandwidths are desirable. Such filters are called constant-Q (or proportional) filter banks. These filters have logarithmically spaced center frequencies with bandwidths proportional to the center frequencies. This proportional spacing appears as

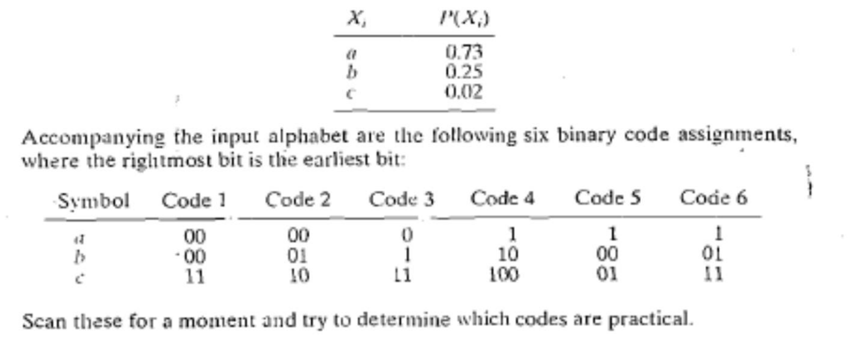
uniform spacing and bandwidth when viewed on a log scale, and it reflects the spectral properties of many physical acoustic sources.

4.8 SOURCE CODING FOR DIGITAL DATA

Coding to reduce the-redundancy of a data source entails the selection of an (usually) efficient binary representation of that source. Often this requires the substitution of one binary representation of the source symbols with an alternative representation. The substitution is usually temporary and is performed to achieve an economy of storage *c.'* transmission the discrete source symbols. The binary code assigned to each source symbol must satisfy certain constraints to permit reversal of the substitution. In addition. the code may be further constrained by system considerations, such as memory limit: T'; implementation ease.

4.8.1 Properties of Codes

. Earlier we alluded to properties that a code must satisfy for it to be useful. Some of these properties are -?.:-l.~s, .-.?? sc-:: are not. It is worth listing and demonstrating the ***desired properties.*** Consider the following three-symbol alphabet with the probability assignments shown:



X_i	$P(X_i)$
a	0.73
b	0.25
c	0.02

Accompanying the input alphabet are the following six binary code assignments, where the rightmost bit is the earliest bit:

Symbol	Code 1	Code 2	Code 3	Code 4	Code 5	Code 6
a	00	00	0	1	1	1
b	00	01	1	10	00	01
c	11	10	11	100	01	11

Scan these for a moment and try to determine which codes are practical.

Uniquely decodable Property. Uniquely decodable codes are those that allow us to insert the mapping to the original symbol alphabet. Obviously. code 1 in the preceding example is not uniquely decodable because the symbols *n* and b care assigned the same binary sequence. Thus, the first requirement of a useful code is that each symbol be assigned a unique binary sequence. By this condition. all the other codes appear satisfactory until me examine codes 3 and 6 carefully. These codes indeed have unique binary sequences assigned to each symbol. The problem (I occurs \\.hen these code sequences are strung together. For instance, try to decode

the binary pattern 1 0 1 1 1 in code 3. Is it *b,* ***n,*** b, 6, ***b*** or *6,* a, *6,* **c** or *6,* ***n, c,*** *b?* Trying to decode the same sequence in code 6 gives similar difficulties. These codes are not uniquely decodable, even though the individual characters ha\.c unique code assignments.

Prefix Free Property A sufficient (but not necessary) condition to assure that ***r7*** code is uniquely decodeable is that no codeword be the prefix of any other code word. Codes that satisfy this condition are called prefix-free codes. Note that code 1 is not prefix-free. but it is uniquely codable prefix-free codes also have the property that the!; al-c instantaneously decodable. Code 4 has a property that may be undesirable; it is not instantaneously decodable. An instantaneously decodable code is one for which the boundary of the present codeword can be identified **,by** the end of the present codeword, rather than by the beginning of the nest codeword. 's; instance, in transmitting the symbol with the binary sequence.1 in code. the receiver cannot determine if this is the whole codeword for symbol b or the partial codeword for symbol c. By contrast, codes 2 and 5 are prefix free.

4.8.2 Code Length and Source Entropy

.At the beginning of the chapter, we described the formal concept of information content and source entropy. We identified the self-information I(Y,,). in bits, about the symbol X,, denoted as I(X,,) = log[l|P(X,,)]. F:.;;n the perspective [hat information resolver, uncertainty we recognize that the information content of a symbol goes to zero as the probability of that symbol goes to unity. We also defined the entropy of a finite discrete source as the average information of that source. From the perspective that information resolves uncertainty, the entropy is the average

amount of uncertainty resolved per use of the alphabet. It also represents the average number of bits per symbol required to describe the source. In this sense. it is also the lower bound of what can be achieved with some variable length data compression codes. A number of considerations may prevent an actual code from achieving entropy bound of the input alphabet. These include uncertainty in probability assignment and buffering constraints. The average bit length achieved by a given code is denoted by ***i?.*** This average length is computed as the sum of the binary code lengths *tli* weighted by the probability of that code symbol P(Xi).

$$\bar{n} = \sum_i n_i P(X_i)$$

4.8.3 Huffman Code

The Huffman code [20] is a prefix-free, variable-length code that can achieve the shortest average code length **;I** for a given input alphabet. The shortest average code length for a particular alphabet may be significantly greater than the entropy of the source alphabet. This inability to exploit the.pro~nised data_compression is related to the alphabet, not to the coding technique. Often the alphabet can be modified to form an extension code. and the same coding technique is then reapplied to achieve better compression performance. Compression performance is measured by the *cot77prec:rion* lurtio. This measure is equal to the ratio of the average number of bits per sample before compression to the average number-of bits

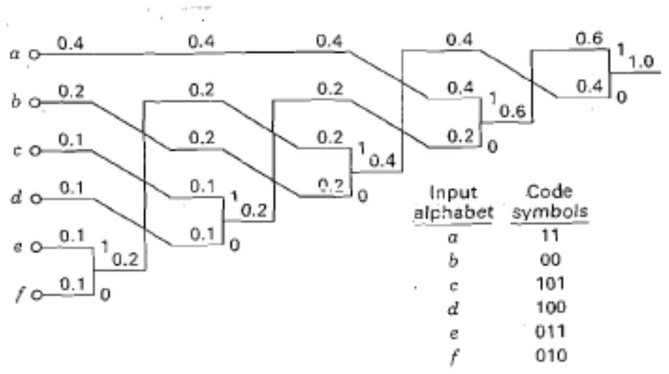
per sample after compression. The Huffman coding procedure can be applied for transforming between any two alphabets. We will demonstrate the application of the procedure between an

arbitrary input alphabet and a binary output alphabet. The Hoffman code is generated as part of a tree-forming process. The process starts by listing the input alphabet symbols, along \\lit11 their probabilities (or relative frequencies), in descending order of occurrence. These tabular entries correspond to the branch ends of a tree, as shown in Figure **13.34.** Each branch is assigned a branch weight equal to the probability of that branch. The process now forms the tree that supports these branches. The t\vo entries with the lowest relative frequency are merged (at a

branch node) to form a new branch with their composite probability. After every merging. the new branch and the remaining branches are reordered (if necessary) to assure that the reduced table preserves the descending probability of occurrence.We call this reordering burbling

[21]. During the rearrangement after each merging. the new branch rises through the table until it can rise no further. Thus, if we form a branch with a weight of 0.2 p!?d during the bubbling process find two other branches already with the 0.2 weight, the new branch is bubbled to the top of the 0.2 group, as opposed to simply joining it. The bubbling to the top of the group results in a code with reduced code length variance but otherwise a code with the same average length as that obtained by simple. joining the group. This reduced code length variance lowers the chance of buffer overflow. As an example of this part of the code process, we will apply the Huffman procedure to the input alphabet shown in Figure **13.34.** The tabulated alphabet ant1 the associated probabilities are shown on the figure. After forming the tree each branch node is labeled with a binary 110 decision to distinguish the two branches. The labeling is arbitrary, but for consistency, at each node we will label the branch going up with a -'I" and the branch going down with a "0". After labeling the branch nodes? we trace the tree path from the base of the tree (far right) to each output branch (far left). The path contains the binary sequence to reach that

branch. In the following table, we have listed at each end branch the path sequence corresponding to each path where ***i*** = **1,** . . .,*6* :


a 0.4 0.4 0.4 0.4 0.6 1 1.0
b 0.2 0.2 0.2 0.4 1 0.6 0.4 0
c 0.1 0.2 0.2 1 0.4 0.2 0
d 0.1 0.1 1 0.2 0.2 0
e 0.1 1 0.2 0.1 0
f 0.1 0

Input alphabet	Code symbols
a	11
b	00
c	101
d	100
e	011
f	010

Figure 13.34 Huffman coding tree for a six-character set.

MODULE :II

2.1 WAVEVEFORM COADING:-

2.1.1 ORTHOGONAL SIGNALS

Two signals x(t) and y(t) are said to be orthogonal if

$$\int_0^T x(t) \cdot y^*(t)dt = 0.$$

The integral above is referred to as the correlation between the two signals x(t) and y(t). Thus, two signals are orthogonal when their correlation is zero.

In signal space, the orthogonality condition becomes

$$\begin{aligned}\int_0^T x(t) \cdot y^*(t)dt &= \int_0^T \left[\sum_{i=1}^N x_i\phi_i(t)\right] \cdot \left[\sum_{j=1}^N y_j^*\phi_j^*(t)\right] dt \\ &= \sum_{i=1}^N \sum_{j=1}^N x_i y_j^* \int_0^T \phi_i(t)\phi_j^*(t)dt = 0 \\ &\Leftrightarrow \sum_{i=1}^N x_i y_i^* = 0.\end{aligned}$$

2.1.2 ANTIPODAL SIGNALS

Antipodal signalling uses a single pulse shape to represent bit values 0 and 1.Often,but not always ,the basic waveform will have a positive mean value .This waveform represents a 1 and the negative of the same pulse shape represents 0.Represented as:

$$s_1(t) = +\sqrt{\frac{E_b}{T_b}}; 0 \le t \le T_b$$

$$s_0(t) = -\sqrt{\frac{E_b}{T_b}}; 0 \le t \le T_b$$

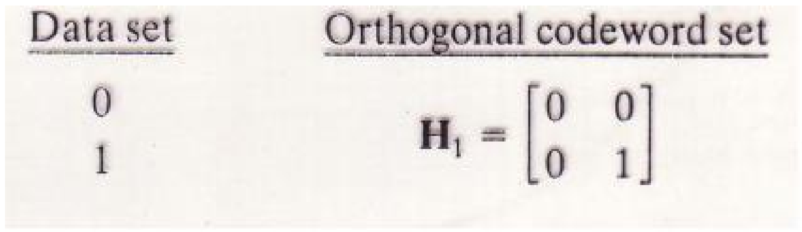
2.1.3 ORTHOGONAL AND BIORTHOGONAL CODES

Si(t) is an orthogonal set if and only if

$$z_{ij} = \frac{\text{number of digit agreements} - \text{number of digit disagreements}}{\text{total number of digits in the sequence}}$$

$$= \begin{cases} 1 & \text{for } i = j \\ 0 & \text{otherwise} \end{cases}$$

A one-bit data set can be transformed, using orthogonal codewords of two digits each, described by the rows of matrix H1 as follows:



Data set	Orthogonal codeword set
0 1	$\mathbf{H}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$

To encode a 2-bit data set, matrix H2 is created

$$
\begin{array}{c}
\underline{\text{Data set}} \\
0\ \ 0 \\
0\ \ 1 \\
\\
1\ \ 0 \\
1\ \ 1
\end{array}
\qquad
\begin{array}{c}
\underline{\text{Orthogonal codeword set}} \\
\mathbf{H}_2 = \left[\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right] = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_1 \\ \mathbf{H}_1 & \overline{\mathbf{H}_1} \end{bmatrix}
\end{array}
$$

BIORTHOGONAL

A biorthogonal signal set of M total signals or codeword can be obtained from an orthogonal set of M/2 signals as follows:

$$B_k = \begin{bmatrix} H_{k-1} \\ \hline \overline{H_{k-1}} \end{bmatrix}$$

A 3-bit data set can be transformed into abiorthogonal as follows:

$$
\begin{array}{c}
\underline{\text{Data set}} \\
0\ \ 0\ \ 0 \\
0\ \ 0\ \ 1 \\
0\ \ 1\ \ 0 \\
0\ \ 1\ \ 1 \\
\\
1\ \ 0\ \ 0 \\
1\ \ 0\ \ 1 \\
1\ \ 1\ \ 0 \\
1\ \ 1\ \ 1
\end{array}
\qquad
\begin{array}{c}
\underline{\text{Biorthogonal codeword set}} \\
\mathbf{B}_3 = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right]
\end{array}
$$

For the same data set, the biorthogonal code requires one-half as many code bits per code word over an orthogonal code. Thus the bandwidth for biothogonal is one-half the requirements for comparable orthogonal ones.

2.2 TYPES OF ERROR CONTROL

. In particular, we present, in this and subsequent chapters, a survey of óError control codingô techniques that rely on the systematic addition of óRedundantô symbols to the transmitted information so as to facilitate two basic objectives at the receiver: óError- detectionô and óError correctionô. We begin with some preliminary discussions highlighting the role of error control coding .The earlier chapters have given you enough background of Information theory and Source encoding. In this chapter you will be introduced to another important signal - processing operation, namely, ñ Channel Encodingò, which is used to provide óreliableô transmission of information over the channel.

Automatic Repeat Request (ARQ) - when a receiver circuit detects errors in a block of data, it requests the data to be retransmitted. Error detection and retransmission, utilizes parity bits (redundant bits added to the data) to detect an error.

Requires one link only, since in this case the parity bits are designed for both the detection and correction of errors. Forward Error Correction (FEC) - the transmitted data is encoded so that the data can correct as well as detect errors caused by channel noise.

2.2.1 TERMINAL CONNECTIVITY

The main task required in digital communication is to construct ócost effective systemsô for transmitting information from a sender (one end of the system) at a rate and a level of reliability that are acceptable to a user (the other end of the system). The two key parameters available are transmitted signal power and channel band width. These two parameters along with power spectral density of noise determine the signal energy per bit to noise power density ratio, Eb/N0 and this ratio, as seen in chapter 4, uniquely determines the bit error for a particular scheme and we would like to transmit information at a rate RMax = 1.443 S/N. Practical considerations restrict the limit on Eb/N0 that we can assign. Accordingly, we often arrive at modulation schemes that cannot provide acceptable data quality (i.e. low enough error performance). For a fixed Eb/N0, the only practical alternative available for changing data quality from problematic to acceptable is to use ñcodingò.

Another practical motivation for the use of coding is to reduce the required Eb/N0 for a fixed error rate. This reduction, in turn, may be exploited to reduce the required signal power or reduce the hardware costs (example: by requiring a smaller antenna size).

The coding methods discussed in chapter 5 deals with minimizing the average word length of the codes with an objective of achieving the lower bound viz. H(S) / log r, accordingly, coding is termed ñentropy codingò. However, such source codes cannot be adopted for direct transmission over the channel.

The encoder/decoder structure using ófixed lengthô code words will be very simple compared to the complexity of those for the variable length codes.

Here after, we shall mean by ñ Block codesò, the fixed length codes only. Since as discussed above, single bit errors lead to ó single block errorsô, we can devise means to detect and correct these errors at the receiver. Notice that the price to be paid for the efficient handling and easy manipulations of the codes is reduced efficiency and hence increased redundancy.

In general, whatever be the scheme adopted for transmission of digital/analog information, the probability of error is a function of signal-to-noise power ratio at the input of a receiver and the data rate. However, the constraints like maximum signal power and bandwidth of the channel (mainly the Governmental regulations on public channels) etc, make it impossible to arrive at a signaling scheme which will yield an acceptable probability of error for a given application. The answer to this problem is then the use of óerror control codingô, also known as óchannel codingô. In brief, ñerror control coding is the calculated addition of redundancyò . The block diagram of a typical data transmission system is shown in Fig.

The information source can be either a person or a machine (a digital computer). The source output, which is to be communicated to the destination, can be either a continuous wave form or a sequence of discrete symbols. The ósource encoderô transforms the source output into a sequence of binary digits, the information sequence u. If the source output happens to be continuous, this involves A-D conversion as well. The source encoder is ideally designed such that (i) the number of bints per unit time (bit rate, rb) required to represent the source output is minimized (ii) the source output can be uniquely reconstructed from the information sequence u.

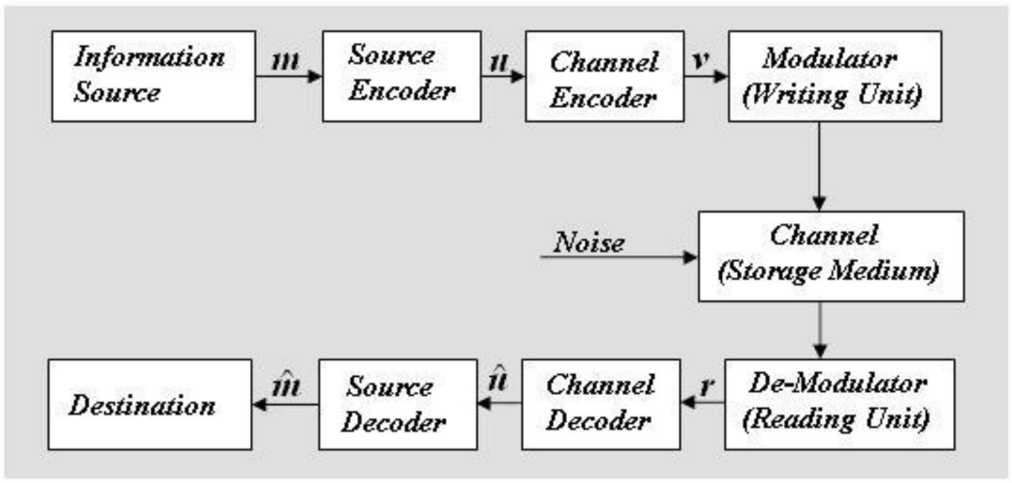


Fig 6.1 Block diagram of a Typical data transmission system.

The ó Channel encoderô transforms u to the encoded sequence v, in general, a binary sequence, although non-binary codes can also be used for some applications. As discrete symbols are not suited for transmission over a physical channel, the code sequences are transformed to waveforms of specified durations. These waveforms, as they enter the channel get corrupted by noise. Typical channels include telephone lines, High frequency radio links, Telemetry links, Microwave links, and Satellite links and so on. Core and semiconductor memories, Tapes, Drums, disks, optical memory and so on are typical storage mediums. A surface defect on a magnetic tape is a source of disturbance. The demodulator processes each received waveform and produces an output, which may be either continuous or discrete ï the sequence r. The channel decoder transforms r into a binary sequence, uĚ which gives the estimate of u, and ideally should be the replica of u. The source decoder then transforms uĚ into an estimate of source output and delivers this to the destination. The switching impulse noise, thermal noise, cross talk and lightning are some examples of noise disturbance over a physical channel.

2.2.2 AUTOMATIC REPEAT REQUEST

Error control for data integrity may be exercised by means of ó forward error correctionô (FEC) where in the decoder performs error correction operation on the received information according to the schemes devised for the purpose. There is however another major approach known as óAutomatic Repeat Requestô(ARQ), in which a re-transmission of the ambiguous information is effected, is also used for solving error control problems. In ARQ, error correction is not done at all. The redundancy introduced is used only for óerror detectionô and upon detection, the receiver requests a repeat transmission which necessitates the use of a return path (feed back channel).

In summary, channel coding refers to a class of signal transformations designed to improve performance of communication systems by enabling the transmitted signals to better withstand the effect of various channel impairments such as noise, fading and jamming. Main objective of error control coding is to reduce the probability of error or reduce the Eb/N0 at the cost of expending more bandwidth than would otherwise be necessary. Channel coding is a very popular way of providing performance improvement. Use of VLSI technology has made it possible to provide as much as 8 ï dB performance improvement through coding, at much lesser cost than through other methods such as high power transmitters or larger Antennas.

We will briefly discuss in this chapter the channel encoder and decoder strategies, our major interest being in the design and implementation of the channel ó encoder/decoderô pair to achieve fast transmission of information over a noisy channel, reliable communication of information and reduction of the implementation cost of the equipment.

2.3 STRUCTURED SEQUENCE:-

In structured sequences, extra bits are added to the message to reduce the probability of errors (detect errors).

ÅK-bit data block is transmitted as n-bit

message (n-k added bits).

ÅThe code is referred to as (n,k) code.

Åk/n is called the code rate

deals with transforming sequences into ñbetter sequencesò by adding structured redundancy (or redundant bits). The redundant bits are used to detect and correct errors hence improves overall performance of the communication system.

2.3.1 CHANNEL MODELS

Discrete Memoryless Channel (DMC)

ï A discrete input alphabet, a discrete output

alphabet

ï P(j|i) is the probability of receiving j given i

was sent.

Gaussian Channel

ï Generalization of DMC

ï Input is discrete alphabet, output is input+Gaussian noise

ï The demodulator output consists of a continuous alphabet, or quantized version of it

(>2 levels), the modulator made a soft decision

ï Since, the decision is not hard(0,1), there is no meaning of probability of error.

Binary Symmetric Channel

ï BSC is a special case of DMC

ï alphabet is 2 elements 0,1

ï P(0|1)=P(1|0)=p

ï P(1|1)=P(0|0)=1-p

ï Demodulator output is either 1 or 0, the modulator is said to make a hard decision

2.3.2 CHANNEL CAPACITY

While commenting on the definition of óChannel capacityô, where maximization should be with respect to all possible sets of input symbol probabilities. Accordingly, to arrive at the maximum value it is necessary to use some Calculus of Variation techniques and the problem, in general, is quite involved.

Clearly, the mutual information I (X, Y) depends on the source probabilities apart from the channel probabilities. For a general information channel we can always make I(X, Y) = 0 by choosing any one of the input symbols with a probability one or by choosing a channel with independent input and output. Since I(X, Y) is always nonnegative, we thus know the minimum value of the Transinformation. However, the question of max I(X, Y) for a general channel is not easily answered.

Our intention is to introduce a suitable measure for the efficiency of the channel by making a comparison between the actual rate and the upper bound on the rate of transmission of information. Shannonôs contribution in this respect is most significant. Without botheration about the proof, let us see what this contribution is.

2.3.3 CHANNEL CODING

class of signal transformations designed to improve communication performance by enabling the transmitted signals to better withstand channel distortions such as noise, interference, and fading.

Channel coding can be divided into two major classes:

1. Waveform coding by signal design
2. Structured sequences by adding redundancy

2.3.4 INFORMATION CAPACITY

It is possible, in principle, to device a means where by a communication system will transmit information with an arbitrary small probability of error, provided that the information rate

$R(=r\times I(X,Y)$, where r is the symbol rate) is less than or equal to a rate ó Cô called ñchannel capacityò.

The technique used to achieve this objective is called coding. To put the matter more formally, the theorem is split into two parts and we have the following statements.

Positive statement:

This theorem indicates that for $R< C$ transmission may be accomplished without error even in the presence of noise. The situation is analogous to an electric circuit that comprises of only pure capacitors and pure inductors. In such a circuit there is no loss of energy at all as the reactors have the property of storing energy rather than dissipating.

ñ Given a source of M equally likely messages, with $M>>1$, which is generating information at a rate R, and a channel with a capacity C. If R Ò C, then there exists a coding technique such that the output of the source may be transmitted with a probability of error of receiving the message that can be made arbitrarily smallò.

Negative statement:

ñ Given the source of M equally likely messages with $M>>1$, which is generating information at a rate R and a channel with capacity C. Then, if $R>C$, then the probability of error of receiving the message is close to unity for every set of M transmitted symbolsò.

You can interpret in this way: Information is poured in to your communication channel. You should receive this without any loss. Situation is similar to pouring water into a tumbler. Once the tumbler is full, further pouring results in an over flow. You cannot pour water more than your tumbler can hold. Over flow is the loss.

This theorem shows that if the information rate R exceeds a specified value C, the error probability will increase towards unity as M increases. Also, in general, increase in the complexity of the coding results in an increase in the probability of error. Notice that the situation is analogous to an electric network that is made up of pure resistors. In such a circuit, whatever energy is supplied, it will be dissipated in the form of heat and thus is a ñlossy networkò.

2.3.5 THE SHANON LIMIT

Shannon defines ñ Cò the channel capacity of a communication channel a s the maximum value of Transinformation, I(X, Y):

C = ⱷ Max I(X, Y) = Max [H(X) ï H (Y|X)] é é é é . (4.28)

The maximization in Eq (4.28) is with respect to all possible sets of probabilities that could be assigned to the input symbols. Recall the maximum power transfer theorem: óIn any network, maximum power will be delivered to the load only when the load and the source are properly matchedô. The device used for this matching purpose, we shall call a ñtransducer ñ. For example, in a radio receiver, for optimum response, the impedance of the loud speaker will be matched to the impedance of the output power amplifier, through an output transformer.

There exists a coding scheme for which the source output can be transmitted over the channel and be reconstructed with an arbitrarily small probability of error. The parameter C/Tc is called the critical rate. When this condition is satisfied with the equality sign, the system is said to be signaling at the critical rate. This theorem is also known as ñThe Channel Coding Theoremò (Noisy Coding Theorem). It may be stated in a different form as below:

R Ò C or rs H(S) Ò rc I(X,Y)Max or{ H(S)/Ts} Ò{ I(X,Y)Max/Tc}

ñIf a discrete memoryless source with an alphabet óSô has an entropy H(S) and produces symbols every óT sô seconds; and a discrete memoryless channel has a capacity I(X,Y)Max and is used once every Tc seconds.

A communication channel, is more frequently, described by specifying the source probabilities P(X) & the conditional probabilities P (Y|X) rather than specifying the JPM. The CPM, P (Y|X), is usually refereed to as the ó noise characteristicô of the channel. Therefore unless otherwise specified, we shall understand that the description of the channel, by a matrix or by a óChannel diagramô always refers to CPM, P (Y|X). Thus, in a discrete communication channel with pre-specified noise characteristics (i.e. with a given transition probability matrix, P (Y|X)) the rate of information transmission depends on the source that drives the channel. Then, the maximum rate corresponds to a proper matching of the source and the channel. This ideal characterization of the source depends in turn on the transition probability characteristics of the given channel.

2.3.6 ERROR CORRECTING CODE :

If the code word with n-bits is to be transmitted in no more time than is required for the transmission of the k-information bits and if Űb and Űc are the bit durations in the encoded and coded words.

There are mainly two types of error control coding schemes ï Block codes and convolutional codes, which can take care of either type of errors mentioned above.

In a block code, the information sequence is divided into message blocks of k bits each, represented by a binary k-tuple, u = (u1, u2 é . uk) and each block is called a message. The symbol u, here, is used to denote a k ï bit message rather than the entire information sequence . The encoder then transforms u into an n-tuple v = (v1, v2 é . vn). Here v represents an encoded block rather than the entire encoded sequence. The blocks are independent of each other.

The encoder of a convolutional code also accepts k-bit blocks of the information sequence u and produces an n-symbol block v. Here u and v are used to denote sequences of blocks rather than a single block. Further each encoded block depends not only on the present k-bit message block but also on m-pervious blocks. Hence the encoder has a memory of order ómô. Since the encoder has memory, implementation requires sequential logic circuits.

2.3.7 REDUNDANCY AND EFFICIENCY

A redundant source is one that produces ódependentô symbols. (Example: The Markov source). Such a source generates symbols that are not absolutely essential to convey information. As an illustration, let us consider the English language. It is really unnecessary to write ñUò following the letter ñQò. The redundancy in English text is e stimated to be 50%(refer J Das etal, Sham Shanmugam, Reza, Abramson, Hancock for detailed discussion.) This implies that, in the long run, half the symbols are unnecessary! For example, consider the following sentence.

However, we want redundancy. Without this redundancy abbreviations would be impossible and any two dimensional array of letters would form a crossword puzzle! We want redundancy even in communications to facilitate error detection and error correction. Then how to measure redundancy? Recall that for a Markov source, $H(S) < H(\quad S)$, where S is an ad- joint, zero memory source. That is, when dependence creeps in, the entropy of the source will be reduced and this can be used as a measure indeed!

ñ The redundancy of a sequence of symbols is measured by noting the amount by which the entropy has been reducedò.

When there is no inter symbol influence the entropy at the receiver would be $H(X)$ for any given set of messages $\{X\}$ and that when inter symbol influence occurs the entropy would be

H (Y|X). The difference [H(X) ï H (Y|X)] is the net reduction in entropy and is called ñ Absolute Redundancy

2.3.8 PARITY CHEK CODES

Definition: A bit string has odd parity if the number of 1s in the string is odd. A bit string has even parity if the number of 1s in the string is even.

Recall: 0 is an even number.

Example:

01100, 000, and 11001001 have even parity.

1000011, 1, and 00010 have odd parity.

Assume we are transmitting blocks of

k bits.

ÅA block w of length k is encoded as wa, where the value of the parity bit

a is chosen so

that wa has even parity.

ï Example: If w = 10110, we send wa = 101101, which has even parity.

ÅIf there are a positive, even number of bit flip errors in transmission, the receiver gets a bit string with even parity, and the error(s) go undetected.

ÅIf there are an odd number of bit flip errors in transmission, the receiver gets a bit string with odd parity, indicating an error occurred. The receiver requests retransmission.

Single parity check code

Any time a block of length n (with parity bit) contains an even number of bit errors, the error cannot be detected. Let p be the probability of an error in a single bit. The probability of 2 bit flips in the block is:

The number of ways to choose 2 bits from n bits, times p 2, the probability of those bits being errors, timesnī 2, the probability of the remaining n ï 2\bits being correct.

The probability of an undetected error is:

For bit strings of length n = 32 and p = 0.001, the probability of an undetectable error is approximately 0.0005.

Rectangular code

Block of bits is organized in rows and columns, say an m × n matrix.

ÅThe parity bit of each row is calculated, and appended to the row before it is transmitted.

ÅThe parity of each column is calculated, and the parity bit of the entire matrix is computed - these are also transmitted to the receiver.

Åm + n + 1 parity bits are computed.

ÅA total of mn + m + n + 1 bits are sent to the receiver.

2.4 LINEAR BLOCK CODES :-

We assume that the output of an information source is Na sequence of binary digits ñ0ò or ñ1ò This binary information sequence is segmented into message block of fixed length, denoted by **u.** Each message block consists of k information digits. There are a total of 2k distinct message. The encoder transforms each input message **u** into a binary n-tuple **v** with n > k **.**This n-tuple **v** is referred to as the code word (or code vector) of the message **u**. There are distinct 2k code words.

This set of 2k code words is called a block code. For a block code to be useful, there should be a one-to-one correspondence between a message **u** and its code word **v.** A desirable structure for a block code to possess is the linearity. With this structure, the encoding complexity will be greatly reduced.

Definition A block code of length n and 2k code word is called a linear (n, k) code iff its 2k code words form a k-dimensional subspace of the vector space of all the n-tuple over the field GF(2). In fact, a binary block code is linear iff the module-2 sum of two code word is also a code word **0** must be code word. Since an (n, k) linear code C is a k-dimensional subspace of the vector space Vn of all the binary ntuple, it is possible to find k linearly independent code word, **g**0 , **g**1 ,é , **g**k-1 in C

2.4.1 VECTOR SPACE AND SUBSPACE

(n,k) code maps a length k-tuples to length n-tuples.

ÅThe set of all binary n-tuples form a vector space Vn That binary field has 2 operations, multiplication and addition (ex-or).

ÅA subset S of V is called a subspace of V iff

ï The all zeros vector is in S AND

ï The sum of any 2 vectors in S is also in S

In Vn the 2n tuples can be represented as points in the space

ÅSome of these points are in S

ÅThere are 2 contradicting objectives

ï Code efficiency means we want to pack Vn with

elements of S .

ï Error detection means that we want the elements of S to

be as far away as possible from each other.

Could be done by using table lookup

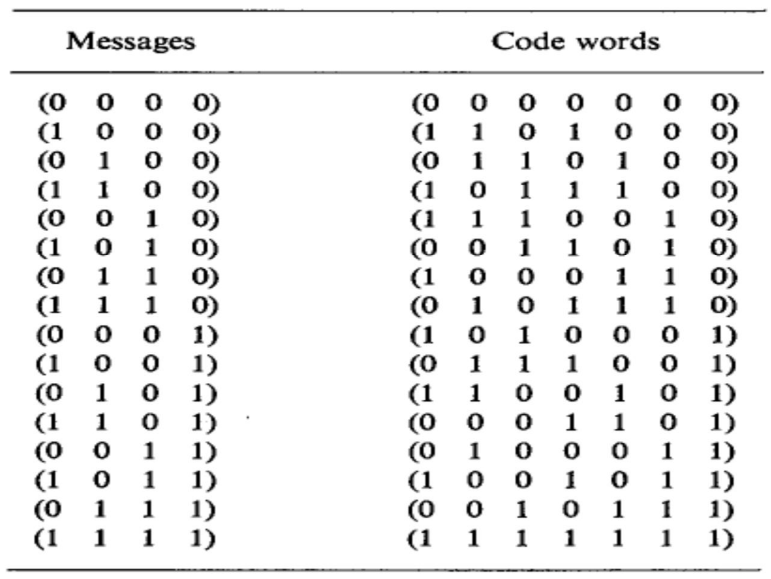
ÅThe size of the table lookup is k2k bits

ÅToo large for large k

EXAMPLE

In fact, a binary block code is linear if and only if the modulo-2 sum of two code words is also a code word. The block code given in Table 1 is a (7, 4) linear code. One can easily check that the sum of any two code words in this code is also a code word.

TABLE 3.1 LINEAR BLOCK CODE WITH $k = 4$ AND $n = 7$



Messages	Code words
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

2.4.2 GENERATOR MATRIX

Since the codewords form a k-dimensional subspace of the n-dimensional space, we can use the basis of the subspace to generate any element in the subspace.

ÅIf these basis are (linearly independent ntuples) V1, V2, . . . ,Vk any code word can be represented as

ÅU=m1V1 + m2V2 + . . . mkVk

$$\mathbf{G} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_k \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & & \cdots & \\ v_{k1} & v_{k2} & \cdots & v_{nn} \end{bmatrix}, \quad \mathbf{m} = m_1, m_2, \cdots, m_k$$

$\mathbf{U} = \mathbf{mG}$

For example, in the previous table

$$\mathbf{G} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$U_4 = [1 \quad 1 \quad 0] \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{bmatrix} = 1 \cdot V_1 + 1 \cdot V_2 + 1 \cdot V_3$$

$$= 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0$$

2.4.3 SYSTEMATIC LINEAR BLOCK CODE :-

A systematic linear block code is a mapping from a k-dimensional space to ndimensional space such that the k-bit message is a part of the n-bit codeword.

$$G = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,n-k} & 1 & 0 & \cdots & 0 \\ p_{21} & p_{22} & \cdots & p_{2,n-k} & 0 & 1 & & 0 \\ \vdots & \vdots & \cdots & & & & & \\ p_{k1} & p_{k2} & \cdots & p_{k,n-k} & 0 & 0 & & 1 \end{bmatrix}$$

$$u_1, u_2, \cdots, u_n = [m_1, m_2, \cdots, m_k] \times \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,n-k} & 1 & 0 & \cdots & 0 \\ p_{21} & p_{22} & \cdots & p_{2,n-k} & 0 & 1 & & 0 \\ \vdots & \vdots & \cdots & & & & & \\ p_{k1} & p_{k2} & \cdots & p_{k,n-k} & 0 & 0 & & 1 \end{bmatrix}$$

$$U = \underbrace{\alpha_1, \alpha_2, \cdots, \alpha_{n-k}}_{parity}, \underbrace{m_1, m_2, \cdots, m_k}_{message}$$

A desirable property for a linear block code is the systematic structure of the code words as shown in Fig.where a code word is divided into two parts

The message part consists of k information digits The redundant checking part consists of n ī k parity-check digits.

A linear block code with this structure is referred to as a linear systematic block code

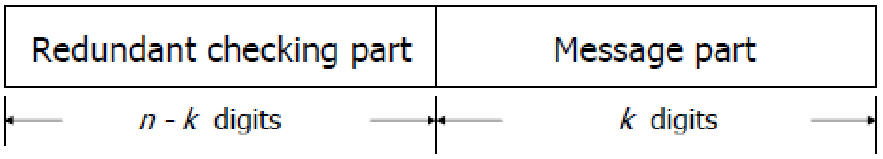


Fig. 3.1 Systematic format of a code word

A linear systematic (n, k) code is completely specified by a k × n matrix G of the following form

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{array}{cccccc|ccccccc} \multicolumn{6}{c}{\text{P matrix}} & \multicolumn{7}{c}{k \times k \text{ identity matrix}} \\ p_{00} & p_{01} & \cdot & \cdot & \cdot & p_{0,n-k-1} & 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ p_{10} & p_{11} & \cdot & \cdot & \cdot & p_{1,n-k-1} & 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 \\ p_{20} & p_{21} & \cdot & \cdot & \cdot & p_{2,n-k-1} & 0 & 0 & 1 & \cdot & \cdot & \cdot & 0 \\ & & & & & & & & & & & & \\ & & & & & & & & & & & & \\ & & & & & & & & & & & & \\ p_{k-1,0} & p_{k-1,1} & \cdot & \cdot & \cdot & p_{k-1,n-k-1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

where $p_{ij} = 0$ or 1

Let **u** = (u0, u1, é , uk-1) be the message to be encoded The corresponding code word is It follows from (3.4) & (3.5) that the components of **v** are vn-k+i = ui for 0 Òi < k

And vj = u0 p0j + u1 p1j + ⋯ + uk-1 pk-1, j for 0 Òj < n-k . shows that the rightmost k digits of a code word **v** are identical to the information digits u0, u1,é uk-1 to be encoded Equation (3.6b) shown that the leftmost n ï k redundant digits are linear sums of the information digits The n ï k equations are called parity-check equations of the code.

2.4.4 PARITY CHECK MATRIX

For any k × n matrix G with k linearly independent rows, there exists an (n-k) ×n matrix H with n-k linearly independent rows such that any vector in the row space of G is orthogonal to the rows of H and any vector that is orthogonal to the rows of H is in the row space of G. An n-tuple v is a code word in the code generated by G if and only if v ÅHT = 0

This matrix H is called a parity-check matrix of the code.The 2n-k linear combinations of the rows of matrix H form an (n, n ï k) linear code Cd.This code is the null space of the (n, k) linear code C generated by matrix G Cd is called the dual code of C .If the generator matrix of an (n,k) linear code is in the systematic form of (3.4), the parity-check matrix may take the following form :

$$\mathbf{H} = \left[\mathbf{I}_{n-k} \quad \mathbf{P}^T\right]$$

$$= \begin{bmatrix} 1 & 0 & 0 & . & . & . & 0 & p_{00} & p_{10} & . & . & . & p_{k-1,0} \\ 0 & 1 & 0 & . & . & . & 0 & p_{01} & p_{11} & . & . & . & p_{k-1,1} \\ 0 & 0 & 1 & . & . & . & 0 & p_{02} & p_{12} & . & . & . & p_{k-1,2} \\ . & & & & & & & & & & & & \\ . & & & & & & & & & & & & \\ . & & & & & & & & & & & & \\ 0 & 0 & 0 & . & . & . & 1 & p_{0,n-k-1} & p_{1,n-k-1} & . & . & . & p_{k-1,n-k-1} \end{bmatrix}$$

Let hj be the jth row of H for 0 Òi < k and 0 Òj < n ï k .This implies that G ÅHT = 0

Let u = (u0, u1, é , uk-1) be the message to be encoded In systematic form the corresponding code word would be v = (v0, v1, é , vn-k-1, u0,u1, é , uk-1).Using the fact that v ÅHT = 0, we obtain vj + u0 p0j + u1 p1j + ⋯ + uk-1 pk-1,j = 0

Rearranging the equation of we obtain the same parity-check equations of An (n, k) linear code is completely specified by its parity check matrix.

2.4.5 SYNDROME TESTING AND ERROR CORRECTION AND DECODER IMPLEMENTAION

Suppose the code vector v= (v0, v1, v2 é v n-1) is transmitted over a noisy channel. Hence the received vector may be a corrupted version of the transmitted code vector. Let the received code vector be r = (r0, r1, r 2é r n-1). The received vector may not be anyone of the 2k valid code vectors. The function of the decoder is to determine the transmitted code vector based on the received vector. The decoder, as in the case of linear block codes, first computes the syndrome to check whether or not the received code vector is a valid code vector. In the case of cyclic codes, if the syndrome is zero, then the received code word polynomial must be divisible by the generator polynomial. If the syndrome is non-zero, the received word contains transmission errors and needs error correction. Let the received code vector be represented by the polynomial

$$R(X) = r_0+r_1X+r_2X^2+\ldots+r_{n-1}X^{n-1}$$

Let *A(X)* be the quotient and *S(X)* be the remainder polynomials resulting from the division of *R(X)* by *g(X)* i.e.

$$\frac{R(X)}{g(X)} = A(X) + \frac{S(X)}{g(X)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (7.21)$$

The remainder *S(X)* is a polynomial of degree *(n-k-1)* or less. It is called the "Syndrome polynomial". If *E(X)* is the polynomial representing the error pattern caused by the channel, then we have:

$$R(X) = V(X) + E(X) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (7.22)$$

And it follows as $V(X) = U(X)\, g(X)$, that:

$$E(X) = [A(X) + U(X)]\, g(X) + S(X) \qquad \ldots\ldots$$

(7.23)

That is, the syndrome of R(X) is equal to the remainder resulting from dividing the error pattern by the generator polynomial; and the syndrome contains information about the error pattern,

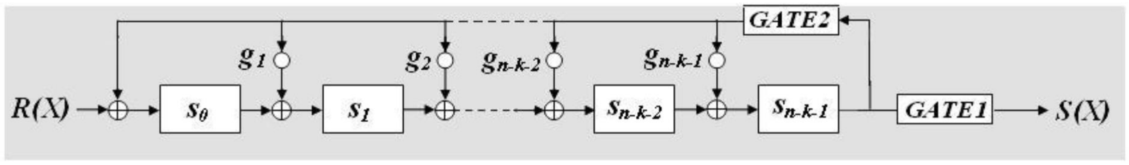


Fig 7.10 Syndrome calculator using (n-k) Shift registers

The syndrome calculations are carried out as below:

1 The register is first initialized. With GATE 2 -ON and GATE1- OFF, the received vector is entered into the register

2 After the entire received vector is shifted into the register, the contents of the register will be the syndrome, which can be shifted out of the register by turning GATE-1 ON and GATE-2 OFF. The circuit is ready for processing next received vector.

Cyclic codes are extremely well suited for 'error detection' .They can be designed to detect many combinations of likely errors and implementation of error-detecting and error correcting circuits is practical and simple. Error detection can be achieved by employing (or adding) an additional R-S flip-flop to the syndrome calculator. If the syndrome is nonzero, the flip-flop sets and provides an indication of error. Because of the ease of implementation, virtually all error detecting codes are invariably 'cyclic codes'. If we are interested in error correction, then the decoder must be capable of determining the error pattern E(X) from the syndrome S(X) and add it to R(X) to determine the transmitted V(X). The following scheme shown in Fig 7.11 may be employed for the purpose. The error correction procedure consists of the following steps:

Step1. Received data is shifted into the buffer register and syndrome registers with switches SIN closed and SOUT open and error correction is performed with SIN open and SOUT closed.

Step2. After the syndrome for the received code word is calculated and placed in the syndrome register, the contents are read into the error detector. The detector is a combinatorial circuit designed to output a ó1ô if and only if the syndrome corresponds to a correctable error pattern with an error at the highest order position Xn-l. That is, if the detector output is a '1' then the received digit at the right most stage of the buffer register is assumed to be in error and will be corrected. If the detector output is '0' then the received digit at the right most stage of the buffer is assumed to be correct. Thus the detector output is the estimate error value for the digit coming out of the buffer register.

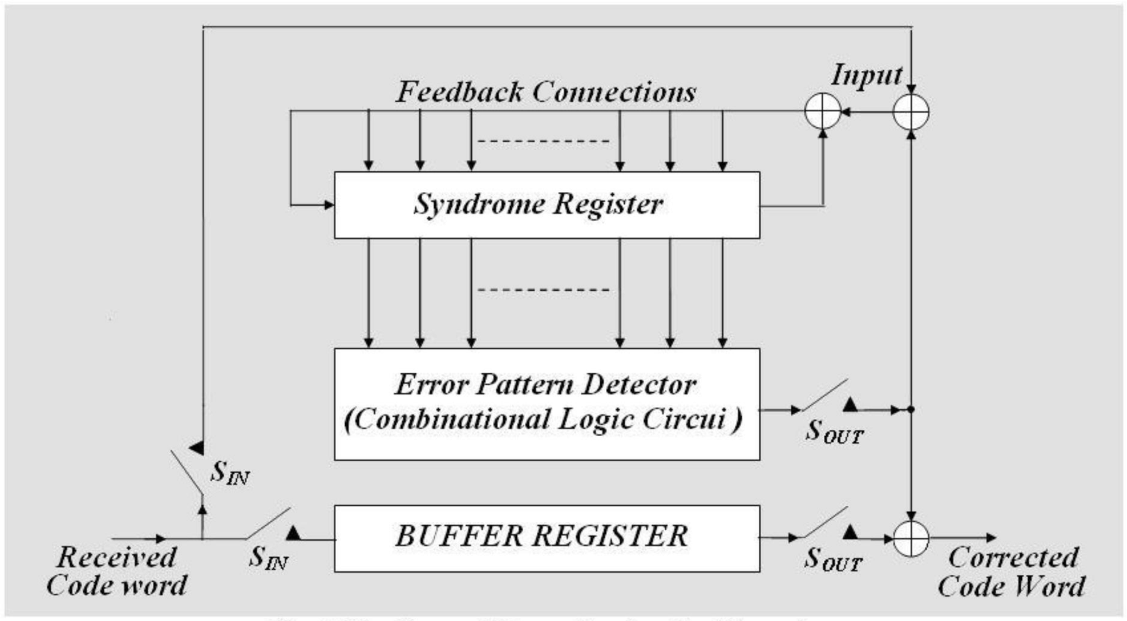


Fig 7.11 General Decoder for Cyclic codes

Step3. In the third step, the first received digit in the syndrome register is shifted right once. If the first received digit is in error, the detector output will be '1' which is used for error correction. The output of the detector is also fed to the syndrome register to modify the syndrome. This results in a new syndrome corresponding to the óaltered óreceived code word shifted to the right by one place.

Step4. The new syndrome is now used to check and correct the second received digit, which is now at the right most position, is an erroneous digit. If so, it is corrected, a new syndrome is calculated as in step-3 and the procedure is repeated.

Step5. The decoder operates on the received data digit by digit until the entire received code word is shifted out of the buffer.

At the end of the decoding operation, that is, after the received code word is shifted out of the buffer, all those errors corresponding to correctable error patterns will have been corrected, and the syndrome register will contain all zeros. If the syndrome register does not contain all zeros, this means that an un-correctable error pattern has been detected. The decoding schemes described in can be used for any cyclic code. However, the practicality depends on the complexity of the combinational logic circuits of the error detector.

2.4.6 WEIGHT AND DISTANCE OF BINARY VECTORS

The distance between two vectors x and y of thesame length over a finite alphabet Ɇ, denoted æ(x, y), is defined as the number of positions at whichthe two strings differ, i.e., æ(x, y) = |{i|xi 6= yi}|. The fractional Hamming distance or relative distance between x, y ∈ Ɇ.

The weight of a vectors x over alphabet Ɇ is defined as the number of non-zero symbols in the string. More formally, the Hamming weight of a string wt(x) = |{i|xi 6= 0}|. Note that wt(x ī y) = æ(x, y).

2.4.7 MINIMUM DISTANCE OF LINEAR CODE

A general code might have no structure and not admit any representation other than listing the entire codebook. We now focus on an important subclass of codes with additional structure called linear codes. Many of the important and widely used codes are linear. Linear codes are defined over alphabets Ɇ which are finite fields. Throughout, we will denote by Fq the finite field with q elements, where q is a prime power. (Later on in the course, it is valuable to have a good grasp of the basic properties of finite fields and field extensions. For now, we can safely think of q as a prime, in which case Fq is just {0, 1, . . . , q ī 1} with addition and multiplication defined modulo q.) Definition 7 (Linear code) If Ɇ is a field and C ⊂ Ɇn is a subspace of Ɇ n then C is said to be a linear code. As C is a subspace, there exists a basis c1, c2, . . . , ck where k is the dimension of the subspace. Any codeword can be expressed as the linear combination of these basis vectors. We can write these vectors in matrix form as the columns of a n×k matrix. Such a matrix is called a generator matrix.

The Hamming weight w(v) of a binary n-tuple code word v is the number of nonzero components in the code word. The Hamming distance d(v, w) between two q-ary (n, k) code vectors v and w is the number of places in which they differ. The Hamming distance d(v, w) is the Hamming weight w(v+w). The minimum weight of a linear block code C is the minimum weight of its nonzero code words. The minimum distance of a linear block code is equal to the minimum weight of its nonzero code words. Let C be an (n, k) linear code with parity-check matrix H. For each code vector of Hamming weight l, there exists l columns of H such that the vector sum of these l columns is equal to the zero vector. Conversely, if there exist l columns of H whose vector sum is the zero vector, there exists a code vector of

Hamming weight l in C. The minimum weight wmin (or minimum distance dmin) of C is equal to the smallest number of columns of H that sum to 0.

Definition (Hamming distance) The Hamming distance between two strings x and y of thesame length over a finite alphabet Ɇ, denoted æ(x, y), is defined as the number of positions at whichthe two strings differ, i.e., æ(x, y) = |{i|xi 6= yi}|. The fractional Hamming distance or relative distance between x, y ∈ Ɇ.

Definition (Hamming weight) The Hamming weight of a string x over alphabet Ɇ is definedas the number of non-zero symbols in the string. More formally, the Hamming weight of a string wt(x) = |{i|xi 6= 0}|. Note that wt(x ī y) = æ(x, y).

Definition:An error correcting code or block code C of length n over a finite alphabet Ɇ is a subset of Ɇn. The elements of C are called the codewords in C, and the collection of all codewords is sometimes called a codebook.The alphabet of C is Ɇ, and if |Ɇ| = q, we say that C is a q-ary code. When q = 2, we say that Cis a binary code. The length n of the codewords of C is called the block length of C.Associated with a code is also an encoding map E which maps the message set M, identified insome canonical way with {1, 2, . . . , |C|} say, to codewords belonging to Ɇn. The code is then theimage of the encoding map.

Definition 5(Distance) The minimum distance, or simply distance, of a code C, denoted æ(C), is defined to be the minimum Hamming distance between two distinct codewords of C. That is,æ(C) = min c1,c2∈C c16=c2 æ(c1, c2) .In particular, for every pair of distinct codewords in C the Hamming distance between them is atleast æ(C).The relative distance of C, denoted ŭ(C), is the normalized quantity æ(C) n , where n is the block length of C. Thus, any two codewords of C differ in at least a fraction ŭ(C) of positions.

2.4.8 ERROR DETECTION AND CORRECTION

The random-error-detecting capability of a block code is dmin -1, that is it can detect any error pattern with dmin -1 or fewer error digits guaranteed. There 2k-1 undetectable error patterns.There are 2n-2k detectable error patterns. Let Ai be the number of code vectors of weight i in C. The number s A0, A1, _, An are called the weight distribution of C.where p is the transition probablity of a BSC. 3.5 Error-Correcting Capabilities of A Block The random-error-correcting capability of block code is t, that is it guarantees correcting all the error patterns of t or fewer digits.where [] denotes the largest integer. A t-error-correcting (n, k) block code is capable of correcting a total of 2n-k error patterns.

Minimum distance criteria:

Given a block code C, its minimum Hamming distance, d min, is defined as the minimum Hamming distance among all possible distinct pairs of code In order to compute the minimum distance d min of a block code C, in accordance with above distances between distinct pairs of code words are needed. The following table-1 shows the hamming distance between different code words of (8 2 5) code. Here minimum weight is equal to minimum distance of code C.

Triangle inequality:

It states that the code C is capable of correcting all error patterns of t or fewer errors. Let v and r be the transmitted and received vectors respectively and let w be any other code vector in C then The Hamming distances among v, r and w satisfy the triangle inequality:

d(v, r) + d(w, r) Ód(v, w) (13)

For a given block code, designed with generator matrix of equation 14, considering v = [10 11 1010], r = [00 11 1010] and w = [01 01 1101], d(v, r) = 1, d(w, r) = 5 and d(v, w) = 6.

Here d(v, r) + d(w, r) = d(v, w). Thus it satisfies the triangle inequality.

Weight distribution W(C):

The Weight distribution W(C) of an error correcting code C, is defined as the set of n + 1integers W(C) = {Ai , 0 Òi Òn} such that there are Ai code words of Hamming weight i in C, fori = 0, 1, . . . , n.

Asymptotic Coding Gain (Ga):

It is the gain that would be delivered if vanishingly small decoded error rates were required It is given by Ga=10 log[R(t+1)] Or Ga=10 log[R*dmin]

If R = Coding gain=¼, t=2, dmin=5,

then Ga= 10 log [3/4] = -1.249 d

Or Ga=10 log[Rd] =10 log [5/4] = 0.969 dB

Thus asymptotic coding gain will be between -1.249 to 0.969 dB.

2.5 CYCLIC CODES:-

2.5.1 ALGEBRIC STRUCTURE:-

Among the _rst codes used practically were the cyclic codes which were generated
using shift registers. It was quickly noticed by Prange that the class of cyclic codes has a rich algebraic structure, the _rst indication that algebra would be a valuable tool in code design.
The linear code C of length n is a cyclic code if it is invariant under a cyclic cyclic code

shift:

c = (c0; c1; c2 : : : ; cn2; cn1) 2 C

if and only if

~c = (cn1; c0; c1 : : : ; cn3; cn2) 2 C :

As C is invariant under this single right cyclic shift, by iteration it is invariant under any number of right cyclic shifts. As a single left cyclic shift is the same as n1 right cyclic shifts, C is also invariant under a single left cyclic shift and hence all left cyclic shifts. Therefore the linear code C is cyclic precisely when it is invariant under all cyclic shifts. There are some obvious examples of cyclic codes. The 0-code is certainly cyclic as is Fn. Less trivially, repetition codes are cyclic. The binary parity check code is also cyclic, and this goes over to the sum-0 codes over any _eld. Notice that this shift invariance criterion does not depend at all upon the code being linear. It is possible to nonlinear cyclic codes, but that is rarely done. The history of cyclic codes as shift register codes and the mathematical structure theory of cyclic codes both suggest the study of cyclic invariance in the context of linear codes.

Cyclic codes form an important subclass of linear codes. These codes are attractive for two reasons: Encoding and syndrome computation can be implemented easily by employing shift registers with feedback connections(or linear sequential circuits). Because they have considerable inherent algebraic structure, it is possible to find various practical methods for decoding them. Cyclic codes were first studied by Prange in 1957.

If the n-tuple v = (v0, v1,é , vn-1) are cyclic shifted one place to the right, we obtain another n-tuple v(1) = (vn-1, v0, v1,é , vn-2) which is called a cyclic shift of v .If the v are cyclically shifted i places to the right, we have v(i) = (vnï i, vnï i+1,é , vn-1, v0, v1, é , vn-i-1) Cyclically shifting v i places to the right is equivalent to cyclically shifting v (n ï i) place to the left

Definition An (n, k) linear code C is called a cyclic code if every cyclic shift of a code vector in C is also a code vector in C The (7, 4) linear code given in Table is a cyclic code

To develop the algebraic properties of a cyclic code, we treat the components of a code vector v = (v0, v1,é , vn-1) as the coefficients of a polynomial as follows:

v(X) = v0 + v1X + v2X2 + ··· + vn-1Xn-1 If vn-1 Í 0, the degree of v(X) is n ï 1

If vn-1 = 0, the degree of v(X) is less than n ï 1.The correspondence between the vector v and the polynomial v(X) is one-to-one

2.5.2 BINARY CYCLIC CODE PROPERTIES:-

"Binary cyclic codesò form a sub class of linear block codes. Majority of important linear block codes that are known to-date are either cyclic codes or closely related to cyclic codes. Cyclic codes are attractive for two reasons: First, encoding and syndrome calculations can be easily implemented using simple shift registers with feed back connections. Second, they posses well defined mathematical structure that permits the design of higher-order error correcting codes.

A binary code is said to be "cyclic" if it satisfies:

1. Linearity property ï sum of two code words is also a code word.
2. Cyclic property ï Any lateral shift of a code word is also a code word.

The second property can be easily understood from Fig, Instead of writing the code as a row vector, we have represented it along a circle. The direction of traverse may be either clockwise or counter clockwise (right shift or left shift).

For example, if we move in a counter clockwise direction then starting at óAô the code word is 110001100 while if we start at B it would be 011001100. Clearly, the two code words are related in that one is obtained from the other by a cyclic shift.

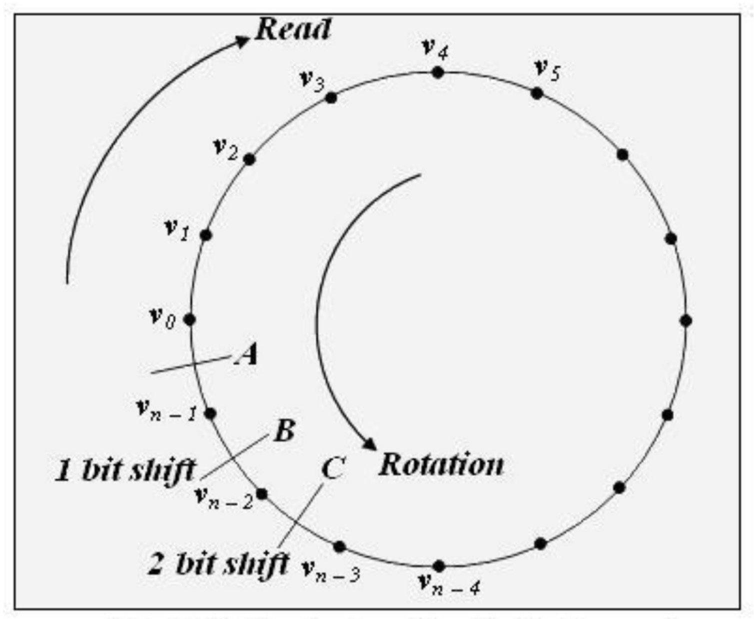


Fig 7.1 Illustrating the Cyclic Property.

2.5.3 ENCODING IN SYSTEMATIC FORM:-

Let us assume a systematic format for the cyclic code as below:

v = (p0, p1, p2 é p n-k-1, u0, u1, u2é u k-1)

The code polynomial in the assumed systematic format becomes:

V(X) = p0 + p1X + p2X2 + é n-k-1Xn-k-1 +u0Xn-k +

+p u1Xnk+1+é +u k-1Xn-1

= P(X) + Xn-kU(X)

V (X) = P (X) +Xn-k U (X) =

Q (X) g (X)

Thus division of Xn-k U (X) by g (X) gives us the quotient polynomial Q (X) and the remainder polynomial P (X). Therefore to obtain the cyclic codes in the systematic form, we determine the remainder polynomial P (X) after dividing Xn-k U (X) by g(X). This division process can be easily achieved by noting that "multiplication by Xn-k amounts to shifting the sequence by (n-k) bits". Specifically in the circuit of Fig 7.5(a), if the input A(X) is applied to the Mod-2 adder after the (n-k) th shift register the result is the division of Xn-k A (X) by B (X).

Accordingly, we have the following scheme to generate systematic cyclic codes. The generator polynomial is written as:

g (X) = 1 +glX+g2X2+g3X3+é +g n-k-1 Xn-k-1 +Xn-k

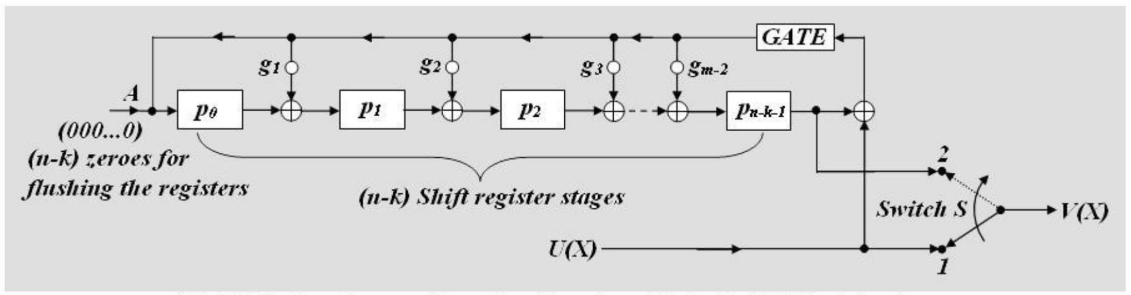


Fig 7.8 Systematic encoding of cyclic codes with (n-k) shift Register stages

1. The switch S is in position 1 to allow transmission of the message bits directly to an out put shift register during the first k-shifts.

2. At the same time the 'GATE' is 'ON' to allow transmission of the message bits into the (n-k) stage encoding shift register

3. After transmission of the kth message bit the GATE is turned OFF and the switch S is moved to position 2.

4. (n-k) zeroes introduced at "A" after step 3, clear the encoding register by moving the parity bits to the output register

5. The total number of shifts is equal to n and the contents of the output register is the code word polynomial V (X) =P (X) + Xn-k U (X).

6. After step-4, the encoder is ready to take up encoding of the next message input

Clearly, the encoder is very much simpler than the encoder of an (n, k) linear block code and the memory requirements are reduced. The following example illustrates the procedure.

2.5.5 DIVIDING CIRCUITS :-

As in the case of multipliers, the division of A (X) by B (X) can be accomplished by using shift registers and Mod-2 adders, as shown in Fig 7.5. In a division circuit, the first co-efficient of the quotient is (an-1 (bm -1) = q1, and q1.B(X) is subtracted from A (X). This subtraction is carried out by the feedback connections shown. This process will continue for the second and subsequent terms. However, remember that these coefficients are binary coefficients. After (n-1) shifts, the entire quotient will appear at the output and the remainder is stored in the shift registers.

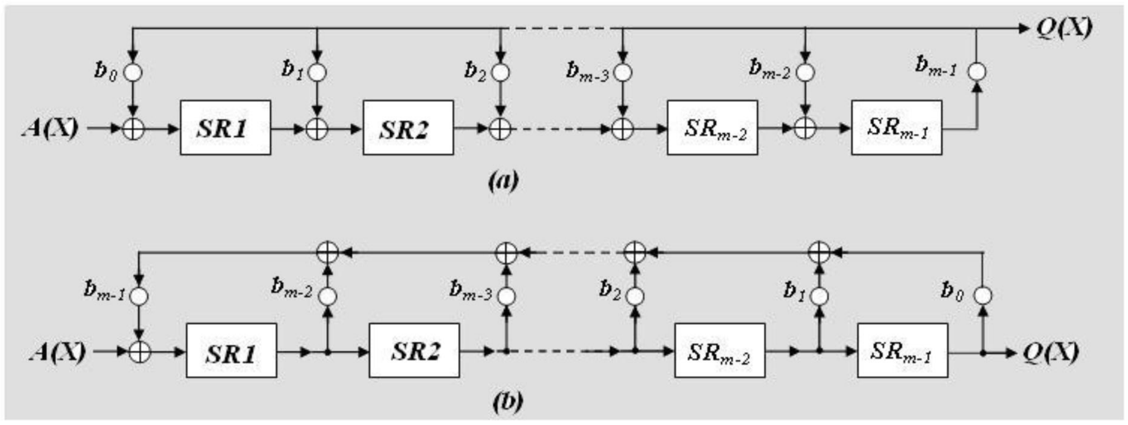


Fig 7.5 Dividing Circuits

It is possible to combine a divider circuit with a multiplier circuit to build a ñcomposite multiplier-divider circuitò which is useful in various encoding circuits. An arrangement to accomplish this is shown in Fig 7.6(a) and an illustration is shown in FIG. We shall understand the operation of one divider circuit through an example. Operation of other circuits can be understood in a similar manner.

2.5.6 SYSTEMATIC ENCODING & ERROR DETECTION USING SHIFT REGISTER :-

An (n, k) cyclic code is specified by the complete set of code polynomials of degree (n-1) and contains a polynomial g(X), of degree (n-k) as a factor, called the "generator polynomial" of the code. This polynomial is equivalent to the generator matrix G, of block codes. Further, it is the only polynomial of minimum degree and is unique.

Theorem "If g(X) is a polynomial of degree (n-k) and is a factor of (Xn +1) then g(X) generates an (n, k) cyclic code in which the code polynomial V(X) for a data vector u = (u0, u1é u k -1) is generated by V(X) = U(X)g(X)

U(X) = u0 + u1 X + u2 X2 + ... + uk-1 Xk-I

is the data polynomial of degree (k-1).

The theorem can be justified by Contradiction: - If there is another polynomial of same degree, then add the two polynomials to get a polynomial of degree < (n, k) (use linearity property and binary arithmetic). Not possible because minimum degree is (n-k). Hence g(X) is unique Clearly, there are 2k code polynomials corresponding to 2k data vectors. The code vectors corresponding to these code polynomials form a linear (n, k) code. We have then, from the theorem

is a polynomial of minimum degree, it follows = gn-k = 1 always and the remaining co-

that g0

efficients may be either' 0' of '1'. Performing the multiplication , we have:

U (X) g(X) = uo g(X) + u1 X g(X) +é +u k-1Xk-1g(X)

Suppose u0=1 and u1=u2= é =u k-1=0. Then from Eq (7.8) it follows g(X) is a code word polynomial of degree (n-k). This is treated as a óbasis code polynomialô (All rows of the G matrix of a block code, being linearly independent, are also valid code vectors and form ó Basis vectorsô of the code). Therefore from cyclic property Xi g(X) is also a code polynomial. Moreover, from the linearity property - a linear combination of code polynomials is also a code polynomial. It follows therefore that any multiple of g(X) as shown in Eq (7.12) is a code polynomial. Conversely, any binary polynomial of degree (n-1) is a code polynomial if and only if it is a multiple of g(X). Construction of encoders and decoders for linear block codes are usually constructed with combinational logic circuits with mod-2 adders. Multiplication of two polynomials A(X) and B(X) and the division of one by the other are realized by using sequential log

Suppose the code vector v= (v0, v1, v2 é v n-1) is transmitted over a noisy channel. Hence the received vector may be a corrupted version of the transmitted code vector. Let the received code vector be r = (r0, r1, r 2é r n-1). The received vector may not be anyone of the 2k valid code vectors. The function of the decoder is to determine the transmitted code vector based on the received vector.

The decoder, as in the case of linear block codes, first computes the syndrome to check whether or not the received code vector is a valid code vector. In the case of cyclic codes, if the syndrome is zero, then the received code word polynomial must be divisible by the generator polynomial. If the syndrome is non-zero, the received word contains transmission errors and needs error correction. Let the received code vector be represented by the polynomial. That is, the syndrome of R(X) is equal to the remainder resulting from dividing the error pattern by the generator polynomial; and the syndrome contains information about the error pattern, which can be used for error correction.

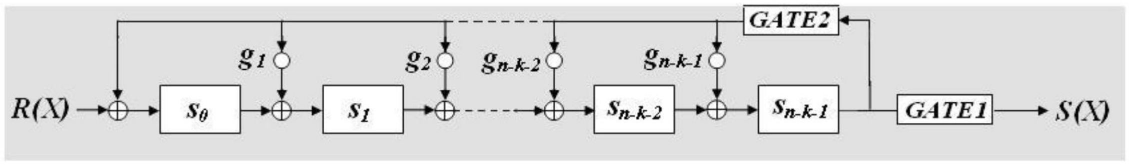


Fig 7.10 Syndrome calculator using (n-k) Shift registers

The syndrome calculations are carried out as below:

1 The register is first initialized. With GATE 2 -ON and GATE1- OFF, the received vector is entered into the register

2 After the entire received vector is shifted into the register, the contents of the register will be the syndrome, which can be shifted out of the register by turning GATE-1 ON and GATE-2 OFF.

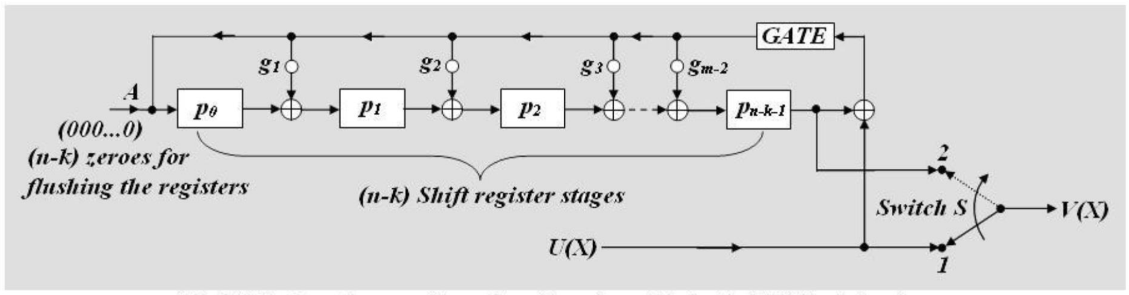


Fig 7.8 Systematic encoding of cyclic codes with (n-k) shift Register stages

1. The switch S is in position 1 to allow transmission of the message bits directly to an out put shift register during the first k-shifts.

2. At the same time the 'GATE' is 'ON' to allow transmission of the message bits into the (n-k) stage encoding shift register

3. After transmission of the kth message bit the GATE is turned OFF and the switch S is moved to position 2.

4. (n-k) zeroes introduced at "A" after step 3, clear the encoding register by moving the parity bits to the output register

5. The total number of shifts is equal to n and the contents of the output register is the code word polynomial V (X) =P (X) + Xn-k U (X).

6. After step-4, the encoder is ready to take up encoding of the next message input

2.6 BLOCK CODES

2.6.1 HAMMING CODE:-

Designed to correct single bit errors.Family of (n, k) block error-correcting codes with parameters:

- Block length: n = 2m ï 1 .
- Number of data bits: k = 2m ï m ï 1.

- Number of check bits: n ï k = m.
- Minimum distance: dmin = 3.
- Single-error-correcting (SEC) code.
- SEC double-error-detecting (SEC-DED) code.
- Encoding: k data bits + (n -k) check bits.
- Decoding: compares received (n -k) bits with calculated (n -k) bits using.
- Resulting (n -k) bits called syndrome word.
- Syndrome range is between 0 and 2(n-k)-1.
- Each bit of syndrome indicates a match (0) or conflict (1) in that bit position.

Hamming code is the first class of linear block codes devised for error correction. The single error correcting (SEC) Hamming codes are characterized by the following parameters.

Code length: n = (2m-1)

Number of Information symbols: k = (2m ï m ï 1)

Number of parity check symbols :(n ï k) = m

Error correcting capability: t = 1, (dmin= 3)

The parity check matrix H of this code consists of all the non-zero m-tuples as its columns. In systematic form, the columns of H are arranged as follows

H = [Q M Im]

Where Im is an identity (unit) matrix of order m   m and Q matrix consists of

(2m-m-1) columns which are the m-tuples of weight 2 or more. As an illustration for k=4 we have from k = 2m ï m ï 1.

m=1 k=0, m=2 k=1, m=3 k=4

Thus we require 3 parity check symbols and the length of the code 23 ï 1 = 7 . This results in the (7, 4) Hamming code.

The parity check matrix for the (7, 4) linear systematic Hamming code is then

p1 p2 m1 p3 m2 m3 m4 p4 m5 m6 m7 m8 m9 m10 m11 p5 m12

Where p1, p2, p3é are the parity digits and m1, m2, m3é are the message digits. For example, let us

consider the non systematic (7, 4) Hamming code.

p1 = 1, 3, 5, 7, 9, 11, 13, 15é

p2 = 2, 3, 6, 7, 10, 11, 14, 15 é

p3 = 4, 5, 6, 7, 12, 13, 14, 15é

It can be verified that (7, 4), (15, 11), (31, 26), (63, 57) are all single error correcting Hamming codes and are regarded quite useful.

An important property of the Hamming codes is that they satisfy the condition of Eq. (6.36) with equality sign, assuming that t=1.This means that Hamming codes are ñ single error correcting binary perfect codesò. This can also be verified from Eq. (6.35)

We may delete any ó l ôcolumns from the parity check matrix H of the Hamming code resulting in the reduction of the dimension of H matrix to m .(2m-l-1).Using this new matrix as the parity check matrix we obtain a ñ shortenedò Hamming code with the following parameters.

Code length: n = 2m-l- k=2m-m-l-1

1 Number of

Information symbols:

Number of parity n ï k = m

check symbols:

Minimum distance: dmin 3

$$H = \begin{bmatrix} 1110 & \vdots & 100 \\ 1101 & \vdots & 010 \\ 1011 & \vdots & 001 \end{bmatrix}$$

The distance ï 4 shortened Hamming codes can be used for correcting all single error patterns while simultaneously detecting all double error patterns. Notice that when single errors occur the syndromes contain odd number of oneôs and for double errors it contains even number of ones. Accordingly the decoding can be accomplished in the following manner.

(1) If s = 0, no error occurred.

(2) If s contains odd number of ones, single error has occurred .The single error pattern pertaining to this syndrome is added to the received code vector for error correction.

(3) If s contains even number of oneôs an uncorrectable error pattern has been detected.

Alternatively the SEC Hamming codes may be made to detect double errors by adding an extra parity check in its (n+1) Th position. Thus (8, 4), (6, 11) etc. codes have dmin = 4 and correct single errors with detection of double errors.

2.6.2 GOLAY CODE:-

Golay code is a (23, 12) perfect binary code that is capable of correcting any combination of three or fewer random errors in a block of 23 bits. It is a perfect code because it satisfies the Hamming bound with the equality sign for t = 3 as:

The code has been used in many practical systems. The generator polynomial for the code is obtained from the relation (X23+1) = (X+ 1) g1(X) g2(X), where:

g1(X) = 1 + X2 + X4 + X5 + X6 + X10 + X11 and g2 (X) = 1 + X + X5 + X6 + X7 + X9 + X11

The encoder can be implemented using shift registers using either g1 (X) or g2 (X) as the divider polynomial. The code has a minimum distance, dmin =7. The extended Golay code, a (924, 12) code has dmin =8. Besides the binary Golay code, there is also a perfect ternary (11, 6) Golay code with dmin = 5.

2.6.3 BCH CODE:-

One of the major considerations in the design of optimum codes is to make the block size n smallest for a given size k of the message block so as to obtain a desirable value of dmin. Or for given code length n and efficiency k/n, one may wish to design codes with largest dmin. That means we are on the look out for the codes that have 'best error correcting capabilities". The BCH codes, as a class, are one of the most important and powerful error-correcting cyclic codes known. The most common BCH codes are characterized as follows. Specifically, for any positive integer m 3, and t < 2m - 1) / 2, there exists a binary BCH code (called 'primitive' BCH code) with the following parameters:

Block length : n = 2m-l

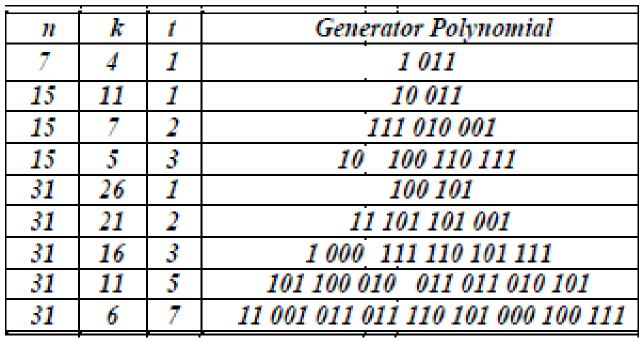
Number of message bits : k n - mt

Minimum distance : dmin 2t + 1

Clearly, BCH codes are "t - error correcting codes". They can detect and correct up toó tô random errors per code word. The Hamming SEC codes can also be described as BCH codes. The BCH codes are best known codes among those which have block lengths of a few hundred or less. The major advantage of these codes lies in the flexibility in the choice of code parameters viz: block length and code rate. The parameters of some useful BCH codes are given below. Also indicated in the table are the generator polynomials for block lengths up to 31.

NOTE: Higher order co-efficients of the generator polynomial are at the left. For example, if we are interested in constructing a (15, 7) BCH code from the table we have (111 010 001) for the co-efficients of the generator polynomial. Hence

g(X) = 1 + X4 + X6 + X7 + X8



n	k	t	Generator Polynomial
7	4	1	1 011
15	11	1	10 011
15	7	2	111 010 001
15	5	3	10 100 110 111
31	26	1	100 101
31	21	2	11 101 101 001
31	16	3	1 000 111 110 101 111
31	11	5	101 100 010 011 011 010 101
31	6	7	11 001 011 011 110 101 000 100 111

For further higher order codes, the reader can refer to Shu Lin and Costello Jr. The alphabet of a BCH code for n = (2m-1) may be represented as the set of elements of an appropriate Galois field, GF(2m) whose primitive element is .The generator polynomial of the t-error correcting BCH code is the least common multiple (LCM) of Ml(X), M2(X),é M2t(X), where Mi(X) is the minimum polynomial of i, i = 1, 2é 2t . There are several iterative procedures available for decoding of BCH codes. Majority of them can be programmed on a general purpose digital computer, which in many practical applications form an integral part of data communication networks. Clearly, in such systems software implementation of the algorithms has several advantages over hardware implementation

MODULE-III

3.1 Covolutional Encoding:

Convolutional codes are commonly described using two parameters: the code rate and the constraint length. The code rate, k/n, is expressed as a ratio of the number of bits into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. The constraint length parameter, K, denotes the "length" of the convolutional encoder, i.e. how many k-bit stages are available to feed the combinatorial logic that produces the output symbols. Closely related to K is the parameter m, which indicates how many encoder cycles an input bit is retained and used for encoding after it first appears at the input to the convolutional encoder. The m parameter can be thought of as the memory length of the encoder. Convolutional codes are widely used as channel codes in practical communication systems for error correction. The encoded bits depend on the current k input bits and a few past input bits. The main decoding strategy for convolutional codes is based on the widely used Viterbi algorithm. As a result of the wide acceptance of convolutional codes, there have been several approaches to modify and extend this basic coding scheme. Trellis coded modulation (TCM) and turbo codes are two such examples. In TCM, redundancy is added by combining coding and modulation into a single operation. This is achieved without any reduction in data rate or expansion in bandwidth as required by only error correcting coding schemes. A simple convolutional encoder is shown in Fig. 3.1.1. The information bits are fed in small groups of k-bits at a time to a shift register. The output encoded bits are obtained by modulo-2 addition (EXCLUSIVE-OR operation) of the input information bits and the contents of the shift registers which are a few previous information bits. If the encoder generates a group of ónô encoded bits per group of ókô information bits, the code rate R is commonly defined as R = k/n. In Fig. 3.1.1, k = 1 and n =

2. The number, K of elements in the shift register which decides for how many codewords one information bit will affect the encoder output, is known as the constraint length of the code.

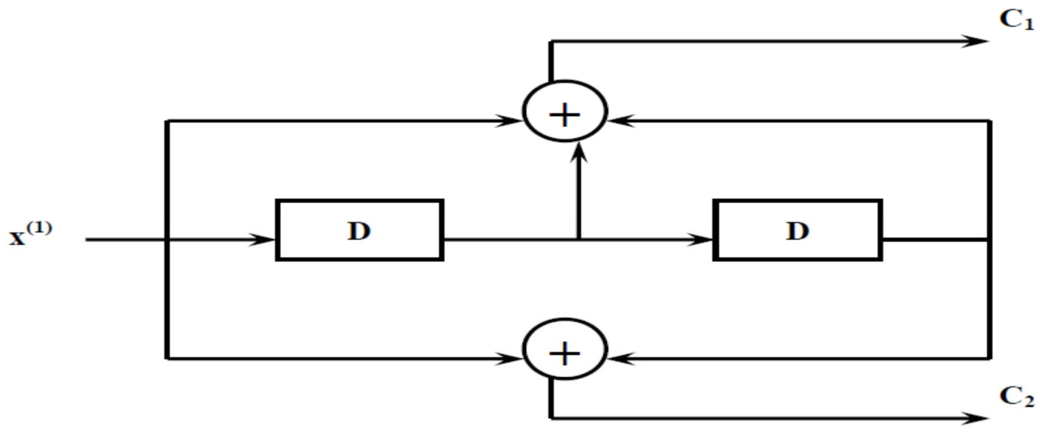


Fig. 3.1.1 A convolutional encoder with k=1, n=2 and r=1/2

3.2 Convolutional Encoder Representation:

The operation of a convolutional encoder can be explained in several but equivalent ways such as, by a) state diagram representation, b) tree diagram representation and c) trellis diagram representation.

3.2.1 State Diagram Representation :

A convolutional encoder may be defined as a finite state machine. Contents of the rightmost (K-1) shift register stages define the states of the encoder. So, the encoder in **Fig. 3.1.1** has four states. The transition of an encoder from one state to another, as caused by input bits, is depicted in the state diagram. **Fig. 3.1.2** shows the state diagram of the encoder in **Fig. 3.1.1**. A new input bit causes a transition from one state to another. The path information between the states, denoted as b/c1c2, represents input information bit ‘bô and the corresponding output bits (c1c2). Again, it is not difficult to verify from the state diagram that an input information sequence b = (1011) generates an encoded sequence **c** = (11, 10, 00, 01).

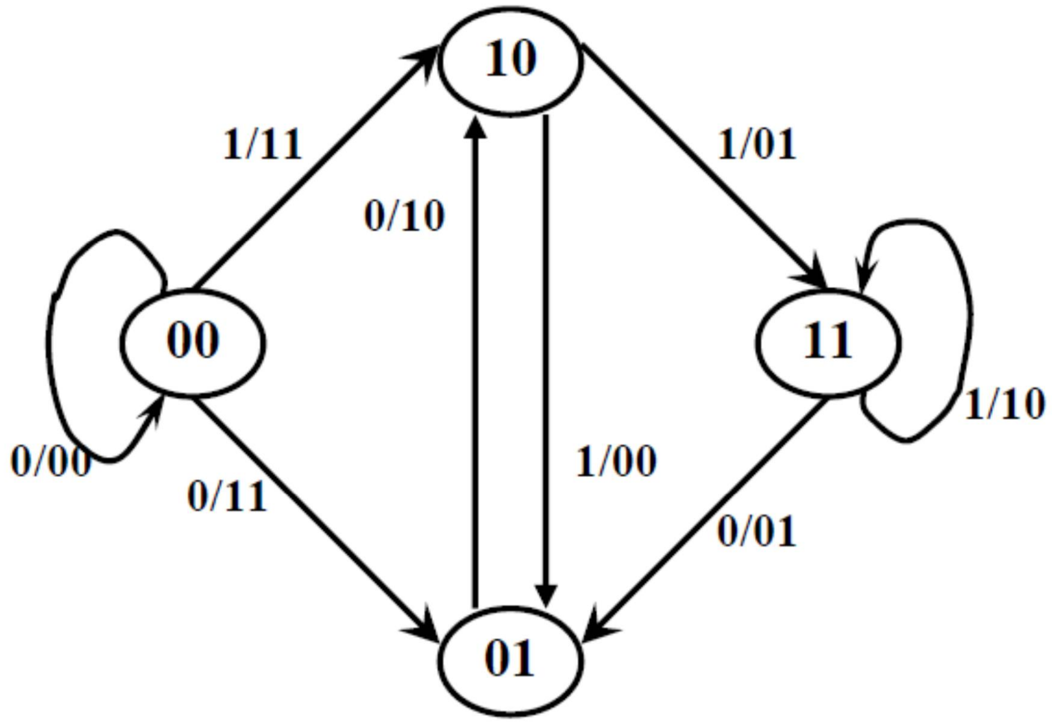


Fig.3.1.2 State diagram representation for the encoder in Fig. 3.1.1

3.2.2 Tree Diagram Representation :

The tree diagram representation shows all possible information and encoded sequences for the convolutional encoder. **Fig. 3.2.3** shows the tree diagram for the encoder in **Fig. 3.1.1**. The encoded bits are labeled on the branches of the tree. Given an input sequence, the encoded sequence can be directly read from the tree. As an example, an input sequence (1011) results in the encoded sequence (11, 10, 00, 01).

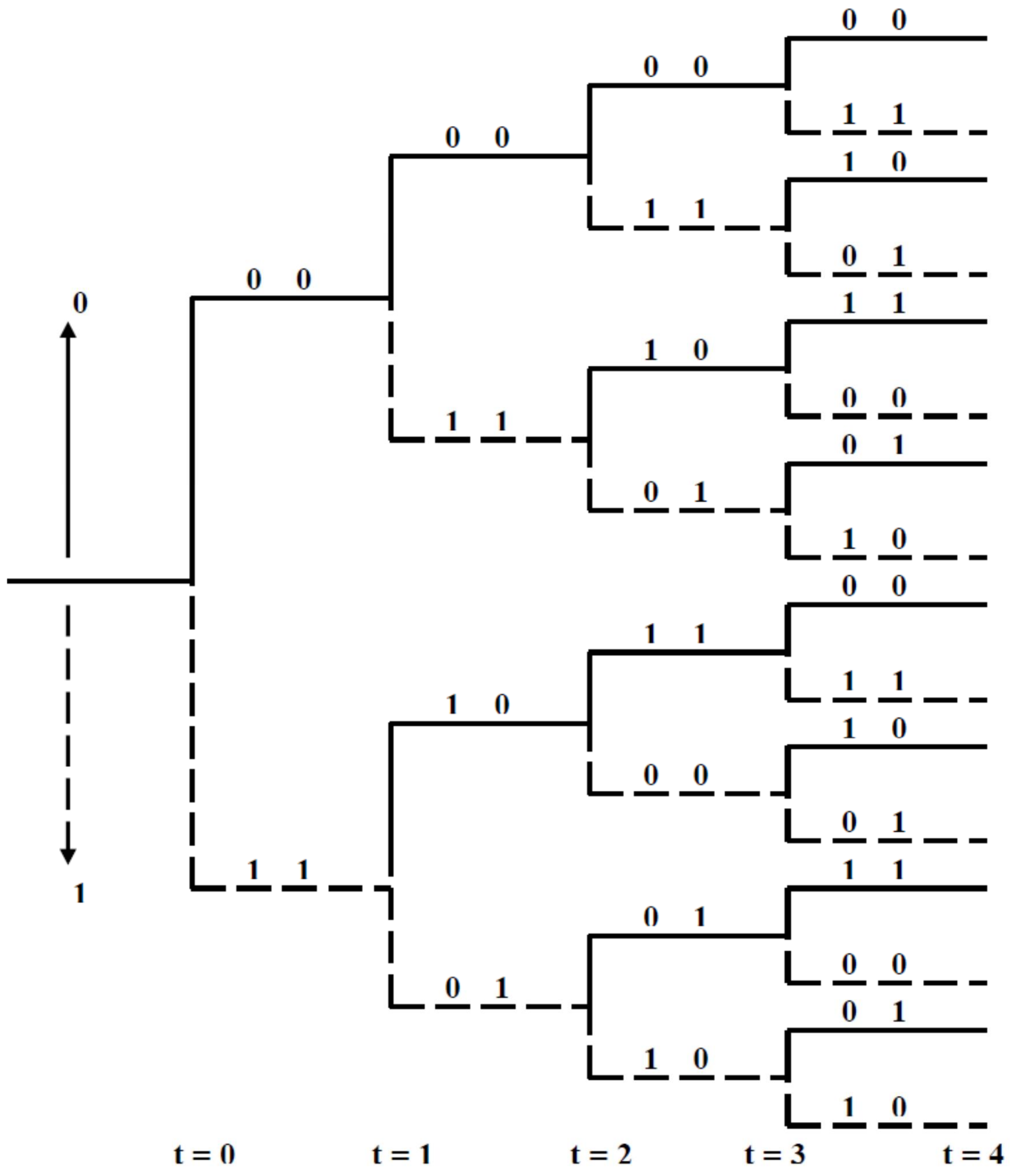


Fig. 3.2.3 A tree diagram for the encoder in Fig. 3.1.1

3.2.4 Trellis Diagram Representation :

The trellis diagram of a convolutional code is obtained from its state diagram. All state transitions at each time step are explicitly shown in the diagram to retain the time dimension, as is present in the corresponding tree diagram. Usually, supporting descriptions on state transitions, corresponding input and output bits etc. are labeled in the trellis diagram. It is interesting to note that the trellis diagram, which describes the operation of the encoder, is very convenient for describing the behavior of the corresponding decoder, especially when the famous ôViterbi Algorithm (VA)ô is followed. **Figure 3.2.4** shows the trellis diagram for the encoder in **Figure 3.1.1**.

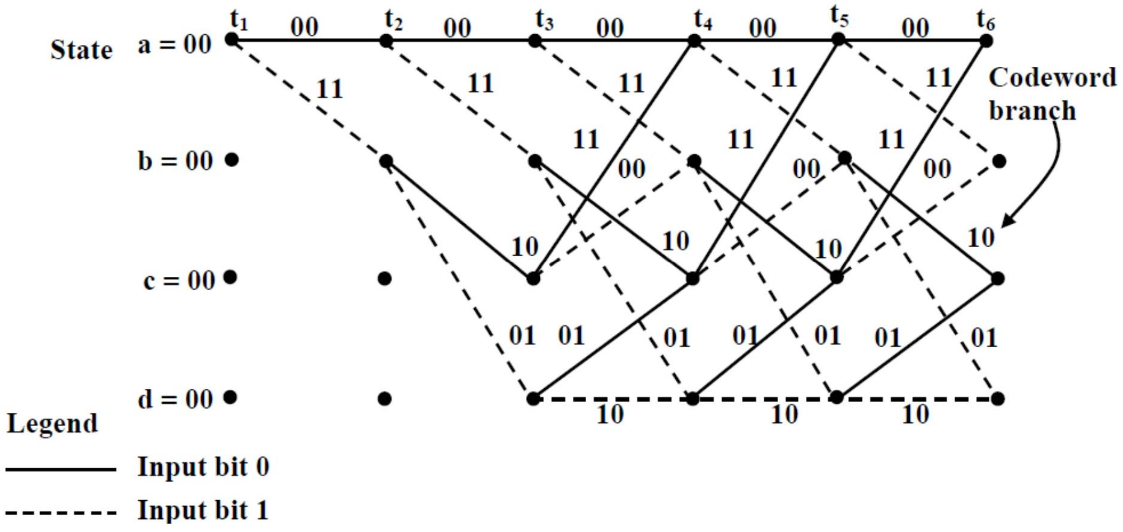


Fig. 3.2.5(a) Trellis diagram for the encoder in Fig. 3.1.1

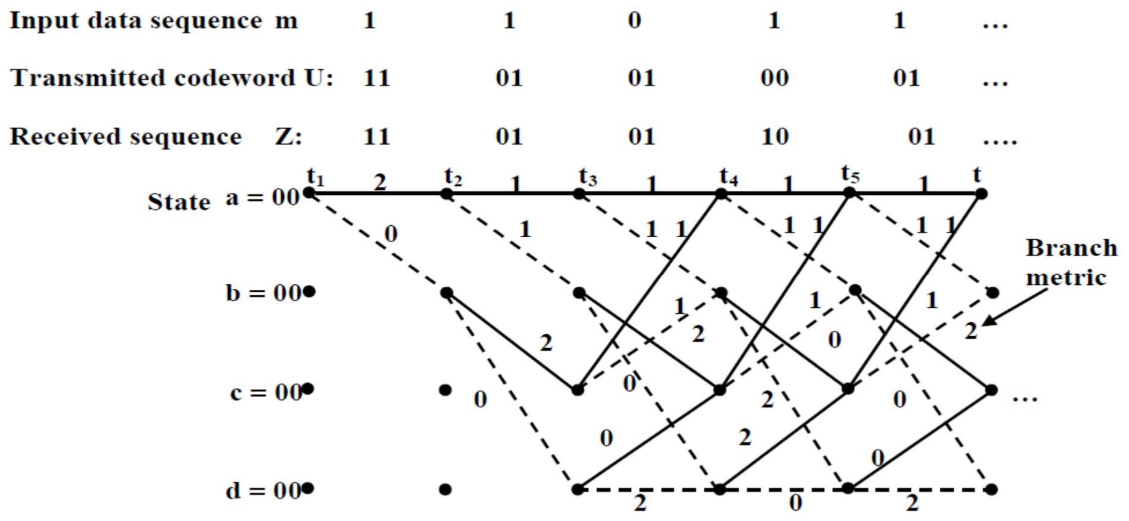

Input data sequence m 1 1 0 1 1 ...
Transmitted codeword U: 11 01 01 00 01 ...
Received sequence Z: 11 01 01 10 01 ....
State a = 00
b = 00
c = 00
d = 00
t1
t2
t3
t4
t5
t
Branch metric

Fig.3.2.4(b) Trellis diagram, used in the decoder corresponding to the encoder in Fig. 6.35.1

3.3 Maximum likelihood decoder:

If all input message sequences are equally likely, a decoder that achieves the minimum probability of error is one that compares the conditional probabilities, also called the likelihood functions P(Z|U<=l), where Z is the received sequence and u <m) is one of the possible transmitted sequences, and chooses the maximum. The decoder chooses u <m') if

P(Z|U(m') = max P(Z|U<'">) over all u <m)

The maximum likelihood concept is a fundamental development of decision theory ; it is the formalization of a "common-sense" way to make decisions when there is statistical knowledge of the possibilities. In the binary demodulation treatment there were only two equally likely possible signals, s 1(t) or s2(t), that might have been transmitted. Therefore, to make the binary maximum likelihood decision, given a received signal, meant only to decide that s1(t) was transmitted if p(z ~ ,) > p(zls2)otherwise, to decide that s2(t) was transmitted. The parameter z represents z(T), the receiver predetection value at the end of each symbol duration timet = T. However, when applying maximum likelihood to the convolutional decoding problem, we observe that the convolutional code has memory (the received sequence represents the superposition of current bits and prior bits). Thus, applying maximum likelihood to the decoding of convolutionally encoded bits is performed in the context of choosing the most likely sequence. There are typically a multitude of possible codeword sequences that might have been transmitted. To be specific, for a binary code, a sequence of L branch words is a member of a set of 2L possible sequences. Therefore, in the maximum likelihood context, we can say that the decoder chooses a particular u <m') as the transmitted sequence if the likelihood P(Z iu <m'l) is greater than the likelihoods of all the other possible transmitted sequences. Such an optimal decoder, which minimizes the error probability (for the case where all transmitted sequences are equally likely), is known as a maximum likelihood decoder. The likelihood functions are given or computed from the specifications of the channe l. We will assume that the noise is additive white Gaussian with zero mean and the channel is memoryless, which means that the noise affects each code symbol independently of all the other symbols. For a convolutional code of rate lin, we can therefore express the likelihood as

P(Z|u <m>) = II P(Z;u <r >) = II II P(zi;u<j)

For the decoding of convolutional codes, either the tree or the trellis structure can be used. In the tree representation of the code, the fact that the paths remerge is ignored. Since for a binary code, the number of possible sequences made up of L branch words is 2L, maximum likelihood decoding of such a received sequence, using a tree diagram,requires the "brute force" or exhaustive comparison of 2L accumulated loglikelihood metrics, representing all the possible different codeword sequences that could have been transmitted. Hence it is nol practical to consider maximum Likelihood decoding with a tree structure. It is shown in a later section that with the use of the trellis representation of the code, it is possible to configure a decoder which can discard the paths that could not possibly be candidates for the maximum likelihood sequence. The decoded path is chosen from some reduced set of surviving paths.

3.4 Channel model:

Soft and hard decision decoding:

Consider that a binary signal transmitted over a symbol interval (0, T) is representedby s 1(t) for a binary one and s2(t) for a binary zero. The received signal is r (r) =s1(1) + n(t), where n(l) is a zero-mean Gaussian noise process. the detection of r(t) in terms of two basic steps. In the first step the received waveform is reduced to a single number, z(T) =a;+ n0Åwhere a, is the signal component of z(T) and n0 is the noise component. The noise component. n0, is a zero-mean

Gaussian random variable, and thus z(T) is a GmLuian random variable with a mean of either a1 or a2 depending on whether a binary one or binary zero was sent. In the second step of the detection process a decision was made as to wl:tich signal was transmitted, on the basis of comparing 2(T) to a tlueshotd. The conditional probabilities of z(T) , p(z~ 1), and p(z~·z) are s labeled likelihood of s1 and likelihood of s2ÅThe demodulator in converts the set of time-ordered random variables {z(T)J into a code sequence Z, and passes it on to the decoder. The demodulator output can be configured in a variety of ways. lt can be implemented to make a firm or hard decision as to whether z(T) represents a zero or a one. In this case, the output of the demoduJator is quantized to two levels, zero and one. and fed into thedecoder (this is exaclly the same threshold decision . Since the decoder operates on the hard decisions made by the demodulator. the decoding is called hard-decision decoding.

The demodulator can also he configured to feed the decode r with a quantized value o f z(T) greater than two levels. Such an implementation furnishes the decoder with more information than is provided in the hard-decision case. When the quantization level of the demodulator

output is greater than two. the decoding i:> called soft-decision decoding. When the demodul ator sends a hard binary decision to the decoder. it sends it a single binary symhol. When the demodulator sends a soft binary decision, quantized to eight levels, it sends the decoder a 3-bit word describing ao interval along z(T). In effect, sending such a 3-bit word in place of a single binary symbol is equivalent to sending the decode r a measure of confidence along with the code-symbol decision. It should be clear that ultimately every message decision out of the decoder must be a hard decision; otherwise one might see computer printouts that read: "think it's a 1." "think it's a o:· and so on . The idea behind the demodulator not making hard decisions and sending more data (soft decisions) to the decoder can be thought of as an interim step to provide
the decoder with more inform ation, wbkh the decoder then uses for recovering the message sequence (with better e rror performance than it could in theca e of harddecision decoding).the 8-Jevel soft-decision me tric is often shown as - 7, -5, -3, -1 , 1, 3. 5, 7. Such a designation lends itselfto a simple interpretation of the soft decision.

3.5 The viterbi convolutional decoding algorithm:

The Viterbi decoding algorithm was discovered and analyzed by Viterbi in 1967.The Viterbi algorithm essentially performs maximum likelihood decoding. however, it reduces the computational load by taking advantage of the special structure in the code trees. The advantage of Viterbi decodin , compared with brute-force decoding, is that the complexity of a Viterbi decoder is not a function of the number of symbols in the codeword sequence. The algorithm involves calculating a measure of similarity, or distance, between the received signal at time ti and all the trellis paths entering each state at time t. The Viterbi algorithm removes from consideration those trellis paths that could not possibly be candidates for the maximum 1ikelihood choice. When two paths enter the same state, the one having the best metric is chosen; this path is called the surviving path. This selection of surviving paths is performed for all tbe states. The decoder continues in this way to advance deeper into the trellis, making decisions hy elimin ating the least likely paths. The early rejection of the unlikely paths reduces the decoding complexity. ln 1969, Omura demonstrated that the Viterbi algorithm is, in fact, maximum likelihood. Note that the goal of(selecting the optimum path can be expressed equivalently,as choosing the codeword with the maximum likelihood metric or as choosing the codeword with the minimum distance metric.

3.6 An Example of Viterbi Convolutional Decoding

For simplicity, a BSC is assumed; thus Hamming distance is a proper dislance measure. We start at time t1 in the 00 state (flushing the encoder between messages provides the decoder with starting-state knowledge). Since in this example, ther'e are nly two possible transitions leaving any state not all branches need be shown initially. The full trellis structure evolves after time t1.

The basic idea behind the decoding procedure can best be understood by examining encoder trellis in concert with decoder trellis. For the decoder trellis it is convenient at each time interval, to label each branch with the Hamming distance between the received code symbols and the branch word corresponding to the same branch from the encoder trellis. The example shows a message sequence m, the corresponding codeword sequence U, and a noise corrupted received sequence Z = 11 01 01 10 01 . .. . The branch words seen on the encoder trellis branches characterize the encoder and are known a priori to both the encoder and the decoder. these encoder branch words are the code symbols that would be expected to come from the encoder output as a result of each of the state transitions. The labels on the decoder trellis branches are accumulated by the decoder · That is, as the code symbols are received, each branch of the decoder trellis is labeled with a metric of similarity (Hamming distance) between the received code symbols and each of the branch words for that time interval. From the received sequence Z we see that the code symbols received at (following) time t1 are 11. In order to label the decoder

brancJ1es at (departing) time t1 with the appropriate Hamming distance metric. Here we see that a 00 transition yields an output branch word of 00. But we received 11.Therefore, on the decoder trellis we label the state 00 --) 00 transition with Hamming distance between them namely 2. Looking at tbe encouer trellis again, we see that a state 00 ~ 10 transition yields an output branch word of 11, whlch corresponds exactly witb the code symbols we received at time t 1ÅTherefore. on the decoder tre llis, we label the stare 00 ~ 10 transition with a Hamming distance of 0. In summary. the metric entered on a decoder trellis br·anch represents the difference (distance) between what was received and what "should have been" received had the branch word associated with that branch been transmilled. In effect, these metrics describe a correlation like measure between a received branch word and each of the candidate branch words. We continue labeling the decoder trellis branches in this way as the symbols are received at each time t1ÅThe decoding algorithm uses these Hamming distance metrics to find the most likely (minimum distance) path through the trellis. The basis of Viterbi decoding is the following observation: If any two paths in trellis merge to a single slate, one of them can always be eliminated in the search for an optimum path. For example, two paths merging at time 15 to state 00. Let us define the cumulative Hamming path metric of a given path at ti as the sum of the branch Hamming distance metrics along that pathup to time t.

3.7 Decoder Implementation:

In the context of the trellis diagram, transitions during any one time interval can be grouped into 2^n- 1 disjoint cells, each cell depicting four possible transitions. where v = K- 1 is called the encoder memory. For the K = 3 example, v = 2 and 2" _, = 2 cells. These cells where a. b, c, and d refer to the states at time t,., and a', b ', c ', and d' refer to the states at time f; + t· Shown on each transition is the branch metric 8..-vÅwllere the subscript indicates that the metric corresponds to the transition from st.ate x to state y. These cells and the associated logic units that update the state metrics (f_..}, where x designates a particular state, represent the basic building blocks ofthe decoder. Consider the same example that was used foT describing Vi terbi decoding. The message seq uence was m = 1 1 0 1 1. the codeword sequence was U = ·11 01 01 00 01, and the receive sequeuce was Z = ll 01 01 10 01 depicts a decoding trellis diagram similar. A branch metric that labels each branch is the Hamming distance between the received code symbols and the corresponding branch word from the encoder trellis. Additionally. the trellis indicates a value at each state x. and for each time from time t2 to r .which is a state metric r .Å We perform the add-compare-select (ACS) operation when there are two transitions entering a state, as there are for times t and later. For example at time t4 , the value of the s tate metric for state n is obtained by incrementing the state metric r = 3 at time 13 with the branch metric yielding a candidate value of 4. Simultaneously, the state metric r,. = 2 at time t3 is incremented

with the branch metric yielding a value of 3. The select operation of the ACS process selects the largest-likelihood (minimum distance) path metric as the new state metric; hence, for state at time 14Åthe new state metric is f 11Å= 3. The winning path is shown with a heavy line and the path that has been dropped is shown with a lighter line. On the trellis , observe the state metrics from left to right. Verify that at each time, the value of each state metric obtained by incrementing the connected state metric from the previous time along the winning path (heavy line) with the branch metric between them.

3.8 Path memory synchronisation:

The storage requirements of the Viterbi decoder grow exponentially with constraint length K. For a code with rate l /11, the decoder retains a set of 2A - 1 paths after each decod ing step. With high prohahility, these paths wi ll not be mutually disjoint very far back from the present decoding depth [1 2). All of the 21<. - 1 paths tend to have a common stem which eventually branches ro the various states. Thus if the decoder sto res enough of the history of the 2". 1 paths, the oldest bits at1 all paths will be the same. A simple decoder implementation. then. contain a fixed amount ofpath history and o utputs the oldest hit on an arbitrary path each time it steps one level deeper into the trellis. The amount of path storage required is [1 2] II = lz2" .
where h is the tength of the inform ation bit path history per state. A refinement which minimize the value of h. uses the oldest bit on the most likely path as the decoder output instead of the oldes t bit on an arbitrary path. It has been demonstrated that a value of h of 4 or 5 times the code constraint length is efficient for n optimum decoder performance. The storage requirement u is the basic limitation on the implementation of Viterbi decoders. Commercial decoders are limited to a constraint length of about K = 10. Efforts to inrease coding gain by further increasing constraint length arc met by the exponential in memory requirements (and complexity).Branch word synchronisation is the process of determining the beginning of a branch word in the received sequence. Such synchronization can take place without new information being added to the transmitted symbol stream because the received data appear to have an excessive error rate when not synchronized. Therefore, a simple way of accomplishing synchronisation is to monitor some concomitant indication of

this large error rate, tbat is, the r at<:: at which the state metrics are increasing or the rate at which the surviving paths in the trellis merge. The monitored parameters are compared to a thre hold, and synchronization is then adjusted accordingly.

3.9 Properties of convolutional codes:

1. distance properties of convolutional code:

We want to evaluate the distance between all possible pairs of codeword sequences. As in the case of block codes. we are interested in the minimum distance between all pairs of such codeword seq uences in the code , since the minimum distance is related to the error-correcting capability of the code. Because a convolutional code is a group or linear code there is no loss in generality in simply finding the minimum distance between each of the codeword sequences and the all-zeros sequence. In other words, for a linear code. any test message is just as "good" as any other Sl message. So, why not choose one that is easy to keep track of- namely the sequence? Assumjng that the all-zeros input sequence was transmitted, the paths of inte-rest are those that start and end in the 00 state and do not return to the 00 s late anywhere in between. An error will occur whenever the distance of any other path that merges with the a= 00 state at time t; is less than that of the allzeros path up to time causing the all-zeros path to be discarded in the decoding

process. In other words. given the all-zeros transmission. an error occurs whenever the all-zero, path does not survive. Thus an error of interest is associated with a surviving path that diverges from and then remerges to the all-zeros path. One might ask, Why is it necessary for the path to remerge? Isn't the divergence enough to indicate an error? Yes, of course, but an error characterized by only a divergence means that the decoder. from thar point on. will be outputting "garbage" for the rest of the message duration. We want to quantify the decode r's capability in terms of errors that will usually take place that is, the easiest" way for the decoder to make an error. The minimum distance for making such an error can be found by exhaustively examining every path from the 00 state to the 00 state. First. let us redraw the trellis diagram, labeling each

branch with its Hamming distance from the all-zeros codeword instead of with its branch word symbols. The Hamming distance between two unequal-length sequences will be found by first appending the necessary number of zeros to the shorter sequence to make the two sequences equal in length. Consider all the paths that diverge from the all-zeros path and then remerge for the first time at some arbitrary node. From Figure 7.16 we can compute the distances of these paths from the all-zeros path. There is one path at distance 5 from the all-zeros path; this path

departs from the all-zeros path at time and merges with it at time . Similarly, there are two paths at distance which departs at time t1 and merges at time t5Åand the other which departs at time r1 and merges af time t1,. and :-;o on. We can also sec from the dashed and solid lines of the diagram that the input bits for the distance 5 path are 1 0 0: it diller in only one input bit from the all-zeros input sequence. Similarly, the input bits for the d istance 6 paths are I I 0 0 and 1 0 1 0 0:each differs in two positions from the all-zeros path. The minimum distance in the set of all arbitrarily long paths that diverge and remerge, called the minimum free distance or simply the free distance.

2.Systematic and Non-systematic Convolutional code:

For linear block codes, any nonsystematic code can be transformed into a systematic code with the same block distance properties .A systematic convolutional code is one in which the input k-tuple appears as part of the output branch word n-tuple associated with that k-tuple. Figure shows a binary, rate t, K = 3 systematic encoder..

This is not the case for convolutional codes. The reason for this is convolutional codes depend largely on free dis1ance; making the convolutional code systematic, in general reduces the maximum possible free distance for a given constraint length and rate.

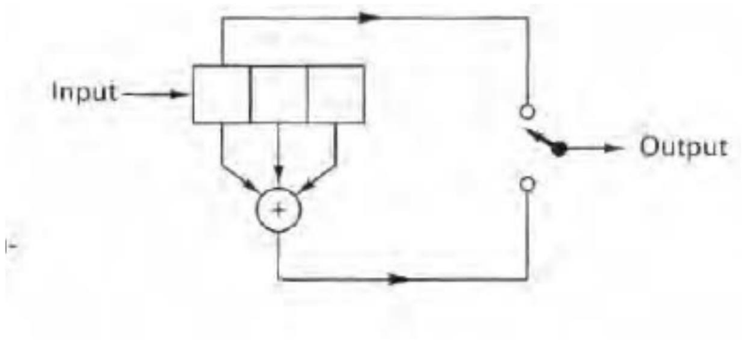
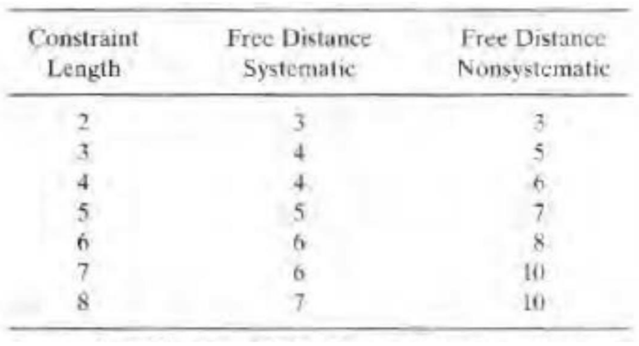


Fig 3.9.1: Systematic Convolutional Encoder, rate=1/2,k=3

Table:Comparision of systematic and non-systematic free distance



Constraint Length	Free Distance Systematic	Free Distance Nonsystematic
2	3	3
3	4	5
4	4	6
5	5	7
6	6	8
7	6	10
8	7	10

3.10 Catastrophic Error Propagation in Convolutional Codes:

A catastrophic error is defined as an event whereby a finite number of code symbol errors cause an infinite number of decoded data bit errors. Massey and Sain have derived a necessary and sufficient condition for convolutional codes to display catastrophic error propagation. For rate in codes with register taps designated by polynomial generators. as described the condition for catastrophic error propagation is that the generators have a common polynomial factor (of degree at least one). For example, Figure illustrates a rate ~ - K = 3 encoder

with upper polynomial g1(X) and lower polynomial glX), as follows:

$g1(X) = 1 + X$

$g2(X) = 1 + X^2$

The generators g1(X) and g2(X) have in common the polynomial factor 1 +X since $1 + X^2 = (I + X)(l - X)$.Therefore, the encoder in F igure 3.10.1a can manifest catastrophic error propagation.

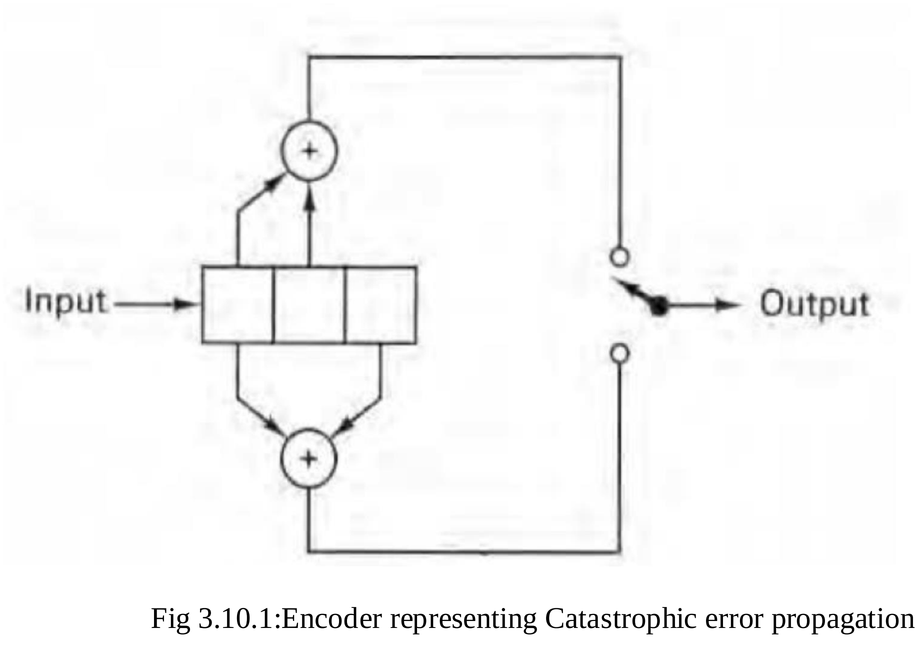


Fig 3.10.1:Encoder representing Catastrophic error propagation

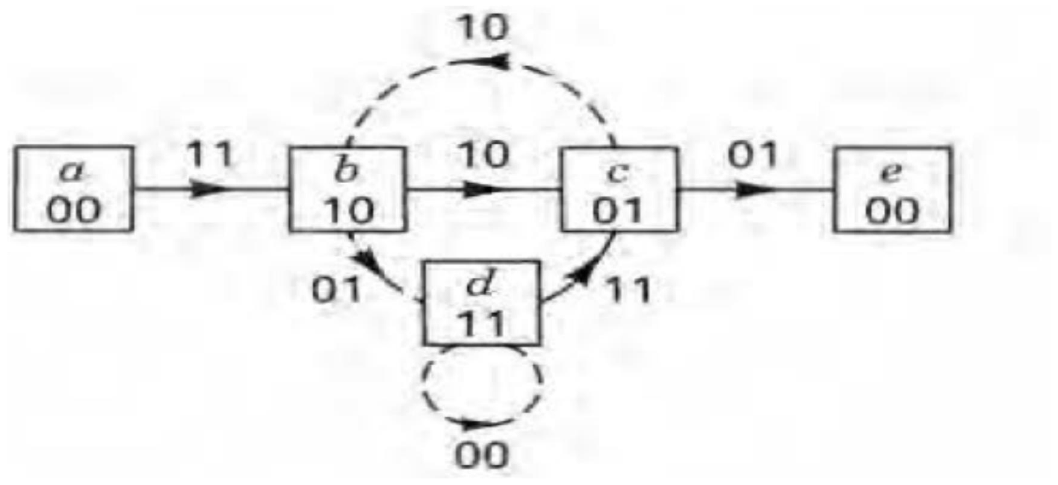


Fig 3.10.2: State diagram

3.11Coding Gain:

Coding gain is defined as the reduction usually expressed in decibels in the required E/N0 to achieve a specified error probability of the coded system over an uncoded system with the same modulation and channelcharacteristics. Table lists an upper bound on the coding gains, compared to uncoded coherent BPSK, for several maximum free distance convolutional codes with constraint lengths varying from 3 to 9 over a Gaussian challnel with hard decision decoding. The table illustrates that it is possible to achieve significant coding gain even with a simple convolutional code. The actual coding gain which vary with the required bit error probability . Table lists the measured coding gains, compared to uncoded coherent BPSK, achieved with hardware implementation or computer simulation over a Gaussian channel with soft-decision decoding . The uncoded E/N0 is given in the leftmost column. From Table we can see that coding gain increases as the bit error probability is decreased. However, the coding gain cannot increase indefinitely; it has an upper bound as shown in the table. This bound in decibels can he

shown where r is the code rate and d1 is the free distance. Examination of Table also reveals that for code rates the weaker codes tend to be closer to the upper bound than are the more powerful codes. Typically Viterbi decoding is u ed over binary input channels with either hard or 3-bit soft quantized outputs. The constraint lellgths vary between 3 and 9, the code rate is rarely smaller than t and to be path memory is usually a few constraint length. The path memory refers to the depth of the input bit history stored hy the decoder. From the Viterbi decoding example one might question the notion of a fixed path memory. It seems from the example that the decoding of a branch word. at any arbitrary node, can take place as soon as the Cr<.l is only a single surviving branch at that node. That is true; however, to actually implement the decoder in this way would entail an extensive amount of processing to continually check when the branch word can be decoded. Instead. a fixed delay is provided, after which the branch word is decoded. It has been shown that tracing back from the state with the lowest state metric, over a fixed amount of path history (about 4 or 5 times the constraint length), is sufficient to limit the degradation from the optimum decoder performance to about 0. L dB for the BSC and Gaussian channels. Typical error performance simulation results are shown for Viterbi decoding with bard decision quantization. Notice that each increment in constraint length improves the required E_b/N_0 by a factor of approximately 0.5 dB .

3.12 Best Known Convolutional Codes:

The connection vectors or polynomial generators of a convolutional code are usually selected based on the code's free distance properties. The first criterion is to select a code that does not have catastrophic error propagation and that has the maximum free distance for th e given rate and constraint length. Then the number of paths at the free distance d1, or tbe number of data bit errors the paths represent, should be minimized. The selection procedure c.-an be furth er refined by considering the number of paths or bit errors at d1 + 1, at d1 + 2, and so on, until only one

code or class of codes remains. A list of the best known codes of rate t K = 3 to 9, and rate!, K = 3 to 8, based on this criterion was compiled by Odenv,;alder and is given in Table. The connection vectors io this table represent the presence or absence (1 or 0) of a tap connection on the corresponding stage of the convolutional encoder, the leftmost term corresponding to the leftmost stage of the encoder register. lt is interesting to note that these connections can be inverted (leftmost and rightmost can be interchanged in the above description). Under the condition of Viterbi decoding that inverted connections give rise to codes with identical distance properties, and hence identical performance. as those inTable

Rate	Constraint Length	Free Distance	Code Vector
$\frac{1}{2}$	3	5	111 101
$\frac{1}{2}$	4	6	1111 1011
$\frac{1}{2}$	5	7	10111 11001
$\frac{1}{2}$	6	8	101111 110101
$\frac{1}{2}$	7	10	1001111 1101101
$\frac{1}{2}$	8	10	10011111 11100101
$\frac{1}{2}$	9	12	110101111 100011101
$\frac{1}{3}$	3	8	111 111 101
$\frac{1}{3}$	4	10	1111 1011 1101
$\frac{1}{3}$	5	12	11111 11011 10101
$\frac{1}{3}$	6	13	10111 110101 111001
$\frac{1}{3}$	7	15	1001111 1010111 1101101
$\frac{1}{3}$	8	16	11101111 10011011 10101001

3.13 Convolutional Code Rate Trade-off:

Performance with Coherent PSK Signaling:

The error-correcting capability of a coding scheme increases as the number of channel symbols n per information bit k increases or the rate k in decreases. However, the channel bandwidth and the decoder complexity both increase with n. The advantage of lower code rates when using convolutional codes with coherent PSK is that the required Eb/N0 is decreased (for a large range of code rates), permitting the transmission of higher data rates for a given amount of power, or permitting reduced power for <t given data rate. Simulation studies have shown for a fixed constraint length, a decrease in the code rate (results in a reduction of the required Eb/N0 of roughly 0.4 d B. However, the corresponding increase in

decoder complexity is about 17%. For smaller values of code rate the improvement in performance relative to the increased decoding complexity diminishes rapidly. Eventually, a point is reached where further decrease in code rate is characterized by a reduction in coding gain.

Performance with Noncoherent Orthogonal Signalling

ln contrast to PSK, there is an optimum code rate of about i for noncoherent orthogonal signaling. Error performance at rates of 1, and i are each worse than those for rate j· For a fixedconstraint length L the rate i and j codes typically degrade by about 0.25, 0.5. and 0.3 dB, respectively relative to the rate performance.

MODULE-IV

4.1 Reed-Solomon codes:

Reed-Solomon(RS) codes are a class of non-binary codes which are particularly useful in burst error correction. Errors in a communication system can be divided into 2 types-Independent errors or Random errors and Burst errors. As far as independent error is considered, this is usually caused by Gaussian noise. This type of error is random or independent in nature i.e. error introduced during a particular time interval does not affect the performance of system during the subsequent time interval. Whereas burst error is encountered due to impulse noise. It usually affects more than one symbol. Burst errors affect the performance of system during the subsequent time intervals i.e. these errors are dependent on each other. Channels having this type of error is said to be having memory.

RS codes are important subclass of BCH codes.Theses codes operate on multiple bits.

RS codes (n, k) on m-bit symbols exist for,

$0 < k < n < 2m + 2$

Where, m -- +ve integer >2

k ï no. of data symbols being encoded

n ï total no. of code symbols in the encoded block.

Most conventional RS(n, k) codes,

$$(n, k) = (2m - 1, 2m - 1 - 2t)$$

t ï symbol error correcting capability of the code.

Therefore, no. of parity bits = n-k = 2t.

Extended RS codes ---

$n = 2^m$ or $n = 2^{m+1}$

Achieve largest possible minimum distance(d_{min}) for any linear code,

$$d_{min} = n-k+1$$

and therefore error correcting capability(t) becomes,

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor = \left\lfloor \frac{n - k}{2} \right\rfloor$$

where LxJ means the largest integer not to exceed.Erasure correcting capability(p) of the code,

p = dmin - 1 = n ï k.Simultaneous error correction and erasure correction capability,

2a + "Y < dmin < n ï k

where a is the number of symbol error patterns that can be corrected, and "Y is the number of symbol erasure patterns that can be corrected.

Major advantages of RS-codes:--

Achieves large d_{min} ,

Eg. For binary(n, k) = (7, 3) code

$2^3 = 8$ no. of codewords have to be chosen from $2^7 = 128$ no. of words.

Therefore, ratio is 1:16

And for RS(7, 3) code, for each symbol comprising of m = 3 bits,

$(2^k)^m = (2^3)^3 = 512$ no. of codewords have to be chosen from a total of

$(2^n)^m = 2^{21} = 2097152$ words.

Therefore, ration is 1:4096.

Now, comparing both ratios we can easily conclude that for RS-codes, there is a larger availability of words per codewords resulting in larger d_{min}. Particularly useful for burst-error correction.

4.1.2 Reed-Solomon Error Probability:-

the R-S decoded symbol error probability, P_E, in terms of the channel symbol error probability, p, can be written as follows,

$$P_E \approx \frac{1}{2^m - 1} \sum_{j=t+1}^{2^m - 1} j \binom{2^m - 1}{j} p^j (1-p)^{2^m - 1 - j}$$

where t is the symbol-error correcting capability of the code, and the symbols are made up of m bits each.

Fig. below shows P_B versus the channel symbol error probability p, plotted for various t-error correcting 32-ary orthogonal Reed-Solomon codes with n=31(thirty-one 5-bit symbols per code block).

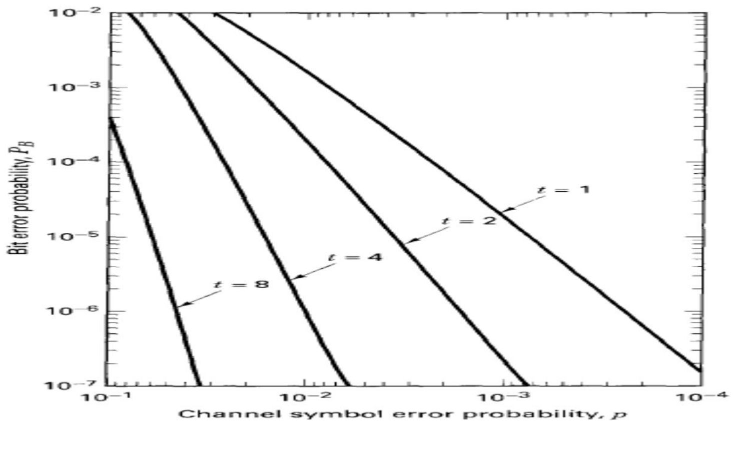


Fig. P_B versus p for 32-ary orthogonal signaling and n=31, t-error correcting Reed-Solomon coding.

Fig. below shows P_B versus E_b/N_0 for such a coded system using 32-ary MFSK modulation an noncoherent demodulation over an AWGN channel. For R-S codes, error probability is an exponentially decreasing function of block length.

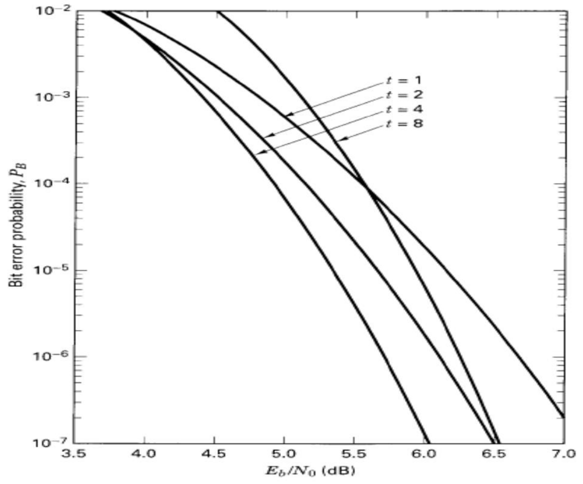


Fig. Bit error probability versus E_b/N_0 performance of several n=31, t-error correcting Reed-Solomon coding systems with 32-ary MFSK modulation over an AWGN channel.

4.1.3 Why R-S Codes perform well against burst noise;-

In R-S codes, each symbol is made up of several bits and error correcting capability is expressed in terms of symbols. Assuming that each symbol is made up of 8 bits and there is a burst noise that is lasting for 21 continuous bits. But in terms of symbols this can be said to be of affecting 3 symbols. So, for a R-S code of having an error correcting capability of just 3 symbols, it will be able to correct this 21 bit long burst error.

So, the property that each symbol is composed of several bits gives R-S code a tremendous burst-noise advantage against binary codes.

4.1.4 R-5 Performance as a Function of Size, Redundancy, and Code Rate:-

R-S codes are form an attractive choice whenever long block lengths are desired. This is clearly visible from the below figure where the rate of the code is held at a a constant 7/8, while its block size increases from n=32 symbols(with m=5 bits per symbol) to n=256 symbols(with m= 8 bits per symbol). So, the block size increases from 160 bit to 2048 bits.

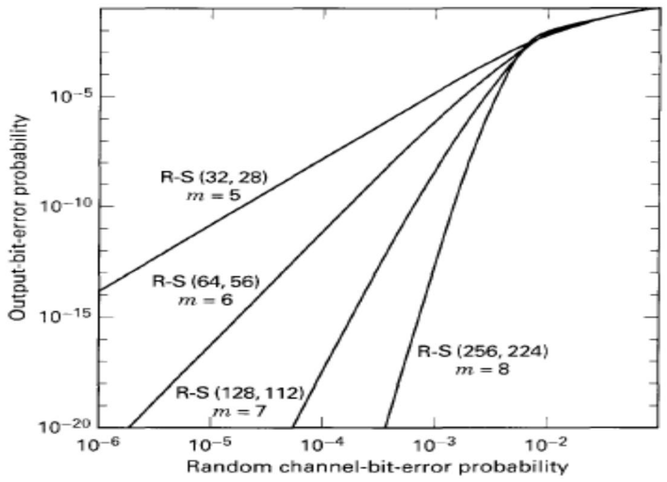


Figure 8.4 Reed–Solomon, rate 7/8, decoder performance as a function of symbol size.

Fig. R-S code, rate 7/8, decoder performance as a function of symbol size.

As the redundancy of an R-S code increases (lower code rate), its implementation grows in complexity (especially for high speed devices). Also, the bandwidth expansion must grow for any real-time communications application. However, the benefit of increased redundancy, just like the benefit of increased symbol size, is the improvement in bit-error performance, as can be seen in figure below, where the code length n is held at a constant 64, while number of data symbols decreases from k = 60 to k = 4 (redundancy increases from 4 symbols to 60 symbols).

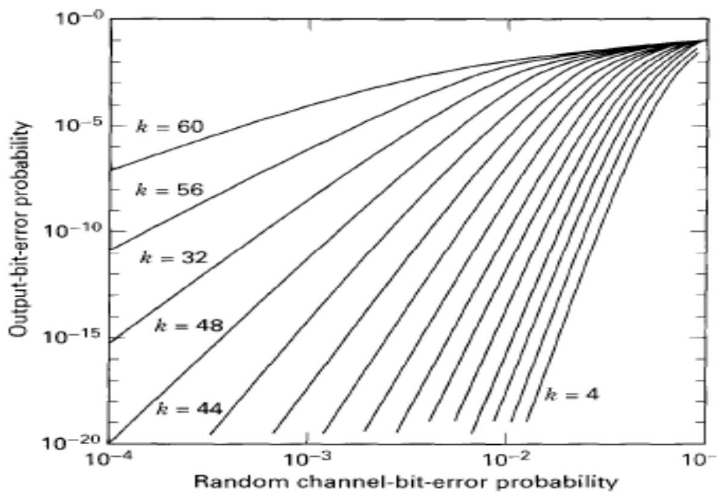


Fig. Reed-Solomon (64, k) decoder performance as a function of redundancy.

4.2 Interleaving and concatenated codes:-

Throughout this and earlier chapters we have assumed that the channel is memoryless, since we have considered codes that are designed to combat random independent errors. A channel that has memory is one that exhibits mutually dependent signal transmission impairments. A channel that exhibits multipath fading, where signals arrive at the receiver over two or more paths of different lengths, is an example of a channel with memory. The effect is that the signals can arrive out of phase with each other, and the cumulative received signal is distorted. Wireless mobile communication channels, as well as ionospheric and tropospheric propagation channels, suffer from such phenomena. Also, some channels suffer from switching noise and other burst noise (e.g., telephone channels or channels disturbed by pulse jamming). All of these time-correlated impairments result in statistical dependence among successive symbol transmissions. That is, the disturbances tend to cause errors that occur in bursts, instead of as isolated events.

Under the assumption that the channel has memory, the errors no longer can be characterized as single randomly distributed bit errors whose occurrence is independent from bit to bit. Most block or convolutional codes are designed to combat random independent errors. The result of a channel having memory on such coded signals is to cause degradation in error performance. Coding techniques for channels with memory have been proposed, but the greatest problem with such coding is the difficulty in obtaining accurate models of the often time-varying statistics of such channels. One technique, which only requires a knowledge of the duration or span of the channel memory, not its exact statistical characterization, is the use of time diversity or interleaving.

The interleaver shuffles the code symbols over a span of several block lengths (for block codes) or several constraint lengths (for convolutional codes). The span required is determined by the burst duration.

Fig. below illustrates a simple interleaving example.

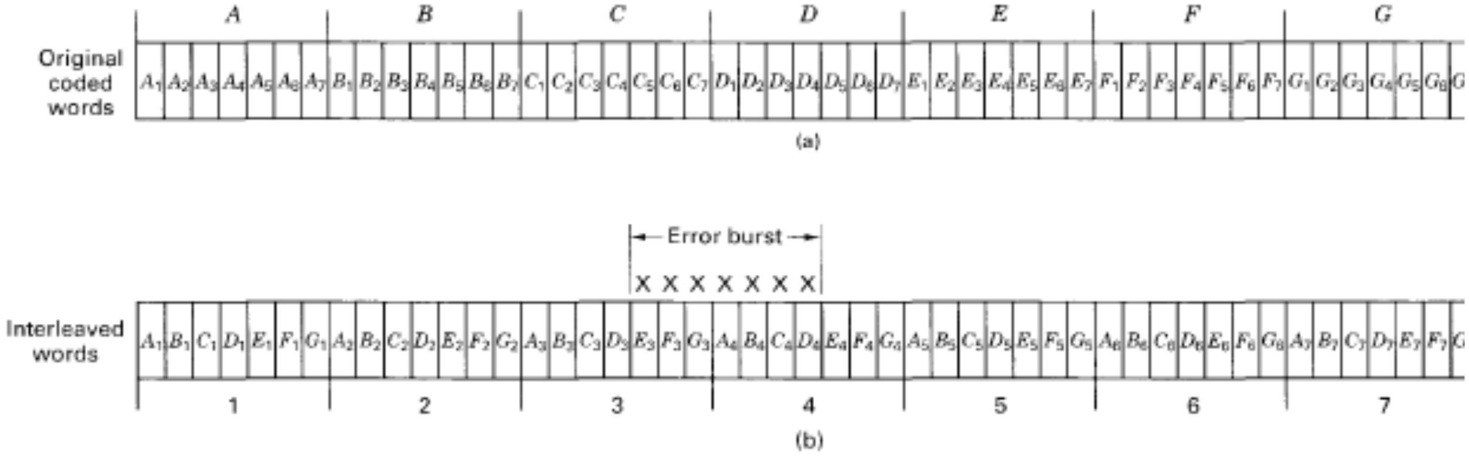
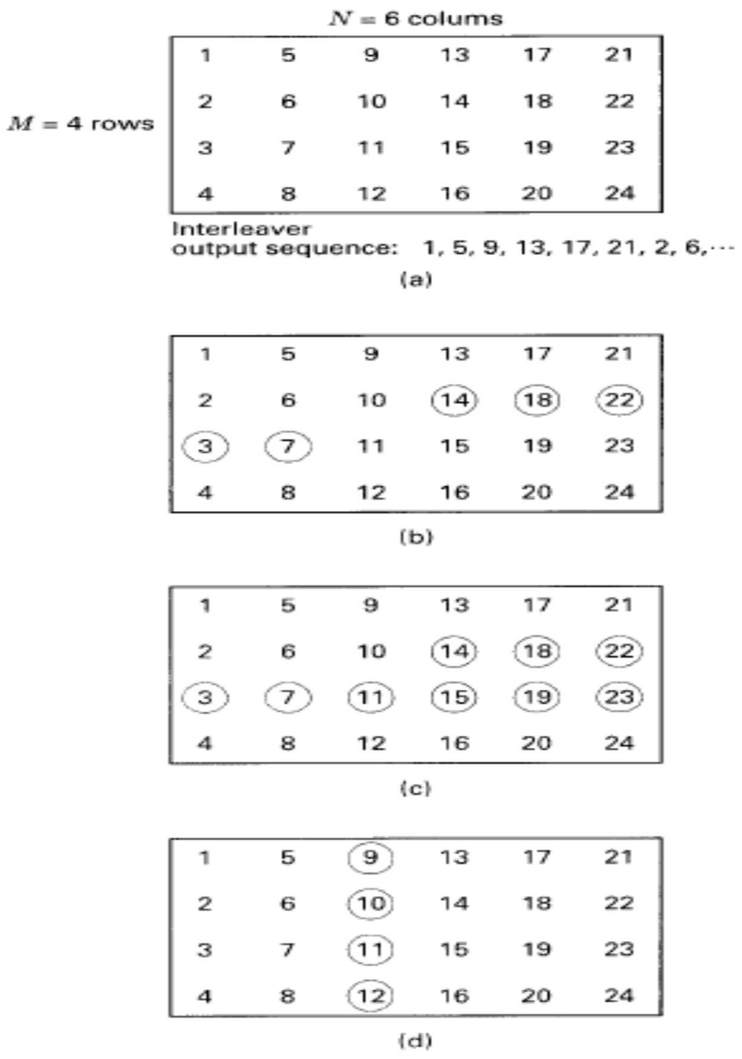


Fig. Interleaving example. (a) Original uninterleaved codewords, each comprised of seven code symbols.

(b) Interleaved code symbols.

4.2.1 Block Interleaving

A block interleaver accepts the coded symbols in blocks from the encoder, permutes the symbols, and then feeds the rearranged symbols to the modulator. The usual permutation of the block is accomplished by filling the columns of an M-row-by N-column (M x N) array with the encoded sequence. After the array is completely filled, the symbols are then fed to the modulator one row at a time and transmitted over the channel. Figure below illustrates an example of an interleaver with M = 4 rows and N = 6 columns



The most important characteristics of such a block interleaver are as follows:

Any burst of less than N contiguous channel symbol errors results in isolated errors at the deinterlever output that are separated from each other by at least M symbols.Any bN burst of errors, where b > 1, results in output bursts from the deinterleaver of no more than I b l symbol errors. Each output burst is separated from the other bursts by no less than M - LbJ symbols. The notation lx l means the smallest integer no less than x, and LxJ means the largest integer no greater than x

A periodic sequence of single errors spaced N symbols apart results in a single burst of errors of length Mat the deinterleaver output.

The interleaver/deinterleaver end-to-end delay is approximately 2MN symbol times. To be precise, only M(N- 1) + 1 memory cells need to be filled before transmission can begin (as soon as the first symbol of the last column of the M x N array is filled). A corresponding number needs to be filled at the receiver before decoding begins. Thus the minimum end-to-end delay is (2M N- 2M+ 2) symbol times, not including any channel propagation delay.

The memory requirement is MN symbols for each location (interleaver and deinterleaver). However, since theM x N array needs to be (mostly) filled before it can be read out, a memory of 2MN symbols is generally implemented at each location to allow the emptying of one M x N array while the other is being filled, and vice versa.

4.2.2 Convolutional Interleaving:-

Convolutional interleavers have been proposed by Ramsey and Forney. The structure proposed by Forney appears in fig. below.

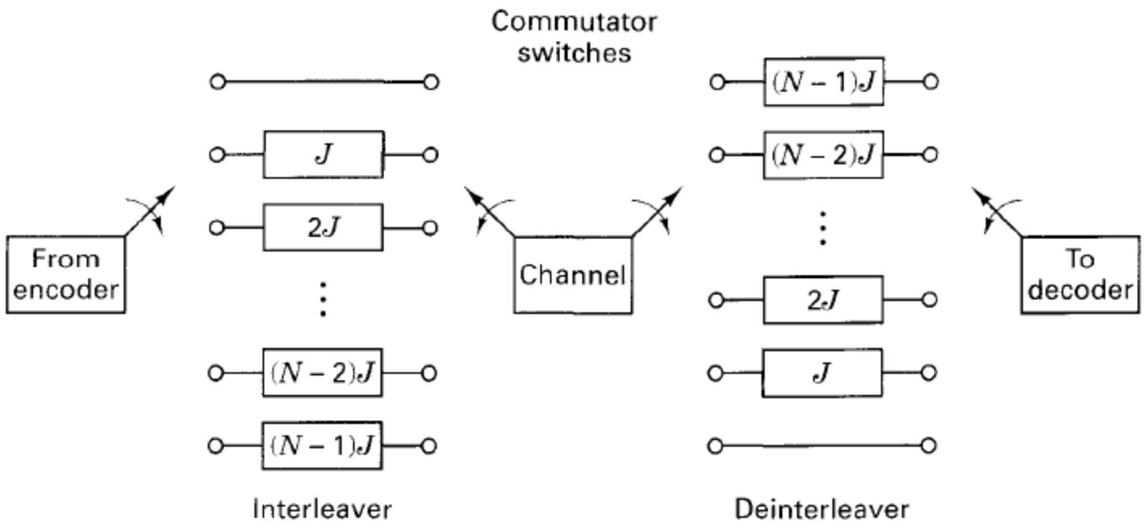


Fig. Shift register implementation of a convolutional interleaver/deinterleaver.

The code symbols are sequentially shifted into the bank of N registers; each successive register provides J symbols more storage than did the preceding one. The zeroth register provides no storage (the symbol is transmitted immediately). With each new code symbol the commutator switches to a new register, and the new code symbol is shifted in while the oldest code symbol in that register is shifted out to the modulator/transmitter. After the (N - 1)th register, the commutator returns to the zeroth register and starts again. The deinterleaver performs the inverse operation, and the input and output commutators for both interleaving and de interleaving must be synchronized.

4.2.3 Concatenated codes:-

A concatenated code is one that uses two levels of coding, an inner code and an outer code, to achieve the desired error performance.

Figure below illustrates the order of encoding and decoding.

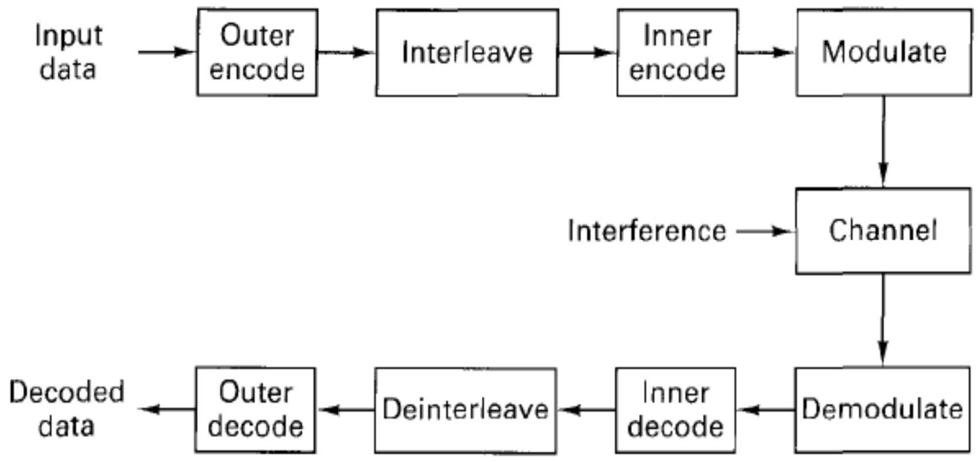


Fig. Block diagram of a concatenated coding system.

The inner code, the one that interfaces with the modulator/demodulator and channel, is usually configured to correct most of the channel errors.

The outer code, usually a higher-rate (lower-redundancy) code then reduces the probability of error to the specified level.

The primary reason for using a concatenated code is to achieve a low error rate with an overall implementation complexity which is less than that which would be required by a single coding operation.

The interleaver required to spread any error bursts that may appear at the output of the inner coding operation.

One of the most popular concatenated coding systems uses a Viterbi-decoded convolutional inner code and a Reed-Solomon (R-S) outer code. with interleaving between the two coding steps.

4.2.4 CODING AND INTERLEAVING APPLIED TO THE COMPACT DISC DIGITAL AUDIO SYSTEM:-

Philips & Sony Corp. defined a standard for digital storage & reproduction of audio signals called compact disc(CD) digital audio system.

World standad

120 mm diameter CD.

- Stores digitized audio waveform.
- Sampled at 44.1 ksamples per second for 20 Khz BW to 2^{16} levels(16 bits per sample).
- Dynamic range 96 dB, harmonic distortion = 0.005%.
- Stores about 10^{10} bits.

Scratches & other damage to CD causes burstlike errors.

- CIRC(corss interleave Reed Solomon code) is used in these systems to encode and combat burst errors.

Approximaterly 4000 bits (2.5 mm) burst errors can be corrected. Prob. of bit error, $P_B = 10^{-4}$.

Hierarchy of errors control in CIRC systemð

(i) Decode first attempts for error correction.

If error correction capability is exceeded, decoder goes for reassure correction.

If the ereasure correction capability is exceeded the decoder attempts to conceal unreliable data samples by interpolation between reliable neighbouring samples.

If the interpolation capability is exceeded, the decoder simply mutes the system for the duration of unreliable samples.

4.3 CIRC Encoding:-

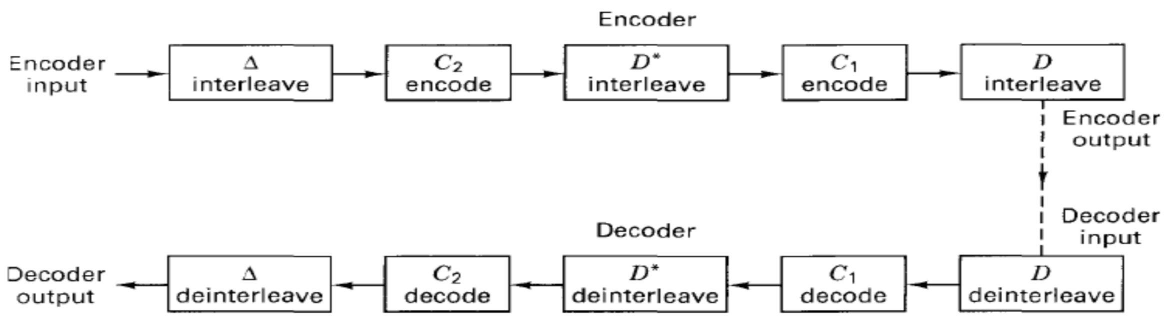


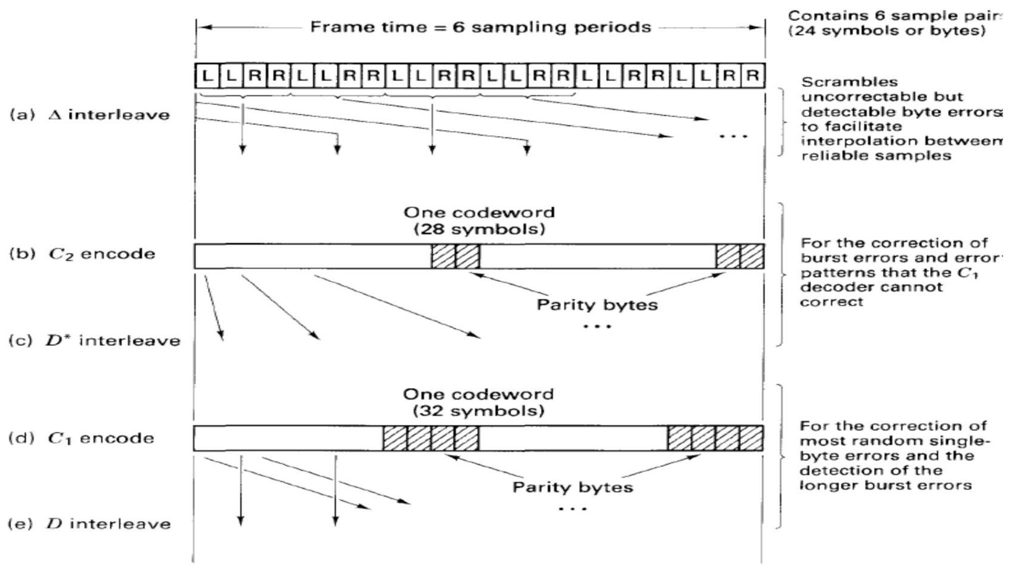
Fig. Block Diagram of CIRC Encoder & Decoder

The steps are as follows:-

L1 interleave. Even-numbered samples are separated from odd-numbered samples by two frame times in order to scramble uncorrectable but detectable byte errors. This facilitates the interpolation process.

C2 encode. Four Reed-Solomon (R-S) parity bytes are added to the 11-interleaved 24-byte frame, resulting in a total of n = 28 bytes. This (28, 24) code is called the outer code.

D* interleave. Here each byte is delayed a different length, thereby spreading errors over several codewords. C2 encoding together with D* interleaving have the function of providing for the correction of burst errors and error patterns that the C1 decoder cannot correct.



Frame time = 6 sampling periods
Contains 6 sample pair
(24 symbols or bytes)
L L R R L L R R L L R R L L R R L L R R L L R R
(a) Δ interleave
Scrambles uncorrectable but detectable byte errors to facilitate interpolation between reliable samples
One codeword (28 symbols)
(b) C_2 encode
Parity bytes
For the correction of burst errors and error patterns that the C_1 decoder cannot correct
(c) D^* interleave
One codeword (32 symbols)
(d) C_1 encode
Parity bytes
For the correction of most random single-byte errors and the detection of the longer burst errors
(e) D interleave

Fig. Compact disc encoder. (a)~ interleave. (b) C2 encode. (c) D* interleave. (d) C1 encode. (e) D interleave.

C1 encode. Four R-S parity bytes are added to the k = 28 bytes of the D*-interleaved frame, resulting in a total of n = 32 bytes. This (32, 28) code is called the inner code.

D interleave. The purpose is to cross-interleave the even bytes of a frame with the odd bytes of the next frame. By this procedure, two consecutive bytes on the disc will always end up in two different codewords. Upon decoding, this interleaving, together with the C1 decoding, results in the correction of most random single errors and the detection of longer burst errors.

4.3.1 CIRC Decoding:

The benefits of CIRC are best seen at the decoder, where the processing steps, shown in Figure 8.17 are in the reverse order of the encoder steps. The decoder steps are as follows:

D deinterleave. This function is performed by the alternating delay lines marked D. The 32 bytes (B_{i1}, ... , B_{i32}) of an encoded frame are applied in parallel to the 32 inputs of the D deinterleaver. Each delay is equal to the duration of 1 byte, so that the information of the even bytes of a frame is cross-deintcrleaved with that of the odd bytes of the next frame.

C1 decode. The D deinterleaver and the C1 decoder are designed to correct a single byte error in the block of 32 bytes and to detect larger burst errors. If multiple errors occur, the C1 ecoder passes them on unchanged, attaching to all 28 remaining bytes an erasure flag, sent via the dashed lines (the four parity bytes used in the C1 decoder are no longer retained).

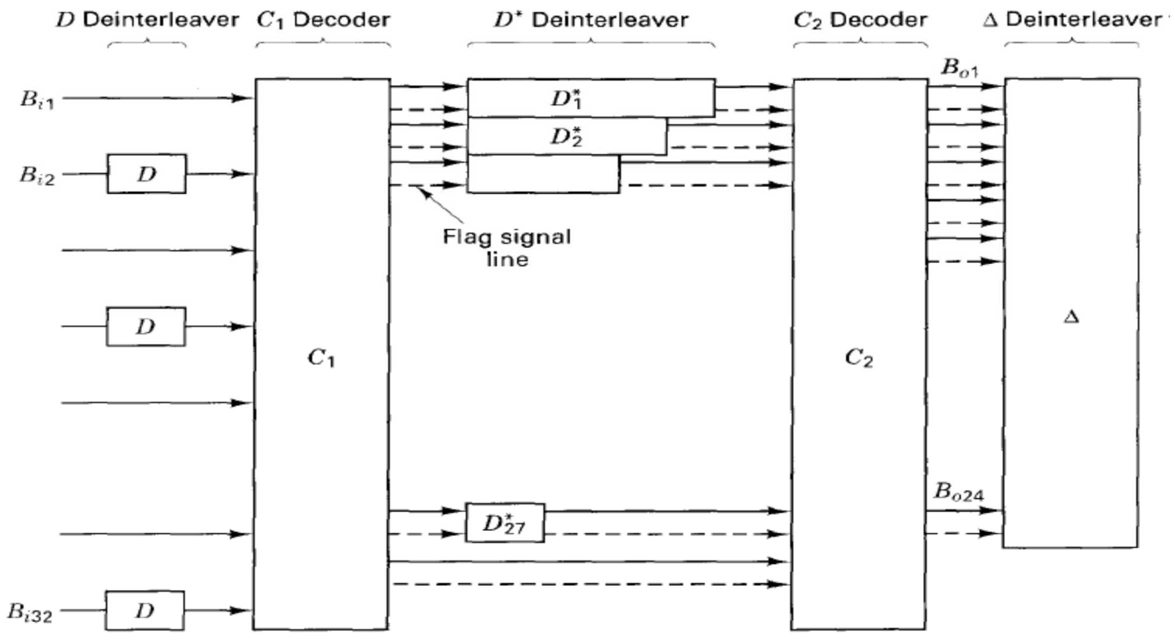


Fig. Compact disc decoder.

D* deinterleave. Due to the different lengths of the deinterleaving delay lines D*(1, ... , 27), errors that occur in one word at the output of the C1 decoder are spread over a number of words at the input of the C2 decoder. This results in reducing the number of errors per input word of the C2 decoder, enabling the C2 decoder to correct these errors

C2 decode. The C2 decoder is intended for the correction of burst errors that the C1 decoder could not correct. If the C2 decoder cannot correct these errors, the 24-byte codeword is passed on unchanged to the ~ deinterleaver and the associated positions are given an erasure flag via the dashed output lines, Bob ... , Bo24·

ɸ deinterleave. The final operation deinterleaves uncorrectable but detected byte errors in such a way that interpolation can be used between reliable neighboring samples.

4.4 TURBO CODES

Powerful codes uses concatenation.Turbo codes finds its origin in the will to compensate for the dissymmetry of the concatenated decoder.In this concept of feedback is used.

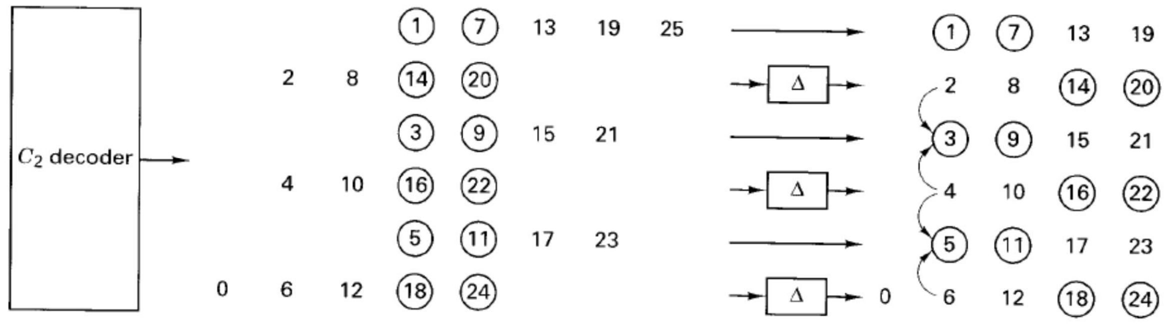


Fig. Effect of interleaving. (Rightmost event is at the earliest time.)

A refinement of the concatenated encoding structure plus an iterative algorithm for the decoding the associated code sequence. Introduced in 1993 by Berrou, Glavieus & Thitimashime. Achieved a BER of 10^{-5}, with rate ½ over AWGN channel & BPSK modulation at E_b/N_0=0.7 dB.

Uses soft decisions information between between the two decoders and iterates it several times to produce more reliable decisions.

4.4.1 Turbo Code Concepts

Likelihood Functions

The mathematical foundations of hypothesis testing rests on Bayes' theorem. For communications engineering, where applications involving an AWGN channel are of great interest, the most useful form of Bayes' theorem expresses the a posteriori probability (APP) of a decision in terms of a continuous-valued random variable x as

$$P(d = i|x) = \frac{p(x|d = i)\, P(d = i)}{p(x)} \quad i = 1, \cdots, M$$

and

$$p(x) = \sum_{i=1}^{M} p(x|d = i)\, P(d = i)$$

where P (d = i/x) is the APP, and d = i represents data d belonging to the ith signal class from a set of M classes. Further, p(x ld = i) represents the probability density function (pdf) of a received contir:uous-valued data-plus-noise signal x, conditioned on the signal class d = i. Also, p(d = i), called the a priori probability, is the probability of occurrence of the ith signal class. Typically x is an "observable" random variable or a test statistic that is obtained at the output of a demodulator or some other signal processor. Therefore, p(x) is the pdf of the received signal x, yielding the test statistic over the entire space of signal classes. In the above equation, for a particular observation, p(x) is a scaling factor since it is obtained by averaging over all the classes in the space. Lower case p is used to designate the pdf of a continuous-valued random variable, and upper case P is used to designate probability (a priori and APP).