Introduction to Information Theory and Coding

(DC II)

Dirk Dahlhaus

Communications Laboratory [ComLab]

Winter Semester 2018/2019

Organisation

- time: in winter semester, annually, Tuesday: 8:00 am-10:00 am/Thursday: 8:00 am-10:00 am
- place: 1332/0425
- workload: 60 hours course attendance,120 hours self-study
- language: English, oral exam (30 minutes either in English or in German)
- exercises: are intergrated in the lecture (on demand)
- regular attendance of the lecture *and* the exercises is mandatory to pass the exam
- the lecture is based on the books (well-known in communications)
  - **John G. Proakis, „Digital Communications“, McGraw-Hill, 4th ed., ISBN 0-07-118183-0**
  - **Thomas M. Cover and Joy A. Thomas, „Elements of Information Theory“, Wiley, 2nd ed., ISBN 0-471-24195-4**
  - **Papoulis S. U. Pillai, „Probability, Random Variables, and Stochastic Processes“, McGraw-Hill, 4th ed.,ISBN 0071226613**
- upon passing the oral exam (5 credit) as well as the Introduction to Information Theory and Coding Lab (1 credit), you obtain **6 credit points for lecture and exercises**

Table of Contents

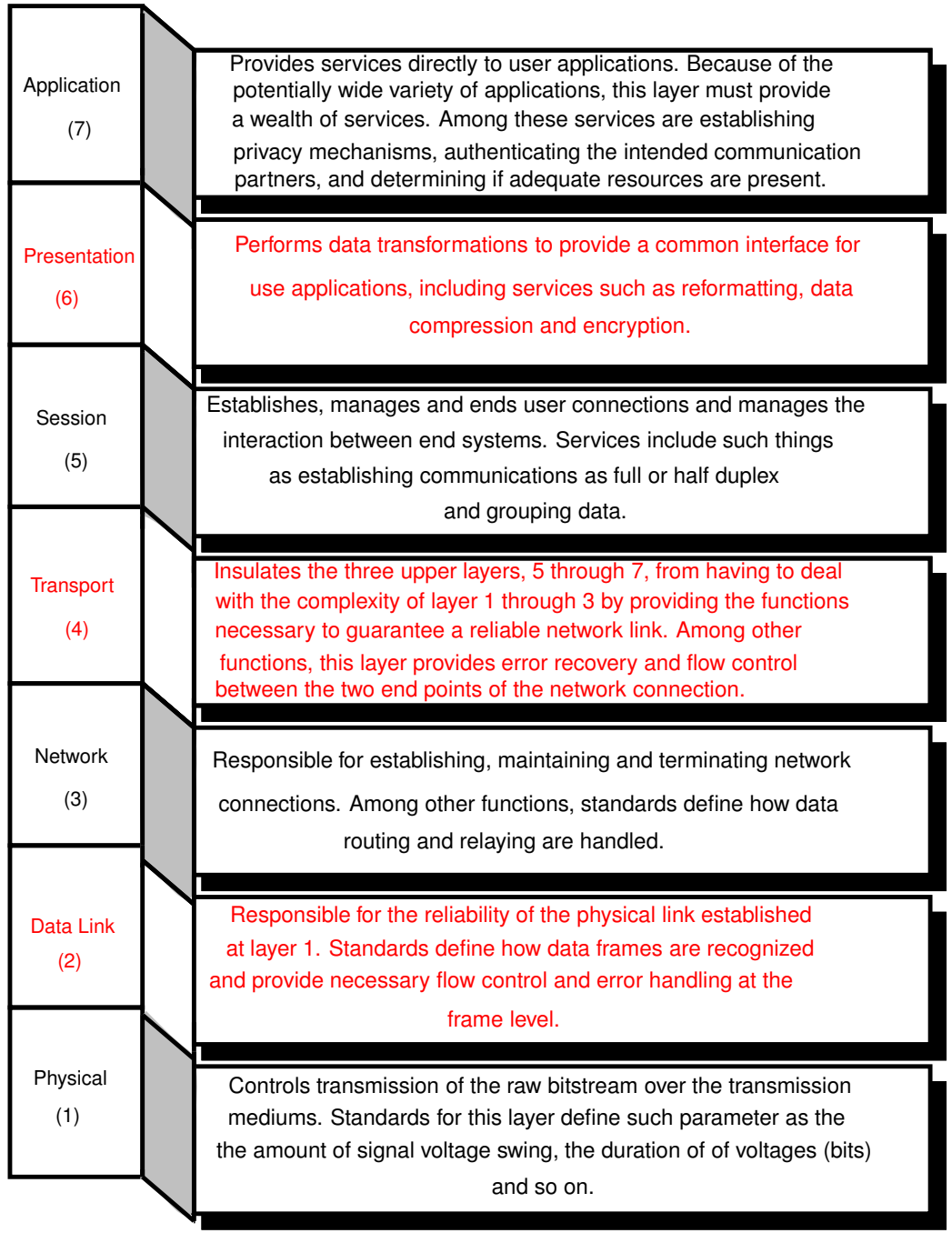
Table of Contents

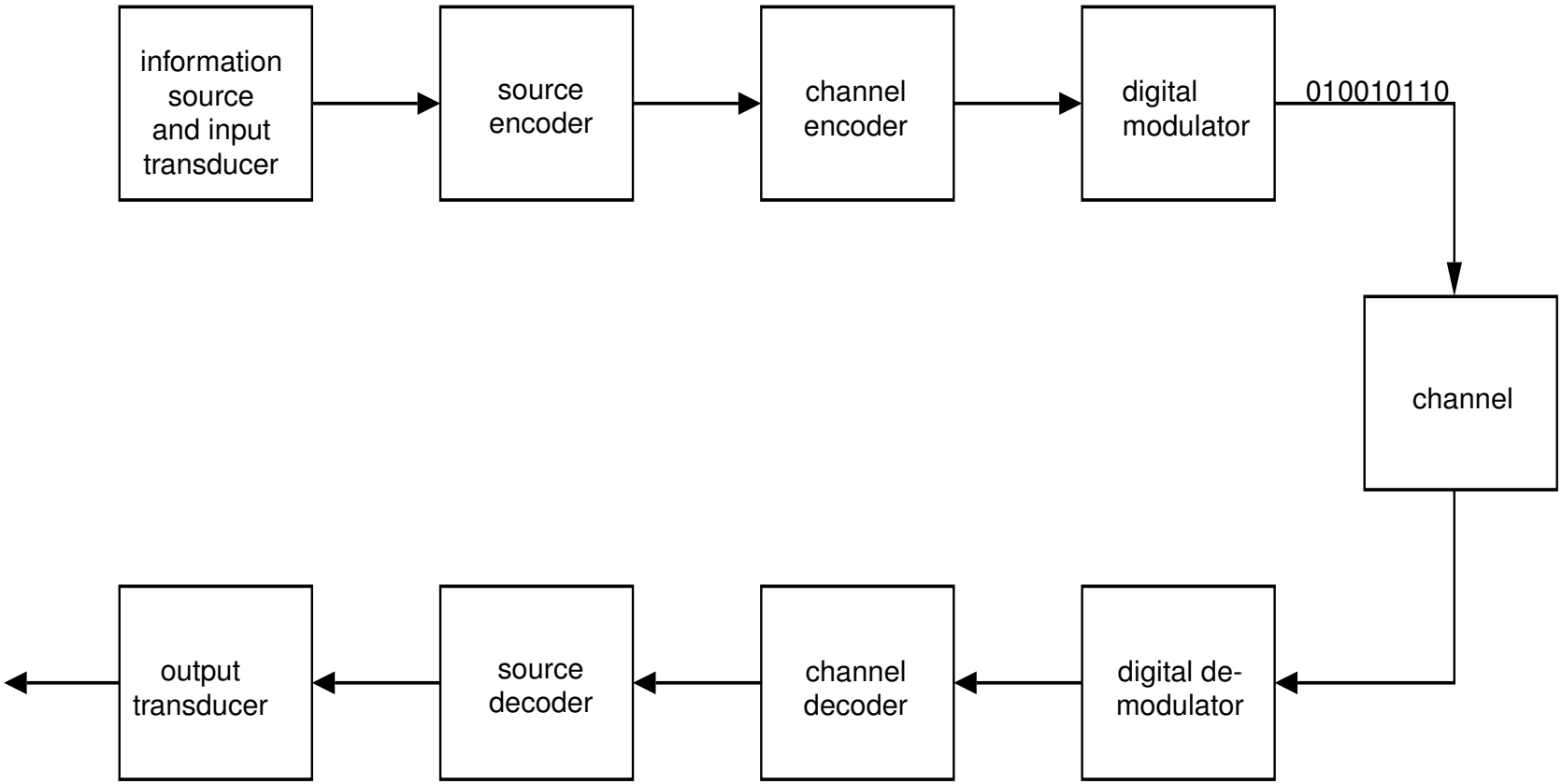
Introduction

- What is *Communications* all about?
- Examples:
  - wireless programming of a cardiac pace maker
  - playing a compact disc (CD) or digital versatile disc (DVD)
  - reading bar codes (super market) and radio frequency identification (RFID) tags (ski lift etc.)
  - mobile communications in cellular radio systems
  - broadband communications (back bone for internet traffic, data base access, etc.)
  - satellite and deep space communications
- Are there any commonalities to these examples which allow them to be classified in a uniform way?

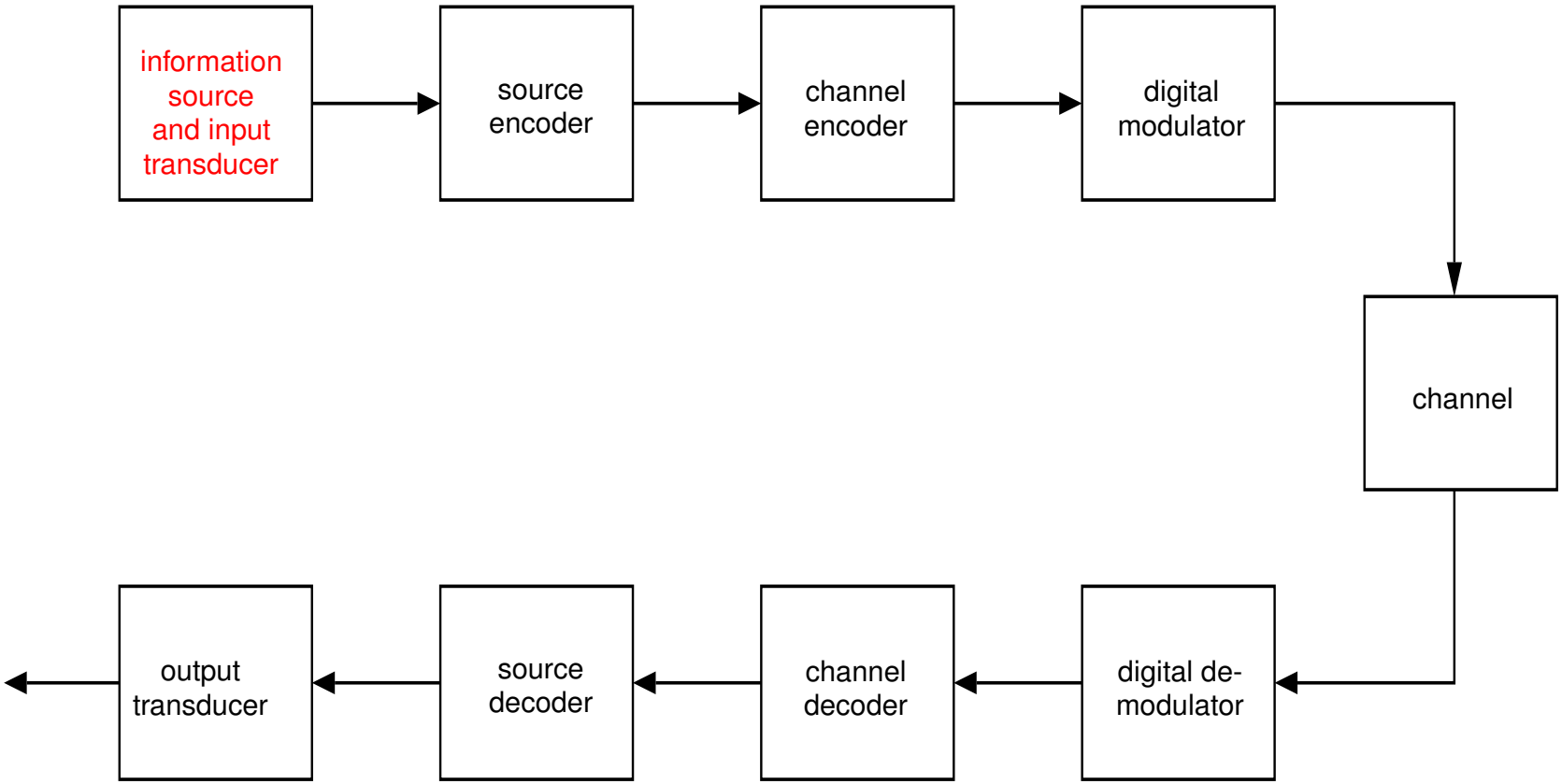
Overview

- Open Systems Interconnection (OSI) Reference Model for standardization of different approaches in communication systems
- in the OSI model, a layer offers a certain service to higher layers
- here, consider exclusively the so-called *physical layer* (PHY), i.e. the transmission from a certain information source or transmitter (TX) to an information drain or receiver (RX)

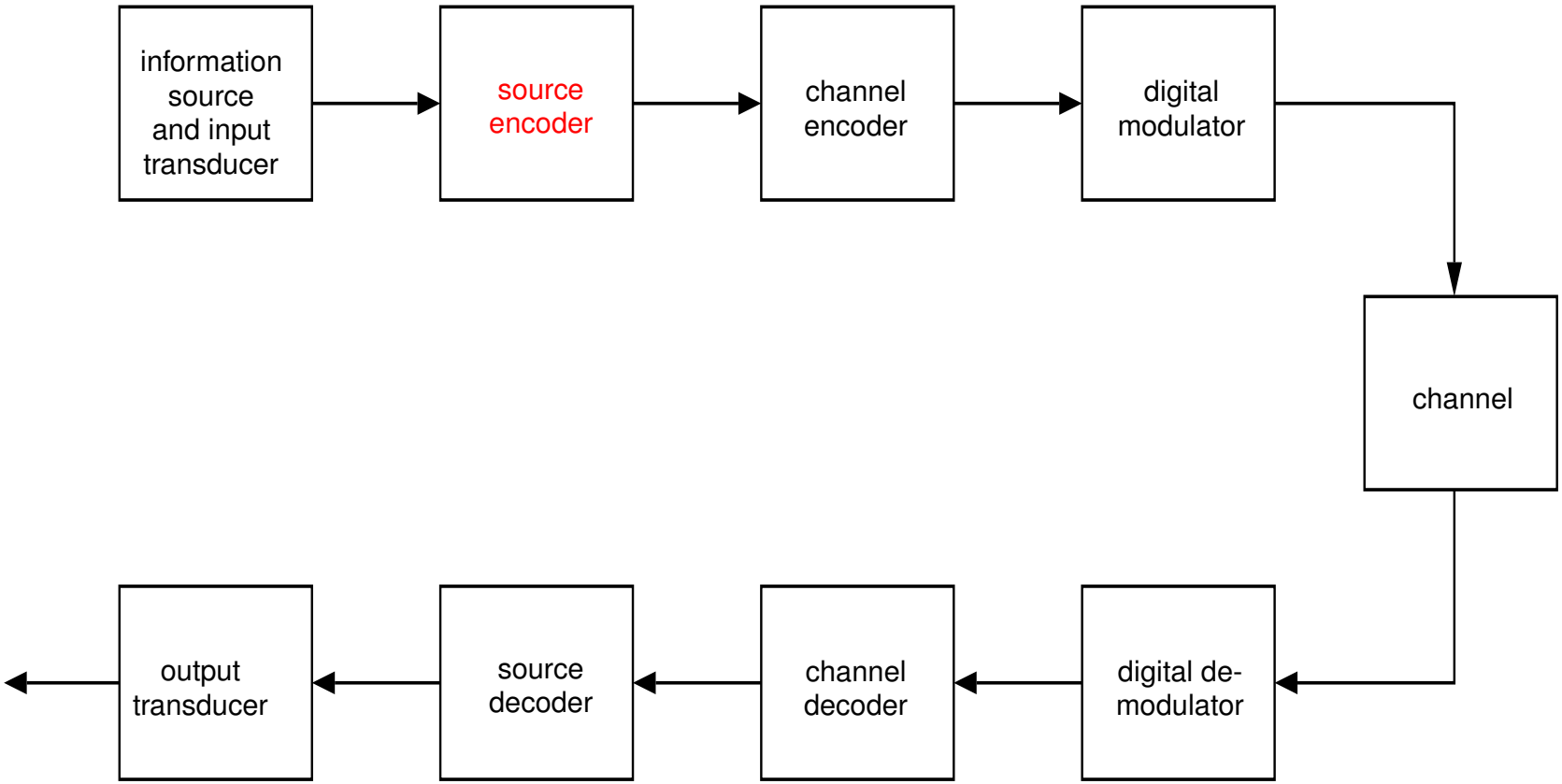




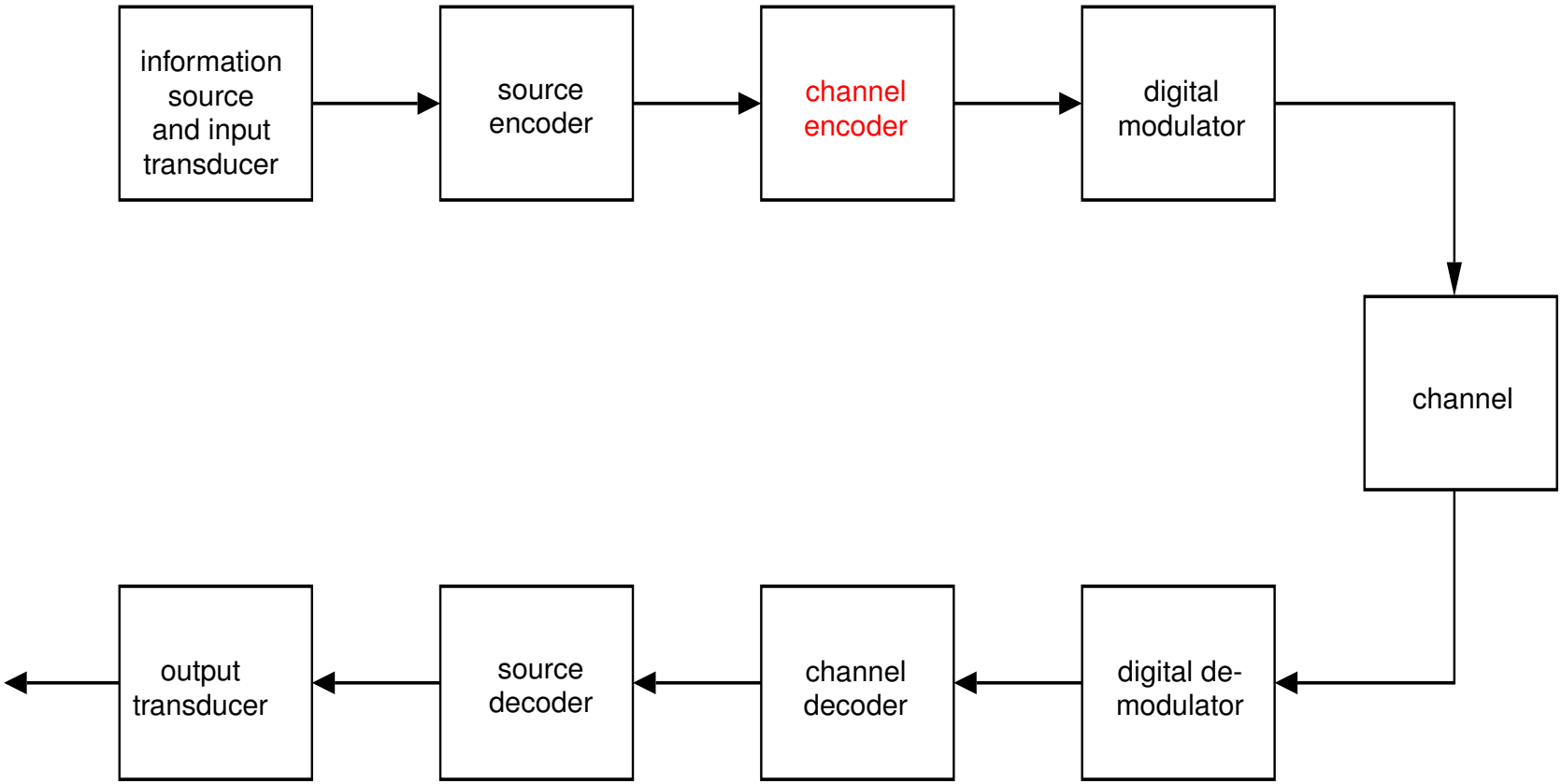
Generic model of a digital communication system.



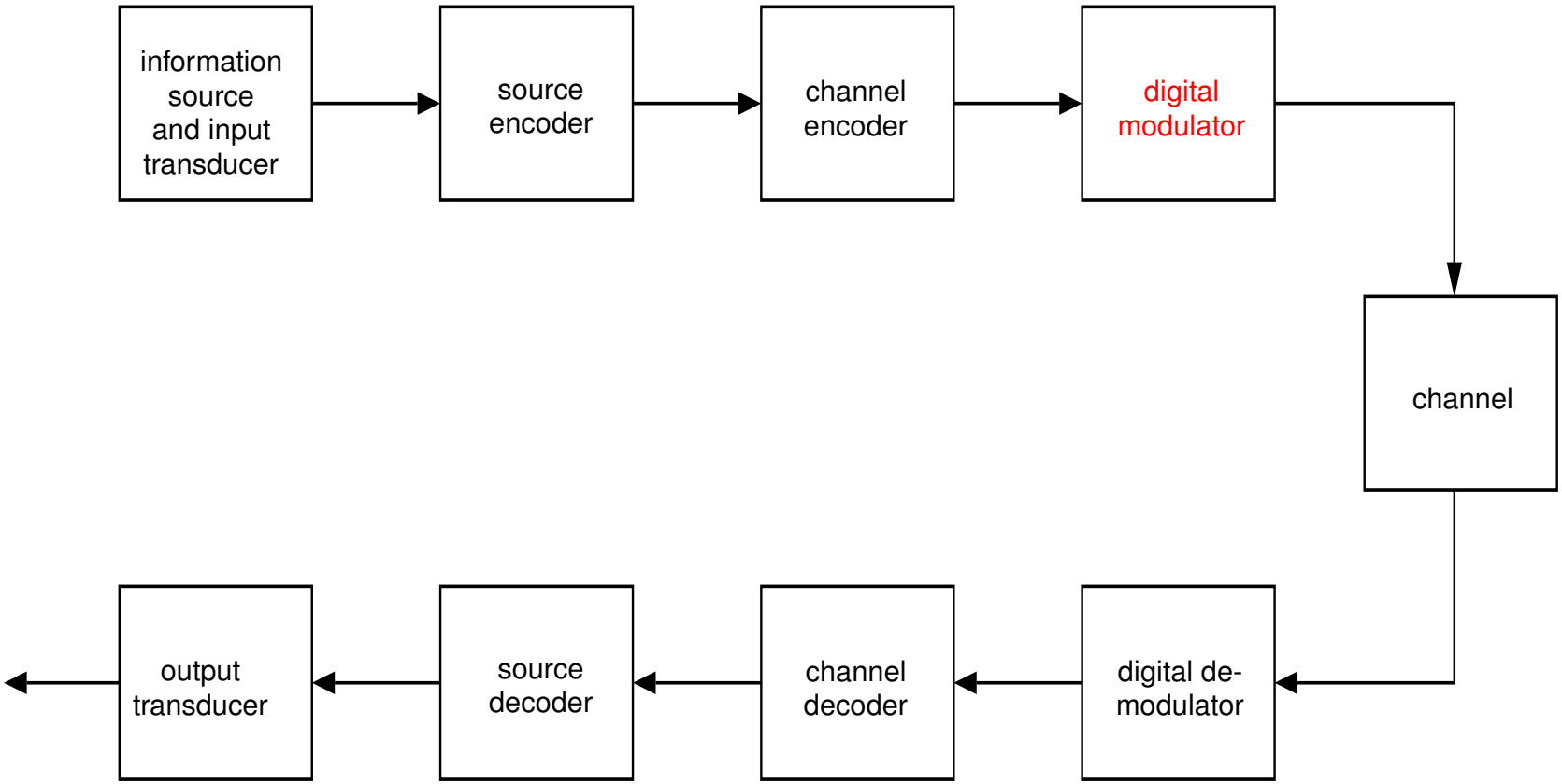
Generic model of a digital communication system.



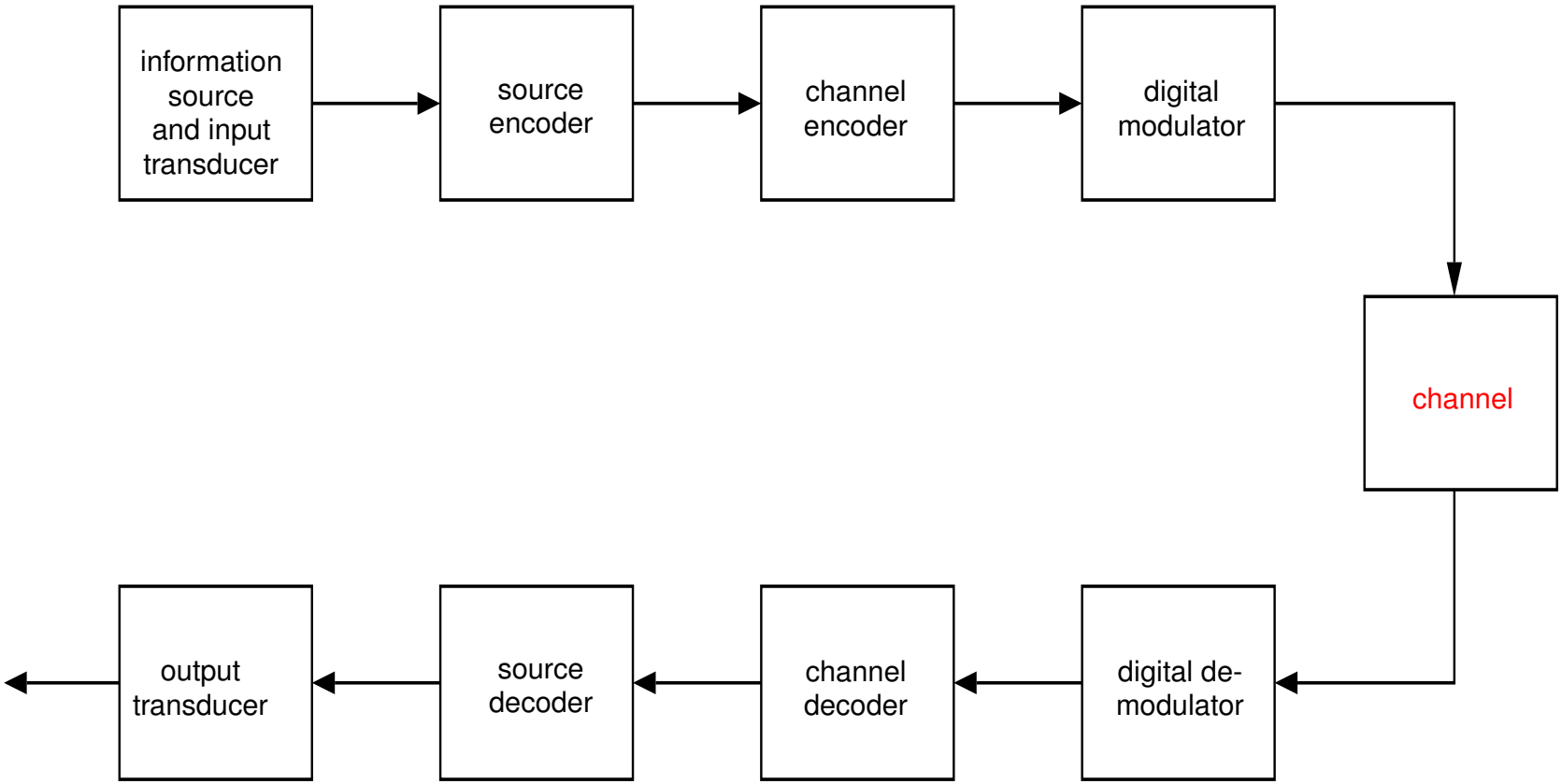
Generic model of a digital communication system.



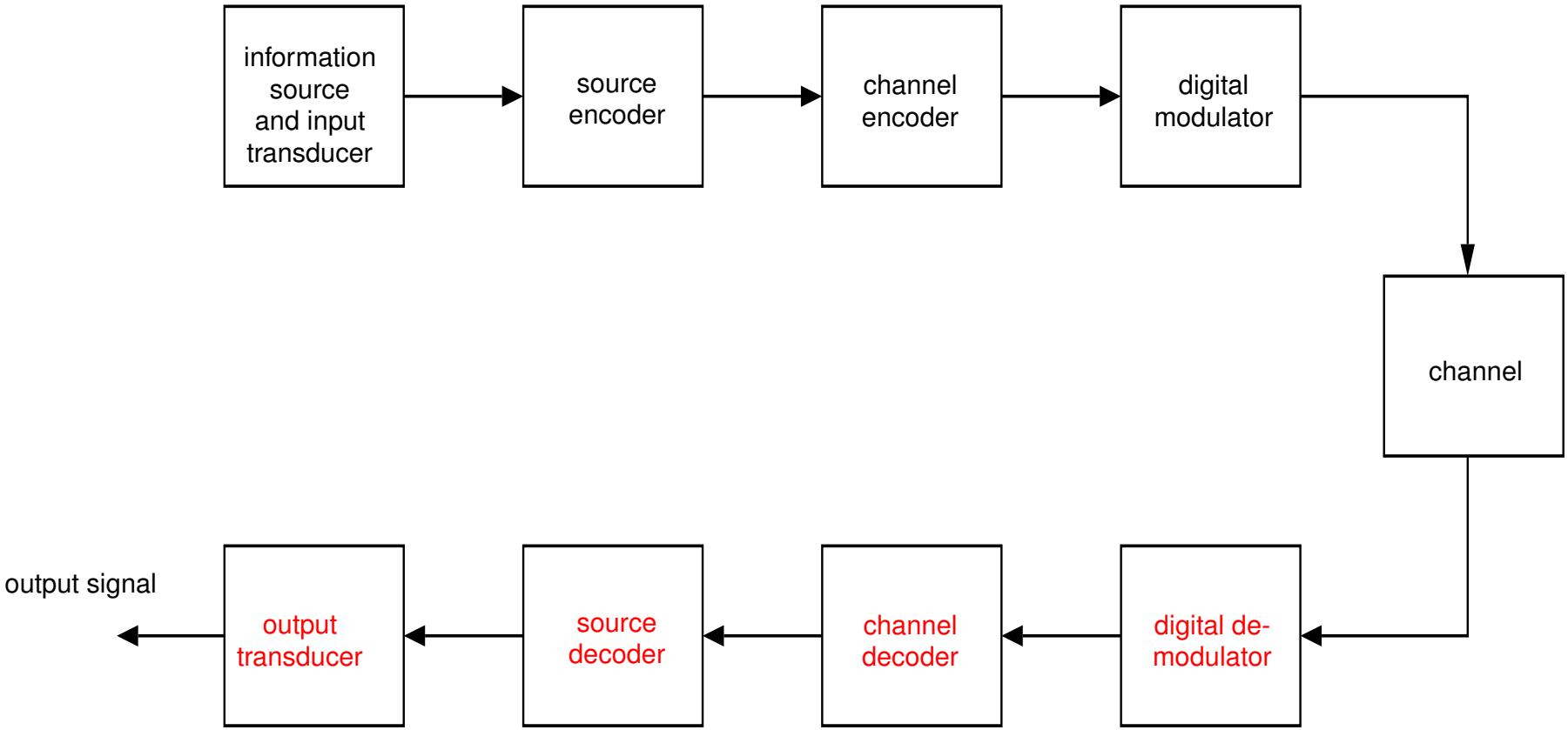
Generic model of a digital communication system.



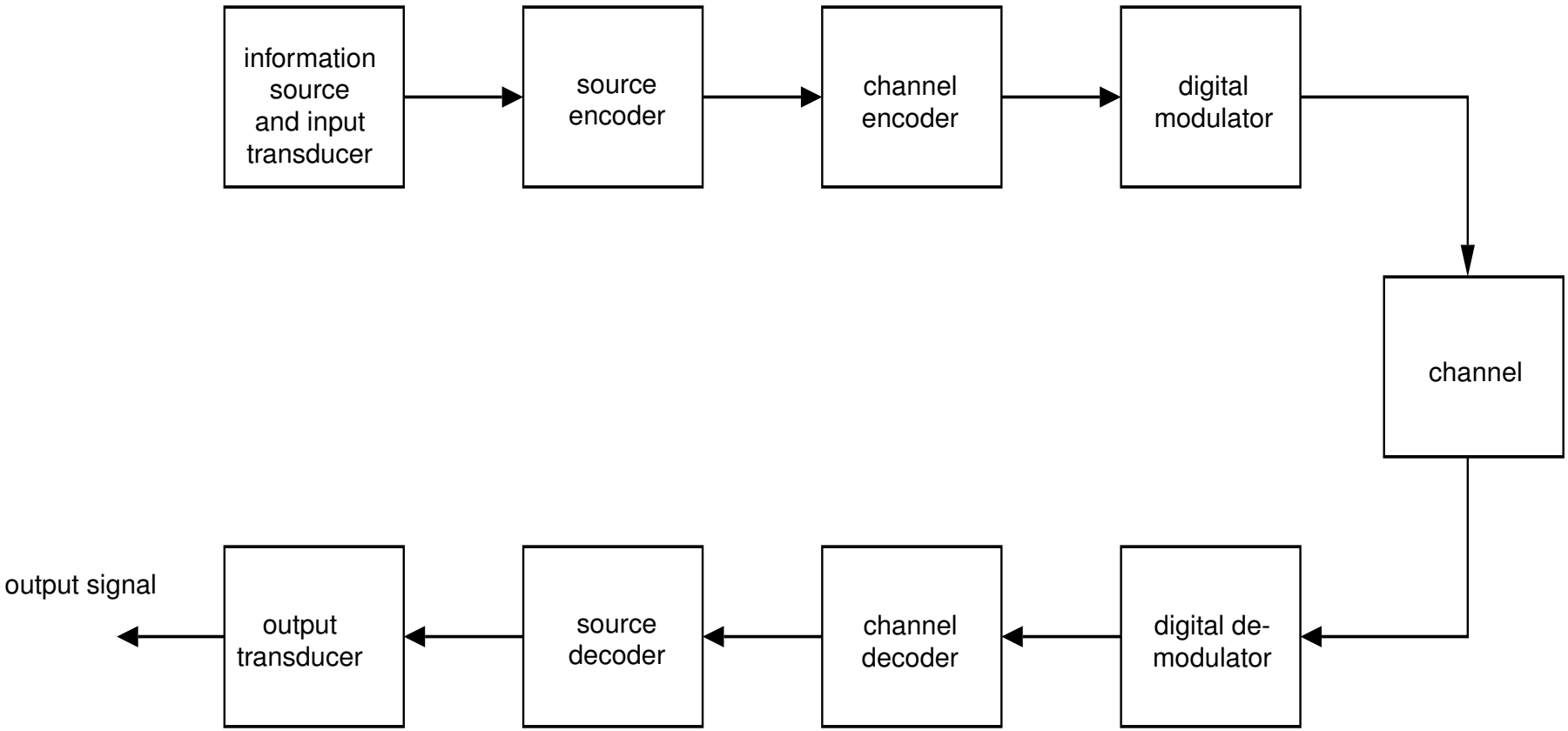
Generic model of a digital communication system.



Generic model of a digital communication system.

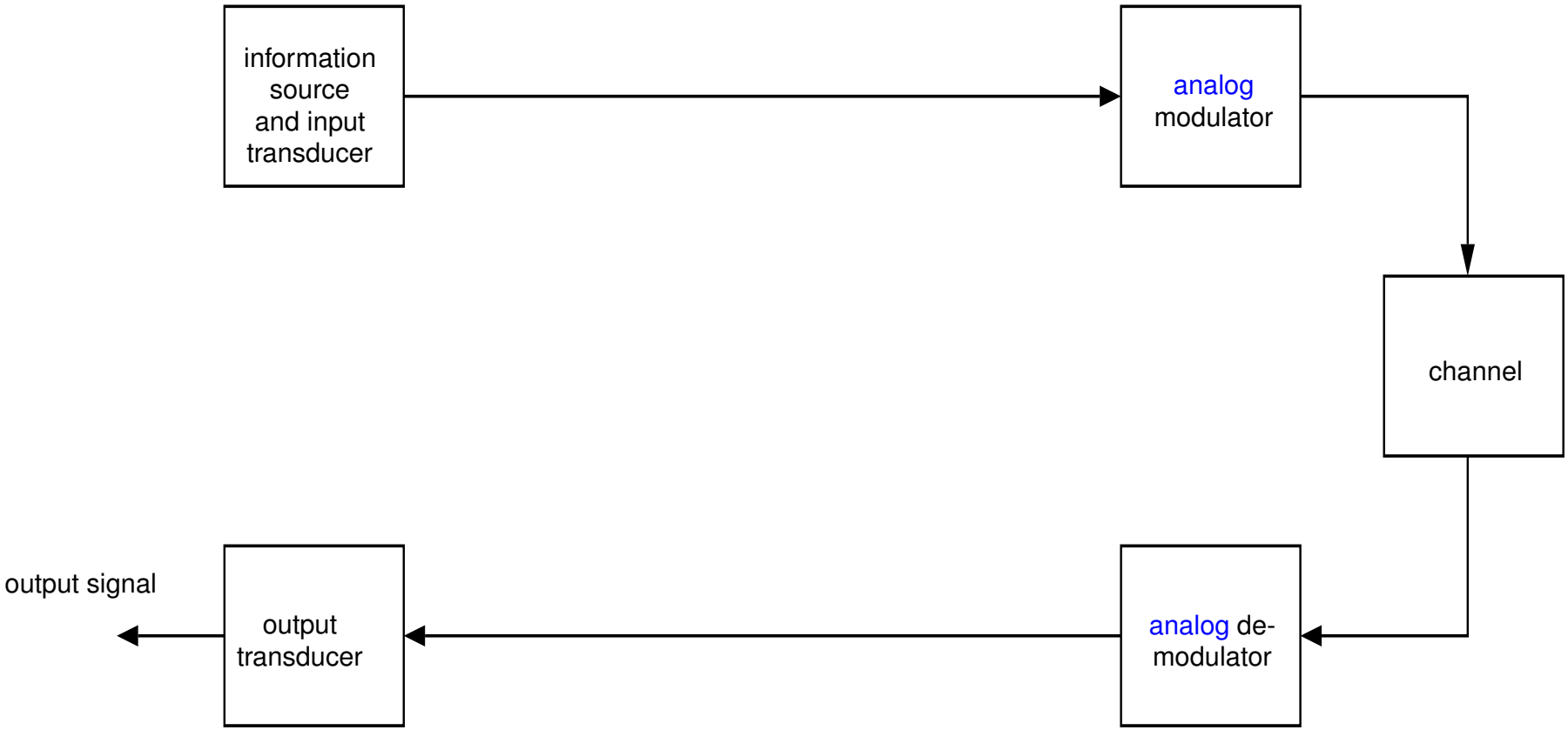


Generic model of a digital communication system.



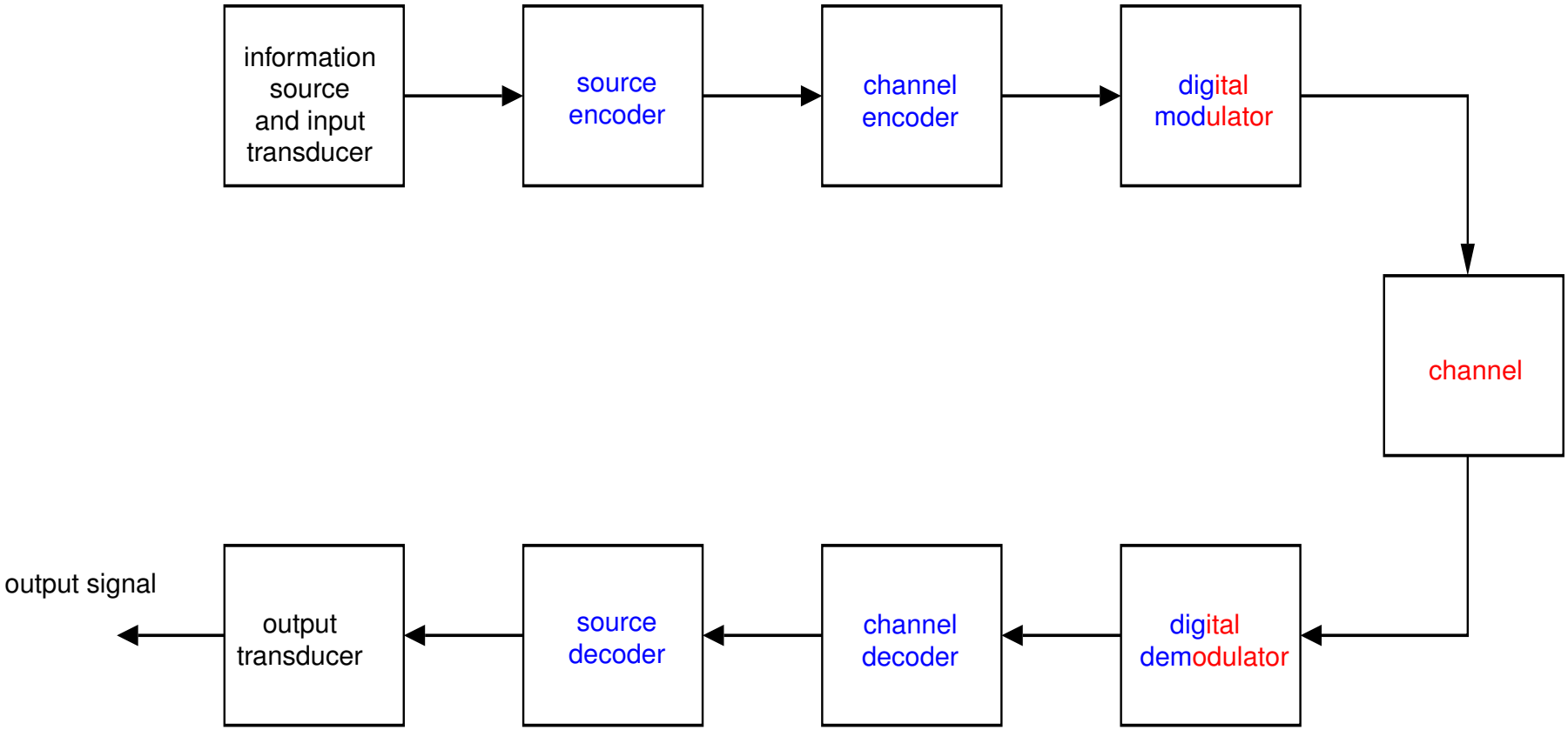
Generic model of a digital communication system.

Are there alternatives to digital communications?



Analog communication system.

Scope of the lectures *Introduction to Information Theory and Coding (DCII)* and *Introduction to Digital Communications (DCI)*:



Generic model of a digital communication system.

Before we discuss the blue parts, we will recap the red parts which have been treated in *Introduction to Digital Communications (DCI)*.

Mathematical Models for Communication Channels

Important questions:

- Is there a generic model for describing transmission channels?
- What are the limitations of the model?
- Which tools are required for a proper description of the model?

Before answering the questions above, we consider a number of channels typically encountered in digital communication systems:

1. wireline channels in telephone networks
2. fibre-optic channels in broadband backbone networks
3. wireless channels in (mobile) radio systems
4. acoustic channels in underwater digital communication systems
5. storage channels in data storing systems.

$\Rightarrow$ common feature of these channels: linearity of the transmission medium

The output signal $r(t)$ of a linear channel for an excitation with input signal $s(t)$ results to

$$r(t) = \int_{\mathbb{R}} k(t;u)s(u)\mathrm{d}u$$

where the expression $k(t;u)$

- is the so-called kernel of the linear system
- and represents the system response to the input signal $\delta(t-u)$.

Analogous in linear algebra with vectors

$$\begin{aligned} \mathbf{s} &= [s_1, \ldots, s_M]^{\mathrm{T}} \\ \mathbf{r} &= [r_1, \ldots, r_N]^{\mathrm{T}} \end{aligned}$$

and $(N \times M)$-dimensional matrix $\mathbf{K} = \{K\}_{N,M}$ of suitable finite dimensions M and N:

$$r(t) = \int_{\mathbb{R}} k(t;u)s(u)\mathrm{d}u \qquad \Leftrightarrow \qquad \mathbf{r} = \mathbf{K}\mathbf{s}.$$

Observations:

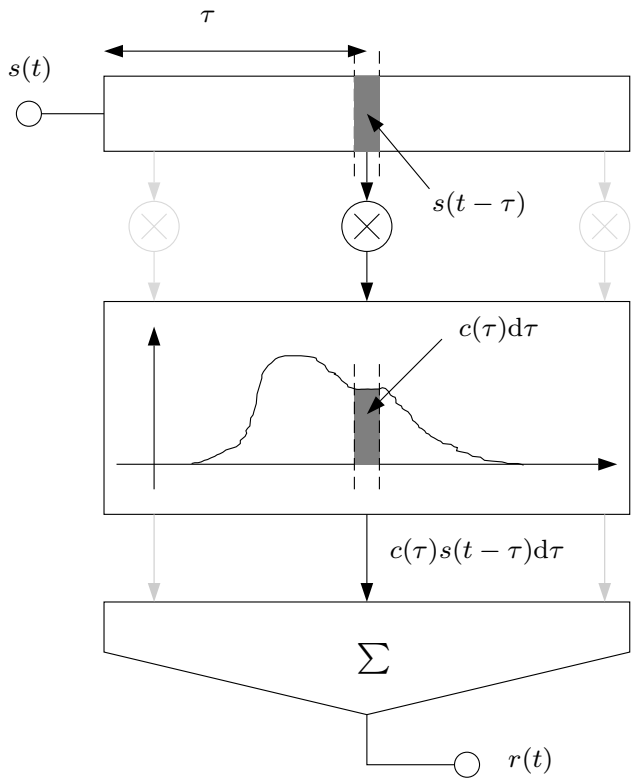
- Obviously, the linear operator (matrix) $\mathbf{K}$ corresponds to $k(t;u)\mathrm{d}u$.
- The variable t corresponds to the *row* index of the matrix $\mathbf{K}$, the variable u to the column index.
- The matrix-vector product $\mathbf{Ks}$ (sum of products $K_{t,u}s_u$) corresponds to $\int_{\mathbb{R}} k(t;u)s(u)\mathrm{d}u$ (integral of products $k(t;u)\mathrm{d}u\ s(u)$).

For time-invariant systems, we have $k(t;u) = k(t-u)$ and the resulting integral represents a *convolution*:

$$r(t) = k(t) \star s(t) = \int_{\mathbb{R}} k(t-u)s(u)\mathrm{d}u = s(t) \star k(t).$$

Upon substitution $\tau = t - u \Rightarrow$ interpret $c(\tau) = k(\tau)$ as a transversal filter:

$$r(t) = \int_{\mathbb{R}} c(\tau)s(t-\tau)\mathrm{d}\tau$$



Interpretation of convolutional integral as transversal filter.

Since $r(t) = c(t-u)$ for $s(t) = \delta(t-u) \Rightarrow\ c(t)$ represents the channel impulse response.

Consider the Fourier transform of the convolutional integral given by

$$R(f) = \mathcal{F}\{r(t)\} = \mathcal{F}\{c(t) \star s(t)\} = \mathcal{F}\{c(t)\}\,\mathcal{F}\{s(t)\} = C(f)S(f).$$

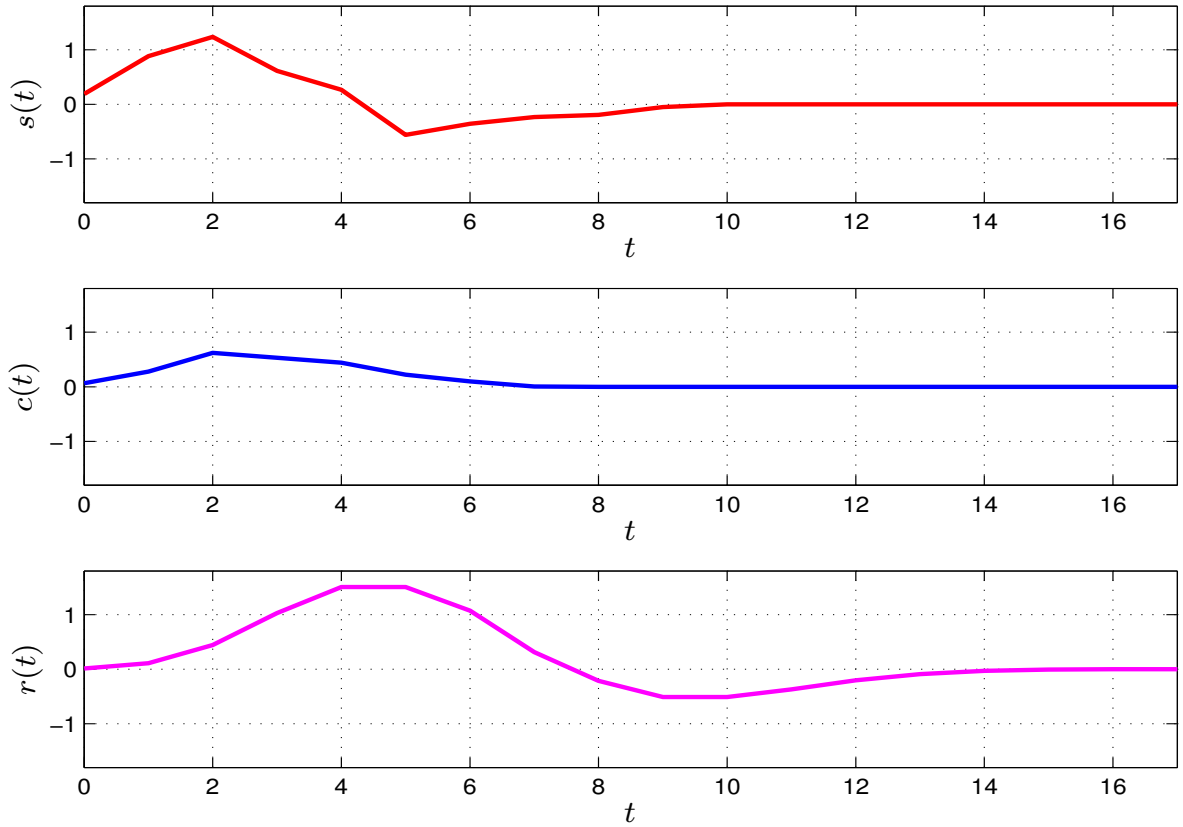
Obviously, the time signal $\sigma(f_0; t) = \exp(\jmath 2\pi f_0 t)$ is an *eigenfunction* of the channel. Clearly, if $\sigma(f_0; t)$ is the channel input signal, the output is proportional to $\sigma(f_0; t)$ as follows from

$$R(f)|_{s(t)=\sigma(f_0;t)} = C(f)\delta(f - f_0) \bullet\!\!-\!\!\circ\, r(t) = C(f_0)\exp(\jmath 2\pi f_0 t).$$

Here, $C(f_0)$ is the *eigenvalue* of the linear time-invariant (LTI) channel for the eigenfunction $\sigma(f_0; t)$ and is known as the transfer function of the channel

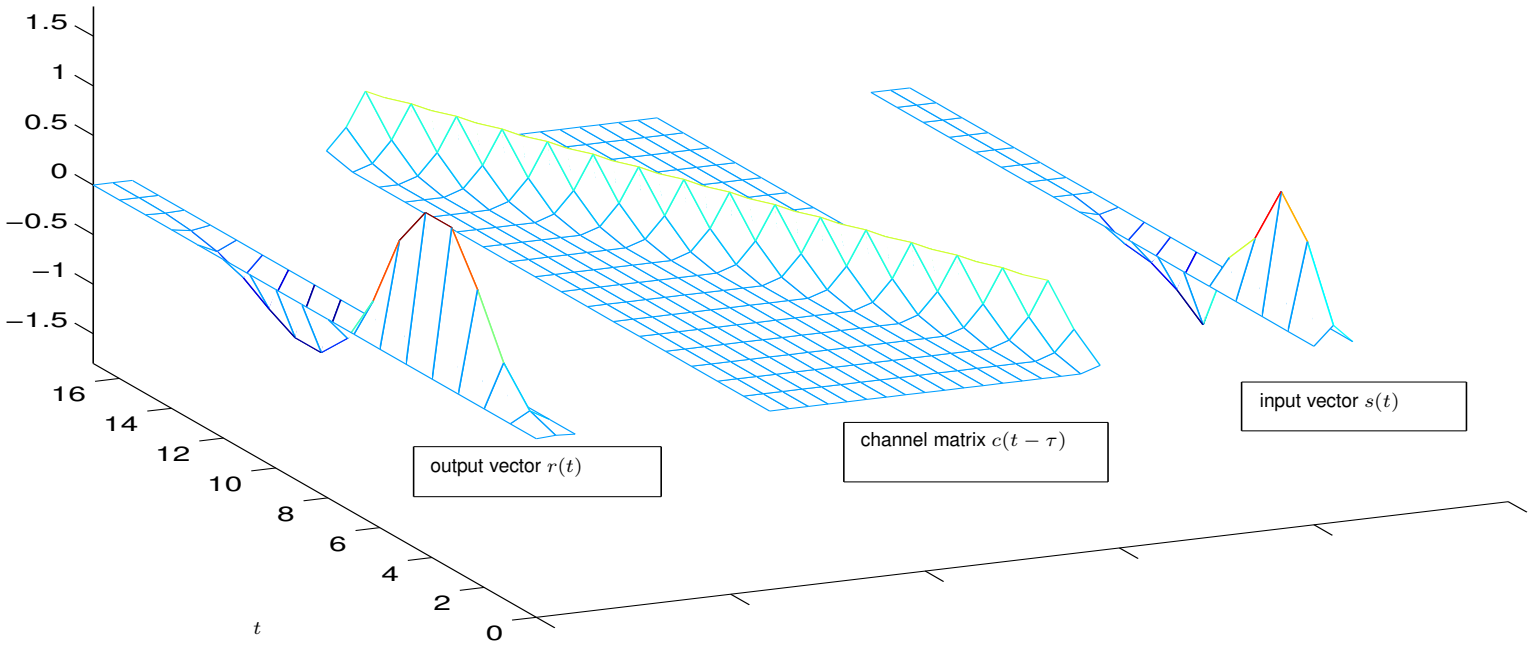
Nota Bene (N.B.): considerations of eigenfunctions and eigenvalues are completely analogous to the corresponding notions in linear algebra, i.e. eigenvectors and eigenvalues

Example: convolution of signals with finite temporal support, usual representation



Convolutional integral.

Example: convolution of signals, vector-matrix representation



Convolutional integral.

Since we can interpret

$$s(t) = \int_{\mathbb{R}} S(f) \exp(\jmath 2\pi f t) \mathrm{d}f$$

as the linear superposition of complex exponentials $\exp(\jmath 2\pi t)$ weighted by complex numbers $S(f)\mathrm{d}f$, the convolution can be interpreted as follows:

For arbitrary $c(t)$, each frequency component $S(f)$ of $s(t)$ is weighted by a complex number $C(f)$ and the resulting components $R(f) = C(f)S(f)$ are linearly superimposed to give the output signal $r(t) = \int_{\mathbb{R}} R(f) \exp(\jmath 2\pi f t) \mathrm{d}f$.

This property makes the Fourier transform a powerful tool in the characterization of LTI systems.

For time-variant channels, the time-invariant channel impulse response $c(t)$ is not sufficient for the channel description. Instead, upon substituting $\tau = t - u$, we obtain in analogy with the time-invariant case

$$\begin{aligned} r(t) &= \int_{\mathbb{R}} k(t;u)s(u)\mathrm{d}u \\ &= \int_{\mathbb{R}} k(t;t-\tau)s(t-\tau)\mathrm{d}\tau \\ &= \int_{\mathbb{R}} c(\tau;t)s(t-\tau)\mathrm{d}\tau \end{aligned}$$

with $c(\tau;t) = k(t;t-\tau)$ denoting the time-variant channel impulse response at time t to a Dirac pulse applied at time $t-\tau$. The last integral expression is a direct extension of the usual convolution integral (arising in the time-invariant case) to a time-variant system behaviour.

Some remarks on the aforementioned linear channel models:

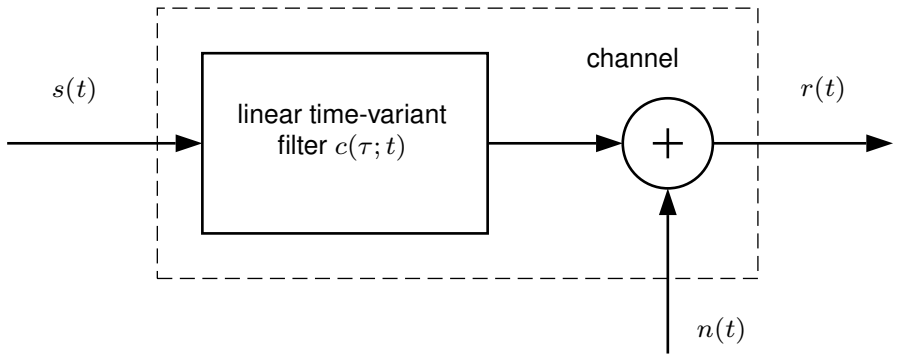
- The channel impulse responses (CIRs) above (either time-variant or time-invariant) are assumed to represent a real-valued or complex-valued function defined on $\mathbb{R}$ (in the case of $c(\tau)$) or $\mathbb{R}^2$ (in the case of $c(\tau; t)$).

- If the received signal was indeed given by $r(t) = k(t) \star s(t)$, the transceiver design would be rather simple and close to the case of the perfect channel $k(t) = \delta(t)$ (except for spectral zeros, we could, for example, equalize the channel in the frequency domain by multiplying $R(\jmath\omega)$ by $1/C(\jmath\omega)$ and taking the inverse Fourier transform afterwards).

- In most cases, however, we do not know about the exact form of the CIR, but still have to design the transceiver such that its performance complies with the specifications for the system at hand. The CIR is subject to *random* effects such as fading, scattering from objects and Doppler effects in wireless systems, random disturbances in front-ends, drift effects etc., so that, in general, the CIR is rather to be characterized as a stochastic process.

- Apart from the transmission channel itself, the receiver is a source of noise which limits the information rate and performance and thus has to be taken into account in the transceiver design. It is one of the major contributions of information theory (initiated by the work of Shannon in the 1940s) to design schemes to overcome the impact of random disturbances in a digital communication system.

- A simple standard approach encountered in practical systems to model noise in receivers is additive noise.



Linear time-variant channel with addditive noise model.

- Again, as in the case of random effects in the transmission medium, the noise $n(t)$ represents a random process and the received signals in a practical system are instances (sometimes also termed realizations) of the corresponding processes.

- Summarizing the comments above, most channels that we will deal with lead to a model of the received signal defined by

$$r(t) = \int_{\mathbb{R}} c(\tau; t) s(t - \tau) \mathrm{d}\tau + n(t)$$

with $c(\tau; t)$ and $n(t)$ representing stochastic processes with properties which are to be chosen for the system at hand.

- Other sources of disturbances include interference from other communication links (in particular in wireless transmission), cross-talk (telephone lines), amplifier saturation effects leading to non-linear channel properties, intermodulation, synchronization errors etc. The aforementioned model represents a reasonable trade-off between simplicity and accuracy and can often be extended to include further channel properties which are critical for the overall transceiver design.

- In the lecture DC I, we have considered the case where the information to be sent from the transmitter to the receiver is represented by a waveform. The objective of the receiver is to detect what information is represented by the received signal. The detection should be based on the minimization of a suitable cost function such as e.g. the probability of error.

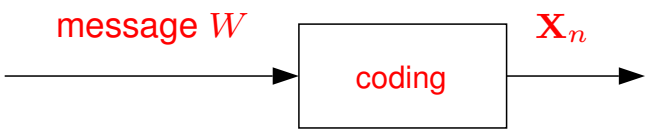
Table of Contents

Efficient Information Transmission

- Until now, we have assumed that information can be transmitted from a source to a destination by transmitting a certain waveform from a predefined set and detecting the signal at the receiver based on the minimization of a certain cost function, e.g. the average probability of symbol error.
- However, we have not yet answered the question what **information** really means and how this quantity can be transmitted efficiently and reliably to a destination.
- Theses topics can be treated using concepts in **information theory**, in particular **channel capacity**.
- We will approach the problem by trying to answer the following question:

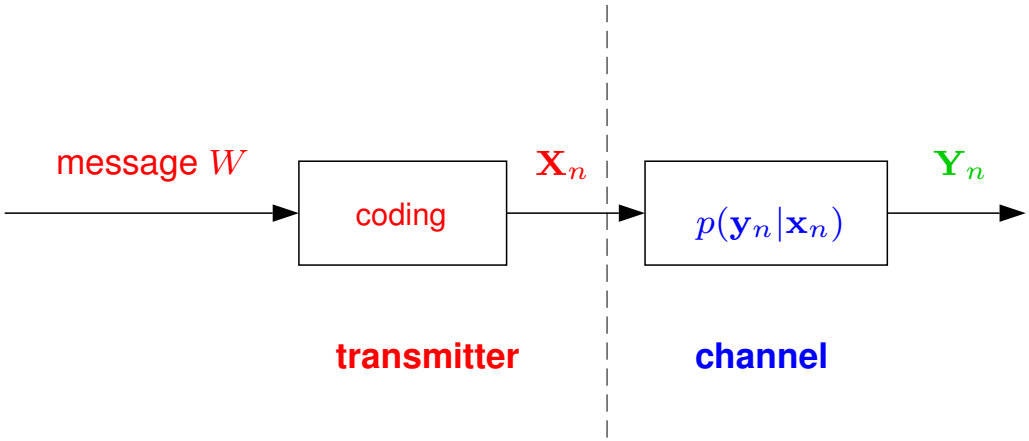
How can information be transmitted efficiently over a channel?

System description



transmitter

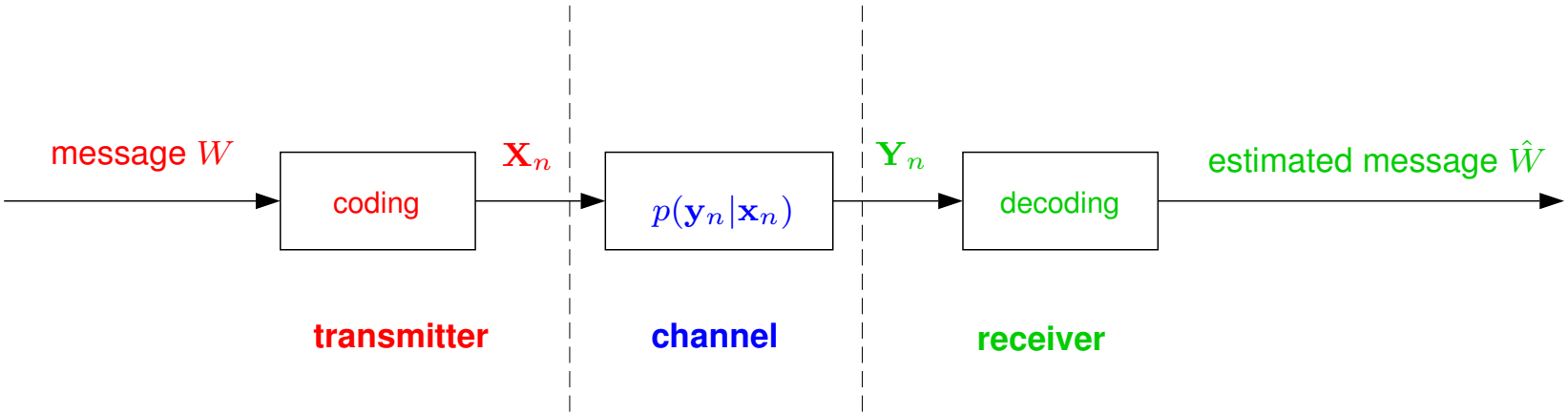
System description



Transmitter: map message W to a sequence $\mathbf{X}_n = X_1, X_2, \ldots, X_n$

Channel: system part (not under control of designer) with transition probabilities $p(\mathbf{y}_n|\mathbf{x}_n)$

System description

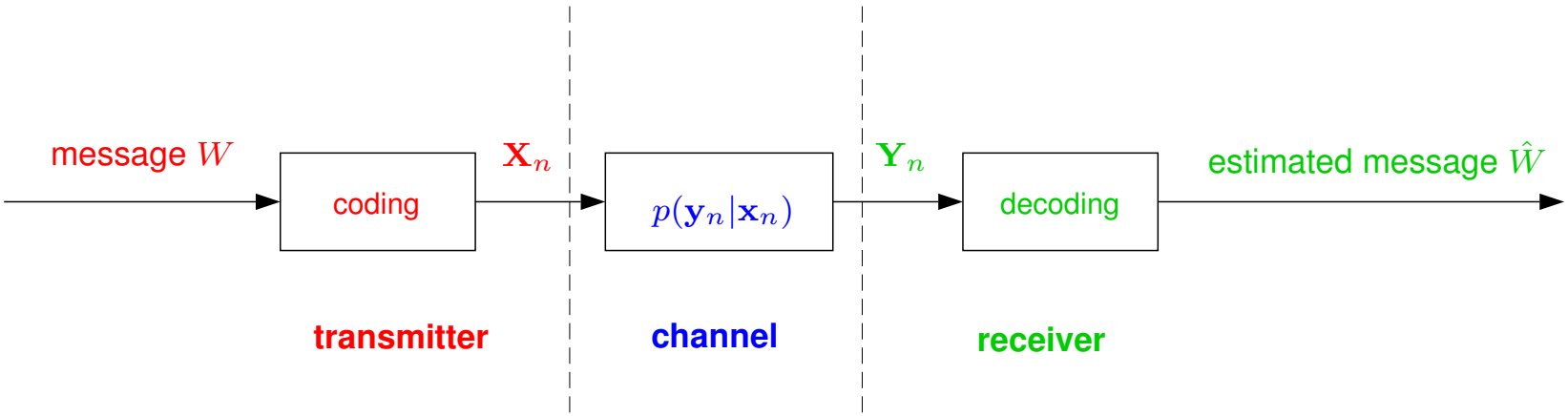


Transmitter: map message W to a sequence $\mathbf{X}_n = X_1, X_2, \ldots, X_n$

Channel: system part (not under control of designer) with transition probabilities $p(\mathbf{y}_n|\mathbf{x}_n)$

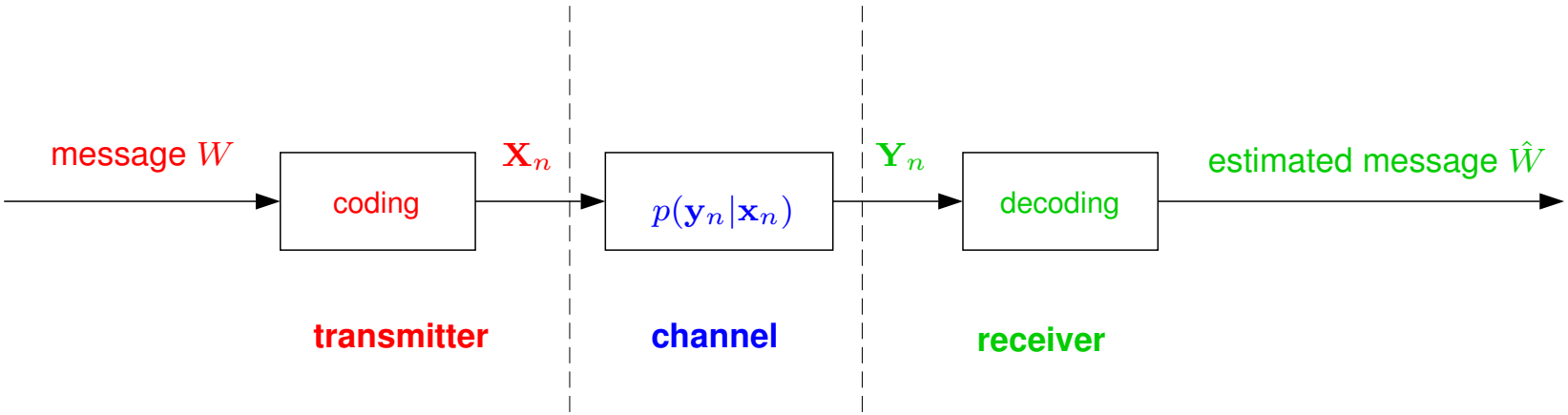
Receiver: map received sequence $\mathbf{Y}_n = Y_1, Y_2, \ldots, Y_n$ to $\hat{W}$

System description



information $\Rightarrow$ (1) *What* is information (of the receiver by the transmitter)?

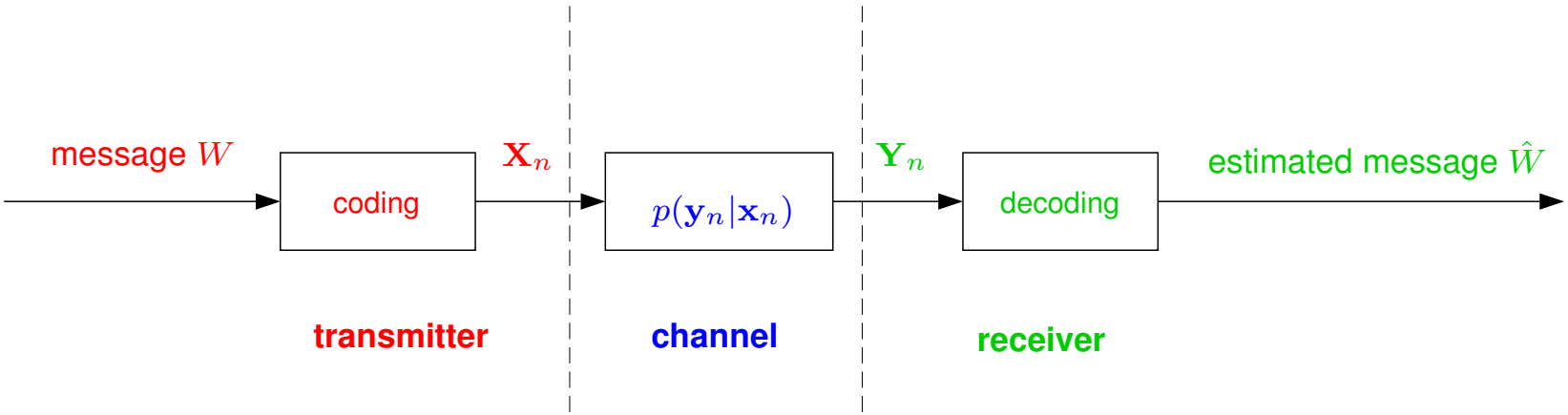
System description



information $\Rightarrow$ (1) *What* is information (of the receiver by the transmitter)?

efficient $\Rightarrow$ (2) *How much* information can be transmitted over the channel?

System description

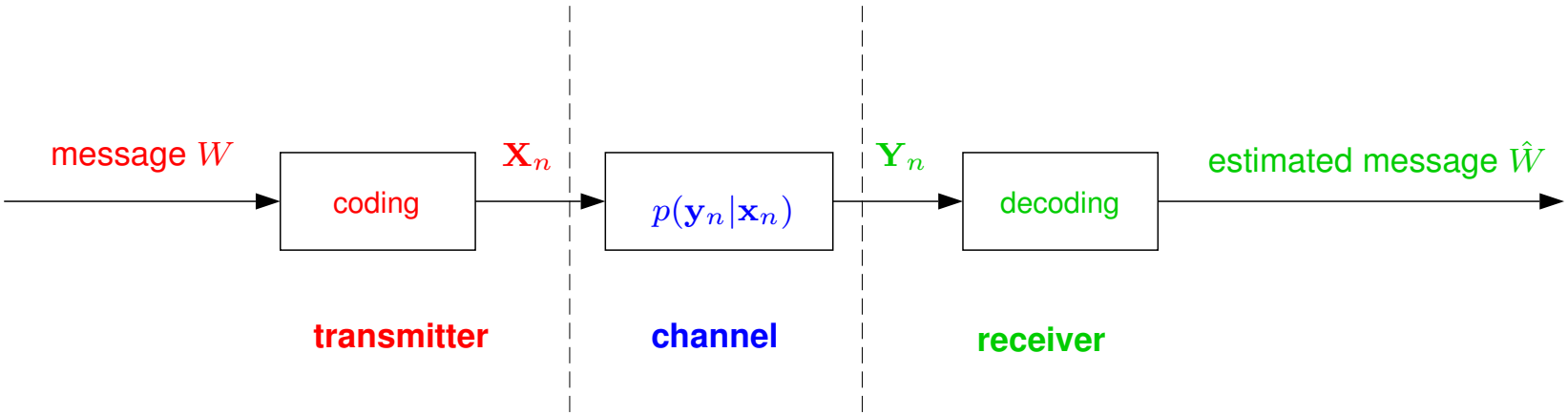


information $\Rightarrow$ (1) *What* is information (of the receiver by the transmitter)?

efficient $\Rightarrow$ (2) *How much* information can be transmitted over the channel?

transmission $\Rightarrow$ (3) *How* can information be transmitted?

System description



information $\Rightarrow$ (1) *What* is information (of the receiver by the transmitter)?

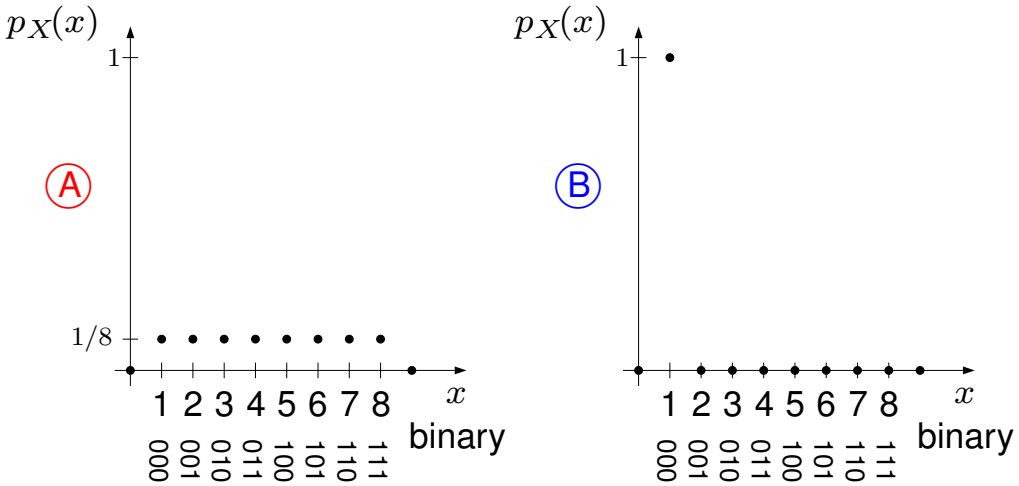
efficient $\Rightarrow$ (2) *How much* information can be transmitted over the channel?

transmission $\Rightarrow$ (3) *How* can information be transmitted?

$\Rightarrow$ **channel capacity**

What is information?

- Example: X = *Bayern München plays against KSV Hessen Kassel 3:1*
  $\Rightarrow$ information content is low, since the result is likely to happen
  X = *Bayern München looses against KSV Hessen Kassel 0:8*
  $\Rightarrow$ information content high, since the result is very unlikely
- Intuition: the lower the probability of a message $X = x_i$, the higher is the information content
- Can we quantify the **average uncertainty** of a random message X?
- We need a measure of uncertainty $f(P(X = x_i)) = f(p_X(x_i))$, which **increases for decreasing probability**



Example (A):
each observation can be described by 3 bit; since all observations are equally likely, the *average* uncertainty is also 3 bit with

$$\mathbf{E}\left\{f(p_X(X))\right\} = \sum_{i=1}^{8} \frac{1}{8} f(p_X(x_i)) = f(p_X(x_i)) = f\left(2^{-3}\right) \stackrel{!}{=} 3 \text{ bit}$$

Example (B):
the drawing $X = 1$ will be observed with probability one, i.e. the uncertainty is equal to zero

$$\mathbf{E}\left\{f(p_X(X))\right\} = f(p_X(1)) = f\left(2^{0}\right) \stackrel{!}{=} 0 \text{ bit}$$

$\Rightarrow$ choose $\boxed{f(p_X(x_i)) = \mathrm{ld}(1/p_X(x_i)) = -\mathrm{ld}(p_X(x_i))}$

Entropy of a discrete random variable

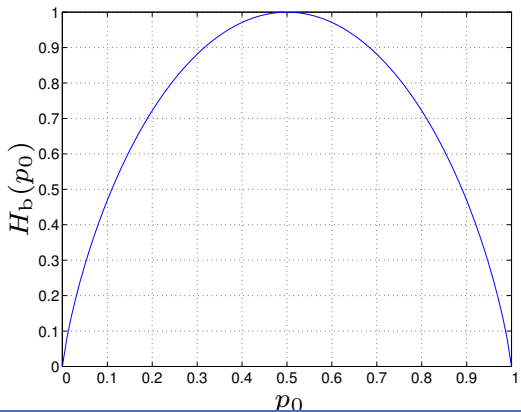
The uncertainty (= entropy) of a discrete random variable X in [bit] is defined by

$$H(X) = \mathbf{E}\left\{-\mathrm{ld}(p_X(X))\right\} = -\sum_{x \in \mathrm{supp}(p_X)} p_X(x)\mathrm{ld}(p_X(x))$$

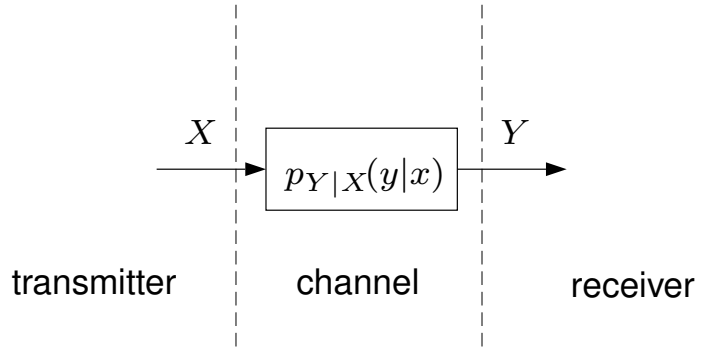
Problem

Using the *information inequality* which says that $\mathrm{ld}(x) \leq (x-1)\mathrm{ld}(\mathrm{e})$ with equality if and only if $x = 1$, show that $0 \leq H(X) \leq \mathrm{ld}(L)$ for $X \in \{x_1, \ldots, x_L\}$ and equality if and only if X is uniformly distributed with $P(X = x_i) = 1/L$ for all i.

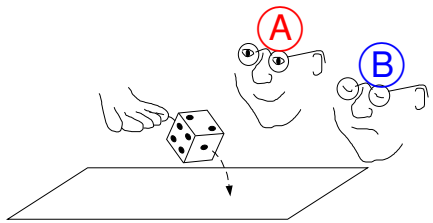
Binary entropy function: Let $X \in \{0, 1\}$ with $P(X = 0) = p_0$ and $P(X = 1) = 1 - p_0$



Entropy of two discrete random variables



Example: observation Y of a die with L faces X by two observers (A) und (B)



(A): upon perfect observation $Y = X$, we can identify X uniquely; thus, **in each tossing,** $\mathrm{ld}(L)$ **bit are transmitted labeling about** $2^{H(X)}$ **addresses**

(B): useless observation Y of the sound (being independent of X) so that X is unknown; as a result, the uncertainty of X upon reception of Y is still $\mathrm{ld}(L)$ bit and no information is being transmitted

Entropy of two discrete random variables and mutual information

Shannon: information = reduction of uncertainty

Thus, we have the following suggestive **definition of transmitted information** $I(X;Y)$:

$$\underbrace{\text{information [bit/use]}}_{I(X;Y)} = \underbrace{\text{entropy of } X}_{H(X)} - \underbrace{\text{entropy of } X \text{ conditioned on } Y}_{H(X|Y)}$$

Here, the uncertainty of X conditioned on Y (conditional entropy) is defined by

$$H(X|Y) = \mathbf{E}\left\{-\mathrm{ld}\left(p_{X|Y}(X|Y)\right)\right\}$$

This definition makes sense, as can be shown (problem):

$$I(X;Y) = H(X) - H(X|Y) \geq 0; \qquad 0 \leq H(X|Y) \leq H(X)$$

Entropy of two discrete random variables and mutual information

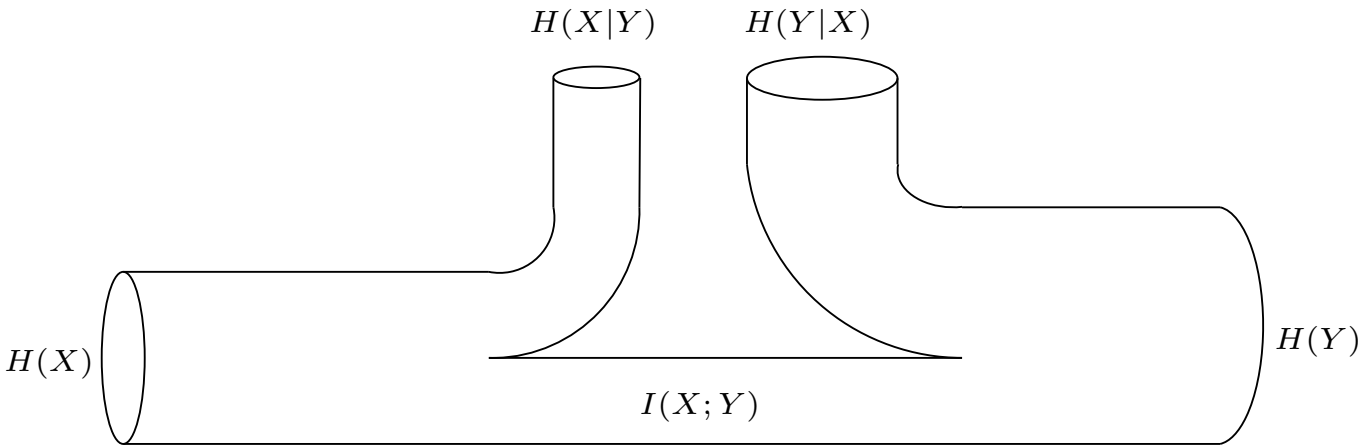
Property of mutual information:

$$
\begin{aligned}
I(X;Y) &= H(X) - H(X|Y) \\
&= \mathbf{E}\left\{-\mathrm{ld}(p_X(X))\right\} - \mathbf{E}\left\{-\mathrm{ld}\big(p_{X|Y}(X|Y)\big)\right\} \\
&= \mathbf{E}\left\{\mathrm{ld}\left(\frac{p_{X|Y}(X|Y)}{p_X(X)}\right)\right\} \\
&= \mathbf{E}\left\{\mathrm{ld}\left(\frac{p_{XY}(XY)}{p_X(X)p_Y(Y)}\right)\right\} \\
&= \mathbf{E}\left\{-\mathrm{ld}\left(\frac{p_X(X)p_Y(Y)}{p_{XY}(XY)}\right)\right\} \\
&= \mathbf{E}\left\{-\mathrm{ld}\left(\frac{p_Y(Y)p_X(X)}{p_{YX}(YX)}\right)\right\} = I(Y;X) = H(Y) - H(Y|X)
\end{aligned}
$$

X is as informative about Y as Y is informative about X
$\Rightarrow$ mutual information $I(X;Y)$

Entropy of two discrete random variables and mutual information

symmetry: $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = I(Y;X)$



chain rule: $H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$

Extension to continuous random variables

Differential entropy of a continuous random variable X with probability density function $p_X(x)$:

$$h(X) = \mathbf{E}\{-\mathrm{ld}(p_X(X))\} = -\int_{x\in\mathrm{supp}(p_X)} p_X(x)\,\mathrm{ld}(p_X(x))\,\mathrm{d}x$$

- Definition of the conditional differential entropies in analogy to the discrete case

Example:

- Let X denote a random variable being uniformly distributed in $[0, a)$ for $a \in \mathbb{R}^+$ with PDF

$$p_X(x) = \begin{cases} 1/a & \text{for } x \in [0, a) \\ 0 & \text{else.} \end{cases}$$

- Then we obtain $h(X) = \mathbf{E}\{-\mathrm{ld}(p_X(X))\} = \mathrm{ld}(a)$.
- Note that, differently from before where we always had $H(X) \in \mathbb{R}_0^+$, now $h(X)$ might take arbitrary (negative and non-negative) values depending on the value of a.

Problem

Let X be a real-valued zero-mean Gaussian random variable with $X \sim \mathcal{N}(0; \sigma^2)$. Show that

$$h(X) = \frac{1}{2}\mathrm{ld}(2\pi\mathrm{e}\sigma^2).$$

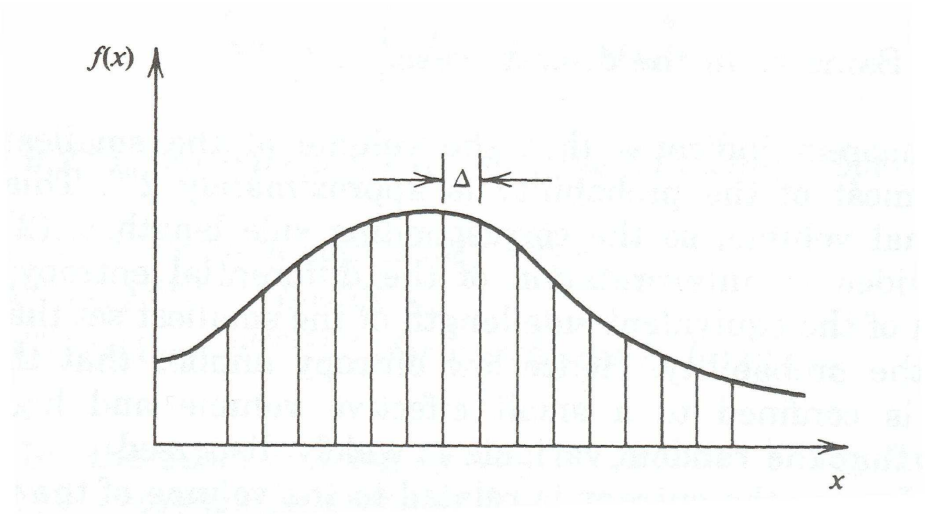
As a result:

- discrete random variable: entropy $H \in \mathbb{R}_0^+$
  continuous random variable: differential entropy $h \in \mathbb{R}$

  $\Rightarrow$ h might be negative

- How can we define the mutual information for the important case of continuous random variables?

To that end, consider the quantization of a continuous random variable X with PDF $f(x)$:



Quantization of a continuous random variable.

- Suppose we divide the range of X into bins of length Δ and the density is continuous within the bins.
- By the mean value theorem, there exists a value x_i within each bin such that

$$f(x_i)\Delta = \int_{i\Delta}^{(i+1)\Delta} f(x)\mathrm{d}x.$$

- Consider the quantized random variable X^Δ, which is defined by

$$X^\Delta = x_i \qquad \text{if } i\Delta \leq X \leq (i+1)\Delta.$$

Then, the probability that $X^\Delta = x_i$ is

$$p_i = \int_{i\Delta}^{(i+1)\Delta} f(x)\mathrm{d}x = f(x_i)\Delta.$$

- The entropy of the quantized version is

$$\begin{aligned} H\left(X^{\Delta}\right) &= -\sum_{i=-\infty}^{\infty} p_i \mathrm{ld}(p_i) \\ &= -\sum_{i=-\infty}^{\infty} f(x_i)\Delta \mathrm{ld}(f(x_i)\Delta) \\ &= -\sum_{i=-\infty}^{\infty} \Delta f(x_i)\mathrm{ld}(f(x_i)) - \sum_{i=-\infty}^{\infty} f(x_i)\Delta \mathrm{ld}(\Delta) \\ &= -\sum_{i=-\infty}^{\infty} \Delta f(x_i)\mathrm{ld}(f(x_i)) - \mathrm{ld}(\Delta). \end{aligned}$$

- We assume $f(x)\mathrm{ld}(f(x))$ to be Riemann integrable, so that $-\sum_{i=-\infty}^{\infty} \Delta f(x_i)\mathrm{ld}(f(x_i)) \overset{\Delta\to 0}{\longrightarrow} h(X)$.

Theorem 1: If the density $f(x)$ of the random variable X is Riemann integrable, then $H\left(X^{\Delta}\right) + \mathrm{ld}(\Delta) \to h(X)$ as $\Delta \to 0$. Thus, the entropy of an n-bit quantization of a continuous random variable X is approximately $h(X) + n$.

Examples:

- If X has a uniform distribution on $[0, 1]$ and we let $\Delta = 2^{-n}$, then $h(X) = 0$, $H\left(X^{\Delta}\right) = n$ and n bits suffice to describe X to n bit accuracy.

- If X has a uniform distribution on $[0, \frac{1}{8}]$, then the first 3 bits to the right of the decimal point must be 0. To describe X to n bit accuracy requires only $n - 3$ bits, which agrees with $h(X) = -3$.

Note that, in general, $h(X) + n$ is the number of bits *on the average* required to describe X to n bit accuracy.

Mutual information and relative entropy

How can we extend the notion of *mutual information* to continuous random variables?

We consider the mutual information between quantized versions X^Δ and Y^Δ of two random variables X and Y respectively. We obtain

$$I\left(X^\Delta;Y^\Delta\right) = H\left(X^\Delta\right) - H\left(X^\Delta|Y^\Delta\right) \approx h(X) - \mathrm{ld}(\Delta) - (h(X|Y) - \mathrm{ld}(\Delta)) = I(X;Y).$$

Thus, we define the mutual information between X and Y as the limit of $I\left(X^\Delta;Y^\Delta\right)$ with $\Delta \to 0$:

Definition: The *mutual information* between two random variables with joint density $f(x,y)$ is defined as

$$I(X;Y) = \int\int f(x,y)\mathrm{ld}\left(\frac{f(x,y)}{f(x)f(y)}\right)\mathrm{d}x\mathrm{d}y$$

and we have $I(X;Y) = h(X) - h(X|Y) = h(Y) - h(Y|X)$.

For later use, we will redefine the mutual information based on the so-called *Kullback-Leibler distance* between two density functions.

Mutual information and relative entropy

Definition: The *relative entropy* (or *Kullback-Leibler distance*) $D(f||g)$ between two densities f and g is defined by

$$D(f||g) = \int f(x) \log \left(\frac{f(x)}{g(x)} \right) \mathrm{d}x.$$

Note that $D(f||g)$ is finite only if the support set of f is contained in the support set of g (motivated by continuity, we set $0 \operatorname{ld}(0/0) = 0$).

Obviously, we can reformulate the mutual information by

$$I(X;Y) = D(f(x,y)||f(x)f(y)).$$

Problem

Show that

- $D(f||g)$ is indeed a distance, i.e. $D(f||g) \geq 0$, using Jensen's inequality and thus $I(X;Y) \geq 0$
- equality in $I(X;Y) \geq 0$ holds iff (i.e. if and only if) X and Y are independent
- $h(X|Y) \leq h(X)$ with equality iff X and Y are independent: conditioning cannot increase differential entropy (in analogy to corresponding entropy condition).

Problem

Show that

- $h(X+c) = h(X)$, i.e. translation does not change differential entropy
- $h(aX) = h(X) + \mathrm{ld}(|a|)$; as an example consider the random variables X uniformly distributed in $[0,1]$ and $Y = 2X$ being uniformly distributed in $[0,2]$; we have $h(X) = 0$ and $h(Y) = 1$ in accordance with our arguing in the context of entropies of quantized random variables.

For the transmission of information in AWGN channels, we have the following

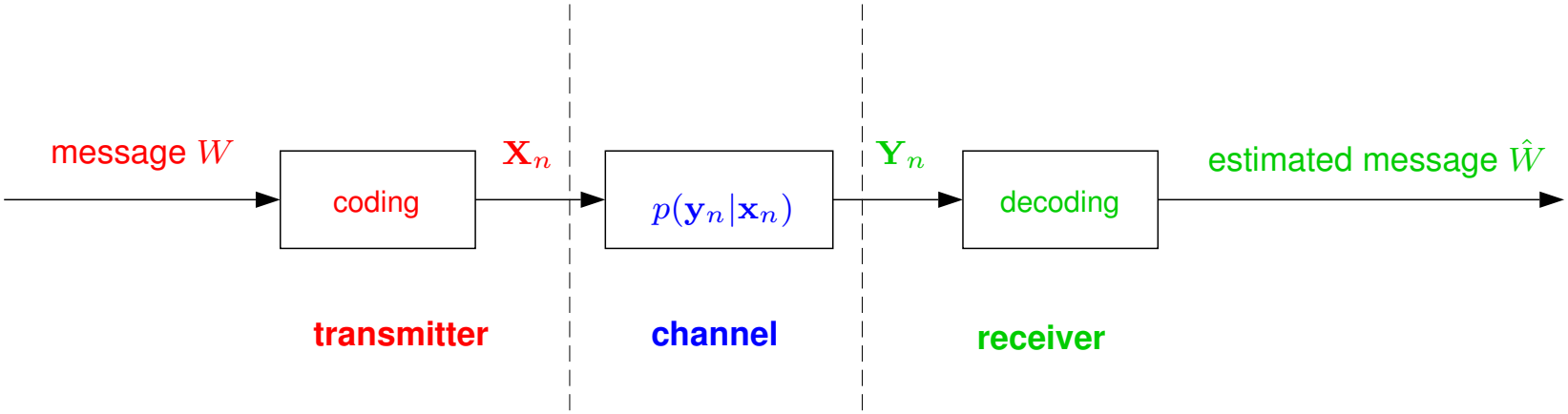
Theorem 2: Let the random variable $X \in \mathbb{R}$ have zero mean and covariance $\sigma^2 = \mathbf{E}\{X^2\}$. Then, $h(X) \leq \frac{1}{2}\mathrm{ld}\left(2\pi \mathrm{e}\sigma^2\right)$ with equality iff $X \sim \mathcal{N}\left(0;\sigma^2\right)$.

Problem

Consider two PDFs $g(x)$ with $\int_{\mathbb{R}} x^2 g(x)\mathrm{d}x = \sigma^2$ and $f(x) = \mathcal{N}\left(0;\sigma^2\right)$. Using $D(f||g) \geq 0$ and $\int f(x)\mathrm{ld}(f(x))\mathrm{d}x = \int g(x)\mathrm{ld}(f(x))\mathrm{d}x$, prove the theorem above.

How Much Information Can be Transmitted?

- What is the maximum amount of information that can be transmitted over the channel? (How a transmission can be designed and what *transmission* really means, will be discussed later on).
- Clearly, the mutual information depends on both the transition probability of the channel and the input distribution of X.
- Since we have no influence on the channel, we can only vary the input distribution $p_{\mathbf{X}_n}(\mathbf{x}_n)$ in order to maximize the mutual information.
- This approach is the basis for the derivation of the so-called **channel capacity**.
- Since the definition of the channel capacity depends on the transmission channel, we consider simple examples and derive the corresponding quantities.



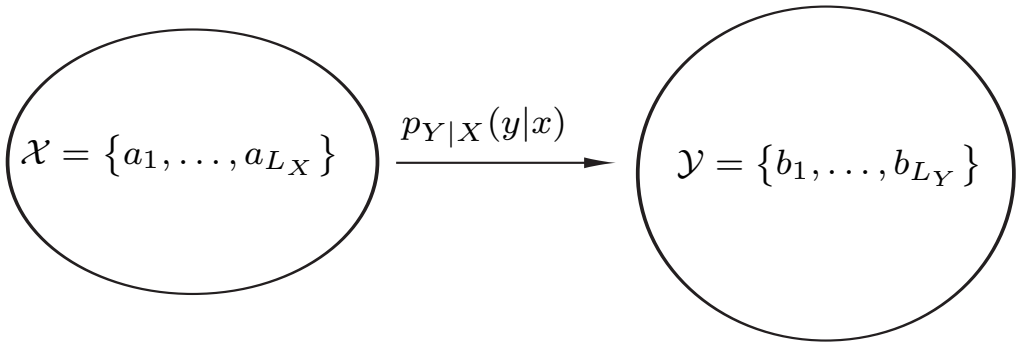
Consider different channels:

- discrete memoryless channel (DMC)
- the noisy typewriter
- the continuous Gaussian channel

Assumption: the system has no feedback channel, i.e.

$$p(x_n|x_1,\ldots,x_{n-1},y_1,\ldots,y_{n-1}) = p(x_n|x_1,\ldots,x_{n-1})$$

Discrete memoryless channel (DMC)



Consider a sequence of input random variables $X_1, X_2, X_3, \ldots$ drawn from a finite set of outcoumes $\mathcal{X}$ resulting in as sequence of output variables $Y_1, Y_2, Y_3, \ldots$ contained also in a finite set of outcoumes $\mathcal{Y}$. We assume the channel to be memoryless, i.e. the conditional PDF $p_{Y|X}(\cdot|x)$ over $\mathcal{Y}$ satisfies for each $x \in \mathcal{X}$ and $n = 1, 2, \ldots$

$$p(y_n|x_1, \ldots, x_n, y_1, \ldots, y_{n-1}) = p_{Y|X}(y_n|x_n)$$

This defines the so-called discrete memoryless channel (DMC)

Discrete memoryless channel (DMC)

Problem

Show that a DMC without feedback (assumed throughout) satisfies

$$p(y_1, \ldots, y_n | x_1, \ldots, x_n) = \prod_{i=1}^{n} p_{Y|X}(y_i|x_i)$$

Note that this is *not the definition of a DMC*, but only holds in the absence of feedback.

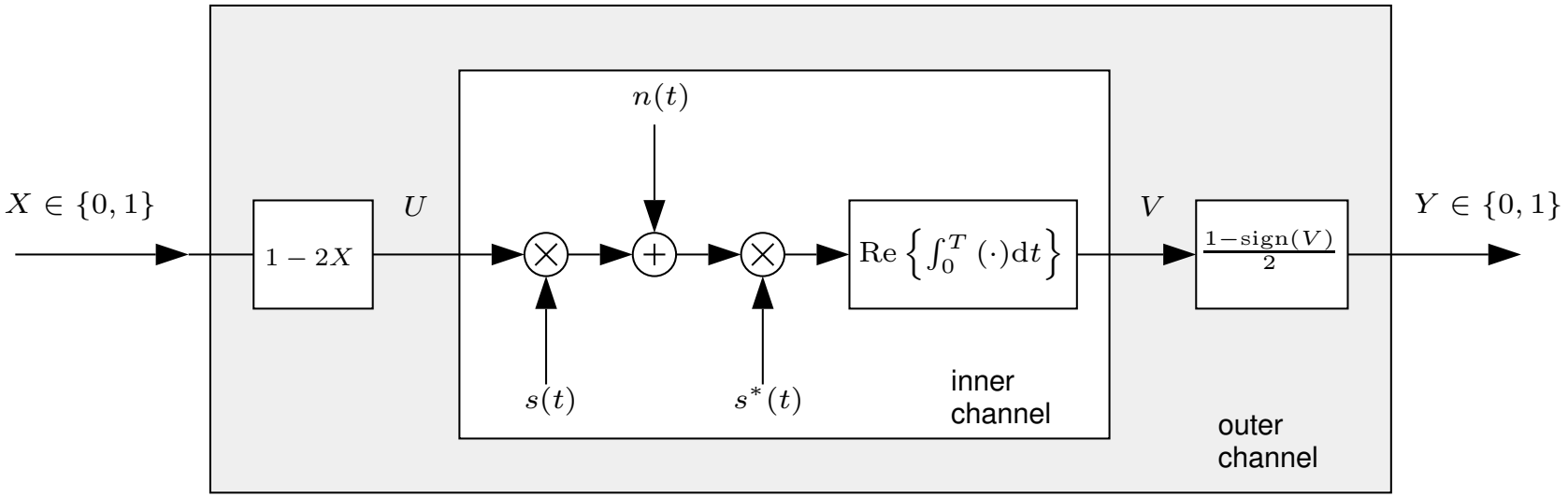
The channel capacity of a DMC is defined by

$$C_{\mathrm{DMC}} = \max_{p_X(x)} I(X;Y)$$

In order to calculate the capacity, we have to define the transition probability of the channel. A simple standard model for this is the so-called binary symmetric channel (BSC).

Binary symmetric channel (BSC)

To find the mapping from a standard waveform channel model to a DMC, consider a BPSK transmission system.



The *inner channel* carries out the mapping $U \to V$, the outer one the mapping $X \to Y$.

Binary symmetric channel (BSC)

Assumptions:

- information symbols $X \in \{0, 1\}$
- BPSK symbols $U = 1 - 2X \in \{1, -1\}$
- modulated waveform $s(t)$ with unit-energy $E_\mathrm{b} = \int_0^T |s(t)|^2 \, \mathrm{d}t = 1$
- thermal noise is modelled as zero-mean additive white Gaussian noise (AWGN) with $\mathbf{E}\{n(t+\tau)n^*(t)\} = N_0\delta(\tau)$ and N_0 denoting the noise power spectral density
- matched filtering at receiver to calculate a sufficient statistics for subsequent symbol detection ($(\cdot)^*$ and $\mathrm{Re}\{\cdot\}$ denote complex conjugation and the real part operator, resp.)
- with $\sigma_0 = \sqrt{N_0/2}$ the output signal conditioned on U is a Gaussian random variable

  $$V \sim \mathcal{N}\big(U; \sigma^2\big),$$

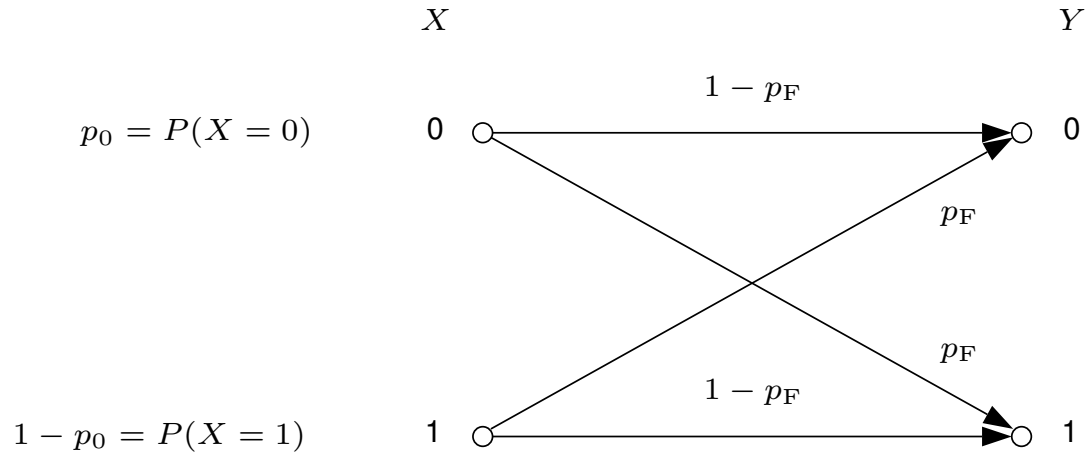
  which is fed to a hard-decision device $Y = (1 - \mathrm{sign}(V))/2$
- the bit-error probability $p_\mathrm{F} = P(Y \neq X | X = x)$ is independent of x and given by

  $$p_\mathrm{F} = Q\left(\frac{1}{\sigma_0}\right)$$

  with $Q(x) = (2\pi)^{-1/2} \int_x^\infty \mathrm{e}^{-t^2/2} \mathrm{d}t$.

Binary symmetric channel (BSC)

As a result, we can model the mapping $X \to Y$ by a so-called binary symmetric channel due to the obvious symmetry properties of that channel.



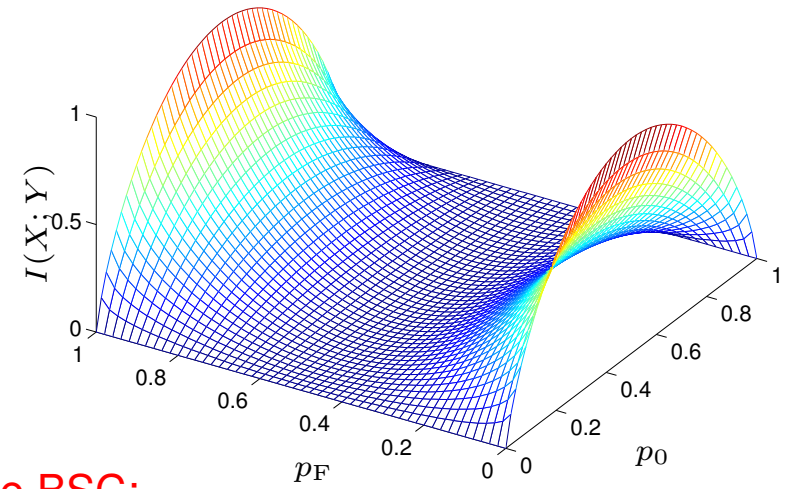
Problem

Calculate the mutual information of the BSC for $P(X = 0) = p_0$ and maximize it w.r.t. the input distribution to find its capacity. Using the symmetry condition $H_{\mathrm{b}}(p) = H_{\mathrm{b}}(1 - p)$ of the binary entropy function show that the following symmetry relations hold: $I(X;Y)|_{p_0,p_{\mathrm{F}}} = I(X;Y)|_{1-p_0,p_{\mathrm{F}}} = I(X;Y)|_{p_0,1-p_{\mathrm{F}}}$.

Binary symmetric channel (BSC)

The mutual information with $P(X = 0) = p_0$ is given by

$$I(X;Y) = H_\mathrm{b}(p_\mathrm{F}(1-2p_0)+p_0) - H_\mathrm{b}(p_\mathrm{F})$$



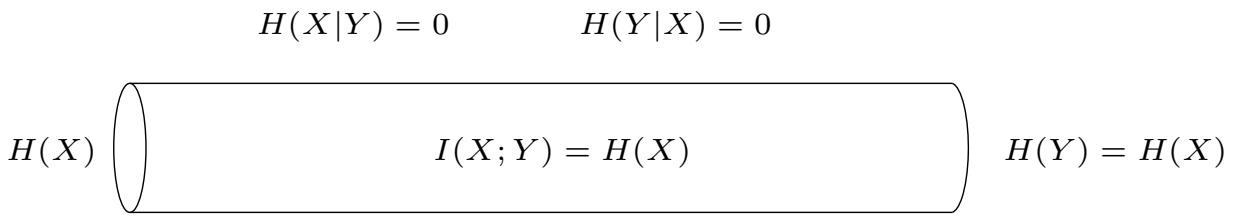
Channel capacity of the BSC:

$$C_\mathrm{BSC} = \max_{p_0} H_\mathrm{b}(p_\mathrm{F}(1-2p_0)+p_0) - H_\mathrm{b}(p_\mathrm{F}) = 1 - H_\mathrm{b}(p_\mathrm{F}).$$

Binary symmetric channel (BSC)

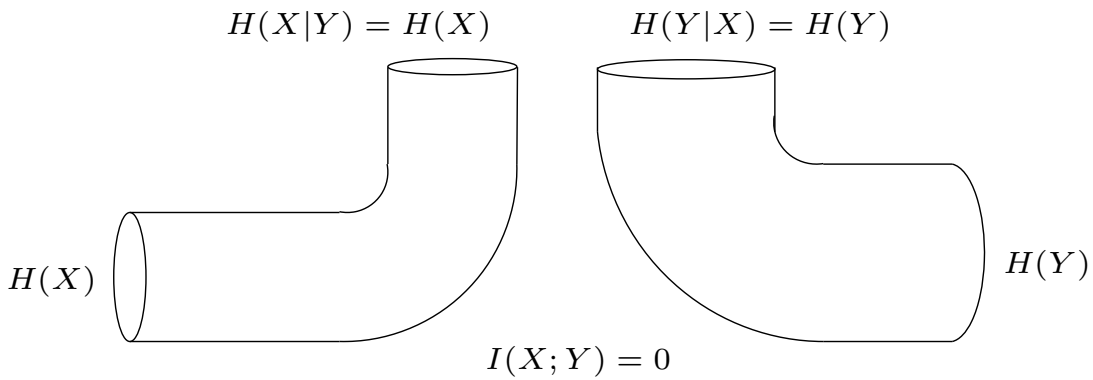
Perfect binary channel $p_{\mathrm{F}} = 0$:

mutual information $= I(X;Y) = H(X) - H(X|X) = H(X) - 0 = H(X) =$ self information

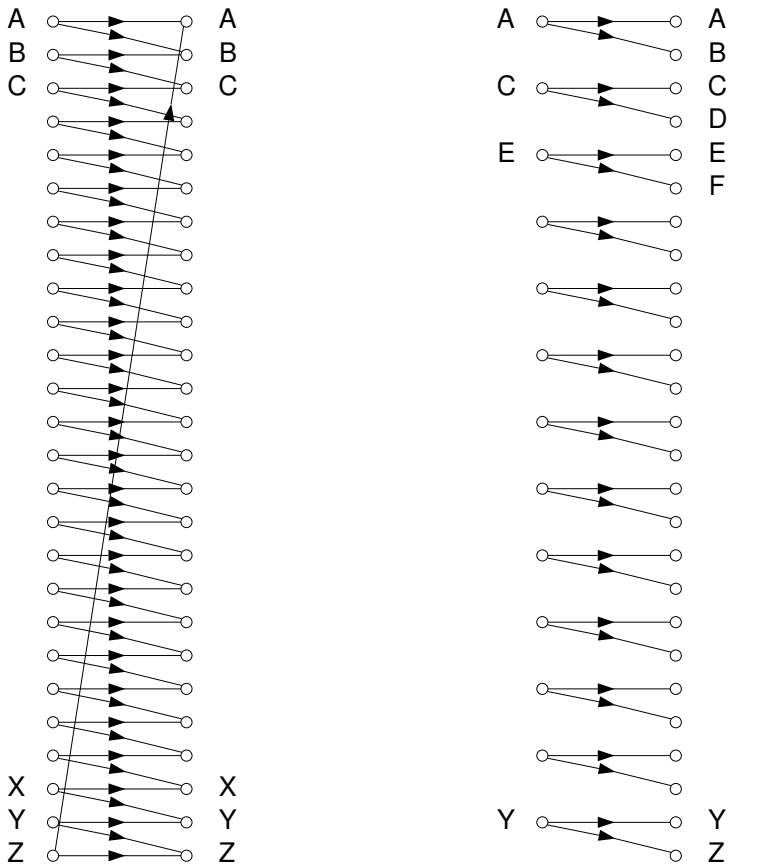


Useless channel $Y = Z$ with Z being independent of X:

mutual information $= I(X;Y) = H(X) - H(X|Y) = H(X) - H(X) = 0$



The noisy typewriter

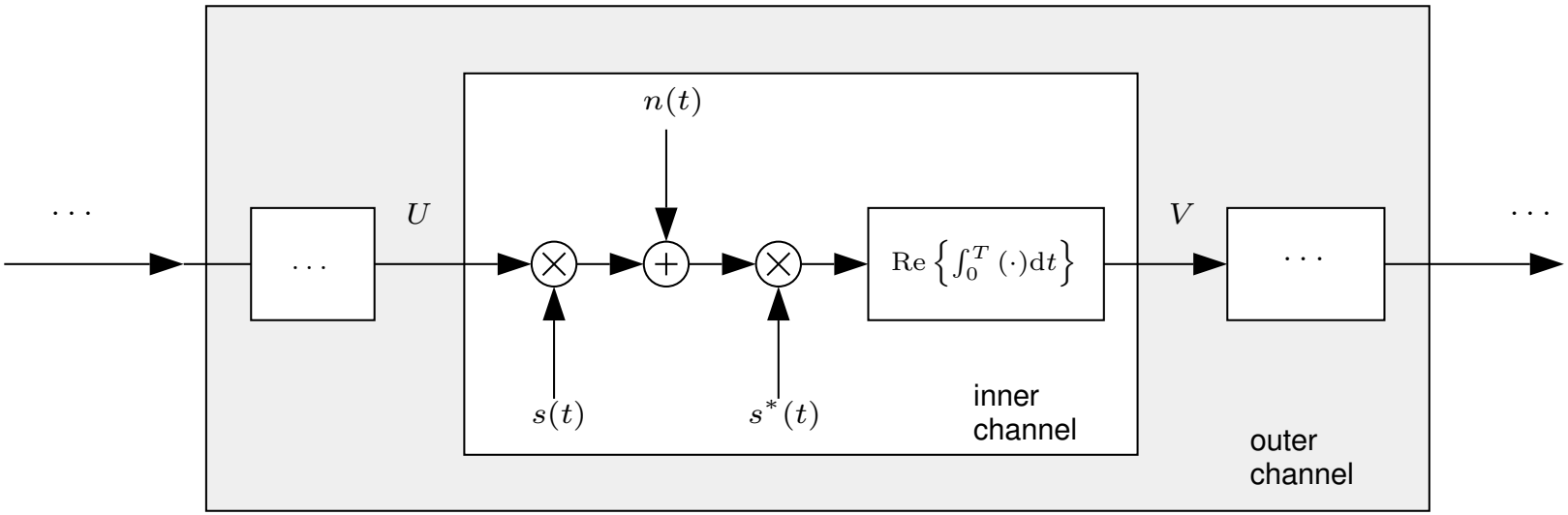


Problem

Calculate the channel capacity of the noisy typewriter for 26 input letters and the both noisy outputs being equally probable. Use intuitive arguments as well as the definition to show that the channel capacity $= \mathrm{ld}(13)\,\mathrm{bit/use}$.

The continuous Gaussian channel

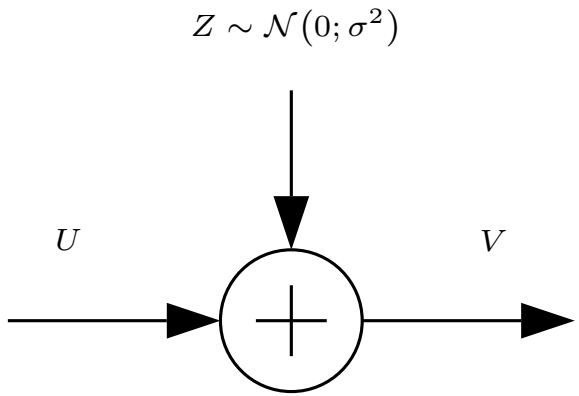
- Until now, we have only considered discrete random variables at the outer channel input. For example, the restriction to a binary signaling provides a maximum capacity of 1 bit/use e.g. in the case of a BSC with $p_{\mathrm{F}} = 0$.
- Thus, it is natural to ask whether this limitation is due to the channel properties, the signaling or both. By intuition, we would expect that we can increase the capacity by using a continuous channel input alphabet.
- To investigate such a situation, reconsider the transition from the waveform channel to the discrete channel. What happens if we let U be a continuous random variable (instead of drawing it from $\{-1, 1\}$) and calculate the capacity of the inner channel mapping U to V?



The continuous Gaussian channel

Equivalent channel model of the inner channel with

- a continuous random variable U as input signal
- an additive zero-mean Gaussian noise $Z \sim \mathcal{N}(0; \sigma^2)$
- a continuous random variable V as output signal



What is the capacity of this channel? Can it be higher than 1 bit/use?

The continuous Gaussian channel

Consider some special cases:

- for $\sigma^2 = 0$, we have $V = U$; since U is a continuous random variable, it should be possible to transmit an infinite amount of information per channel use
- even if $\sigma^2 > 0$, we can choose channel input signals which are spaced sufficiently apart so that in the limit, they are again perfectly distinguishable resulting in infinite capacity again
- obviously, both cases are not practically relevant; instead, we put a constraint on the possible input signals; the most common is to limit the average power P of U
- as a result, we only admit those input signal PDFs $p_U(u)$ satisfying $\mathbf{E}\left\{U^2\right\} \leq P$

The information capacity of the Gaussian channel with power constraint P is given by

$$C_{\mathrm{GC}} = \max_{p_U(u):\mathbf{E}\left\{U^2\right\}\leq P} I(U;V),$$

where the maximum is to be taken over all possible $p_U(x) : \mathbf{E}\left\{U^2\right\} \leq P$.

The continuous Gaussian channel

- We first calculate the mutual information of the Gaussian channel using the result that the differential entropy of a zero-mean Gaussian random variable $Z \sim \mathcal{N}(0;\sigma^2)$ is $h(Z) = \frac{1}{2}\mathrm{ld}(2\pi\mathrm{e}\sigma^2)$:

$$\begin{aligned} I(U;V) &= h(V) - h(V|U) \\ &= h(V) - h(U+Z|U) \\ &= h(V) - h(Z|U) \\ &= h(V) - h(Z) \\ &= h(V) - \frac{1}{2}\,\mathrm{ld}(2\pi\mathrm{e}\sigma^2). \end{aligned}$$

- Thus we look for the PDF $p_V(v)$ for which $h(V)$ is maximum for $\mathbf{E}\{U^2\} \leq P$.
- Since U and Z are mutually independent, this condition leads to a condition on V given by $\mathbf{E}\{V^2\} \leq P + \sigma^2$.
- Which PDF maximizes the differential entropy $h(V)$ for a given value of $P + \sigma^2$?

The continuous Gaussian channel

From Theorem 2, the differential entropy for the average power constraint $\mathbf{E}\{V^2\} \leq P + \sigma^2$ is upper bounded by

$$\frac{1}{2}\mathrm{ld}\big(2\pi\mathrm{e}\big(P+\sigma^2\big)\big)$$

and reached iff $V \sim \mathcal{N}\big(0; P+\sigma^2\big)$. This can only be achieved if we use in turn a Gaussian input signal $U \sim \mathcal{N}(0; P)$. As a result, we obtain

Theorem 3: The capacity of a Gaussian channel with power constraint P and noise variance σ^2 in [bit/use] is

$$C_{\mathrm{GC}} = \max_{p_U(u):\mathbf{E}\{U^2\}\leq P} I(U;V) = \frac{1}{2}\,\mathrm{ld}\left(1+\frac{P}{\sigma^2}\right).$$

How can we identify parameters like e.g. bandwidth and noise power spectral density in a waveform channel in this important result?

The continuous Gaussian channel

- Following the model on p. 381 of Proakis, we consider a waveform channel for $t \in [0, T)$ with the received signal given by

$$Y(t) = X(t) + N(t),$$

  where $X(t)$ is a transmitted white Gaussian random process whose power spectrum is zero for $|f| > W$ and $N(t)$ is a zero-mean white Gaussian noise with noise power spectral density of $N_0/2$.

- We take $N = 2WT$ samples of $Y(t)$ taken at the Nyquist rate of $2W = 1/T_\mathrm{s}$ samples/s so that the received signal is given by

$$Y_i = Y(i/2W) = X(i/2W) + N(i/2W) = X_i + N_i \qquad \text{for } i = 1, \ldots, N.$$

- Since the noise is white, the noise samples N_i are statistically independent and identically distributed random variables with

$$p(n_i) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{n_i^2}{N_0}\right).$$

- Obviously, the DMC condition $p(y_1, \ldots, y_n | x_1, \ldots, x_n) = \prod_{i=1}^{n} p_{Y|X}(y_i|x_i)$ is met.

The continuous Gaussian channel

- Furthermore, the average power P_{av} of $X(t)$ and the average power P of the samples in the time-discrete channel model are related by

$$P = \mathbf{E}\left\{\int_0^{T_{\mathrm{s}}} X^2(t)\mathrm{d}t\right\} = \mathbf{E}\left\{[X(i/2W)]^2\right\} = P_{\mathrm{av}}T_{\mathrm{s}} = \frac{P_{\mathrm{av}}}{2W}.$$

- Inserting $\sigma^2 = N_0/2$ and $P = \frac{P_{\mathrm{av}}}{2W}$ into the channel capacity formula for AWGN, we have for each channel use

$$C_{\mathrm{GC}} = \frac{1}{2}\,\mathrm{ld}\left(1 + \frac{P}{\sigma^2}\right) = \frac{1}{2}\,\mathrm{ld}\left(1 + \frac{P_{\mathrm{av}}}{WN_0}\right).$$

- It is intuitive that the mutual information for N channel uses is maximized if the input random variables are chosen mutually independent. Setting $T = NT_{\mathrm{s}} = \frac{N}{2W}$, we obtain the channel capacity for N uses of the channel:

$$C_{\mathrm{GC}} = \frac{N}{2}\,\mathrm{ld}\left(1 + \frac{P_{\mathrm{av}}}{WN_0}\right) = WT\,\mathrm{ld}\left(1 + \frac{P_{\mathrm{av}}}{WN_0}\right).$$

The continuous Gaussian channel

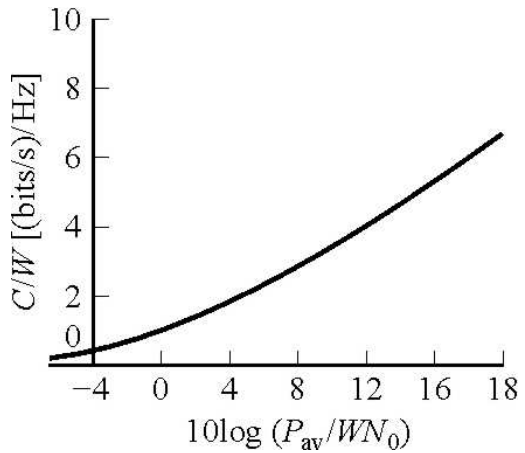
- The capacity in [bit/s] is obtained by normalizing the above result w.r.t. time (cf. Proakis (7.1-31)):

$$C_{\mathrm{GC}}/[\mathrm{bit/s}] = W \mathrm{ld}\left(1 + \frac{P_{\mathrm{av}}}{W N_0}\right).$$

- Often, the capacity is further normalized to the required bandwidth W, thus given in [bit/s/Hz]:

$$C_{\mathrm{GC}}/[\mathrm{bit/s/Hz}] = \mathrm{ld}\left(1 + \frac{P_{\mathrm{av}}}{W N_0}\right).$$

Mind the differences in the meaning of the variables P, σ^2 and P_{av}, W and N_0 in $C_{\mathrm{GC}}/[\mathrm{bit/use}]$ and $C_{\mathrm{GC}}/[\mathrm{bit/s/Hz}]$, respectively.



Normalized channel capacity as a function of SNR for a band-limited AWGN channel.

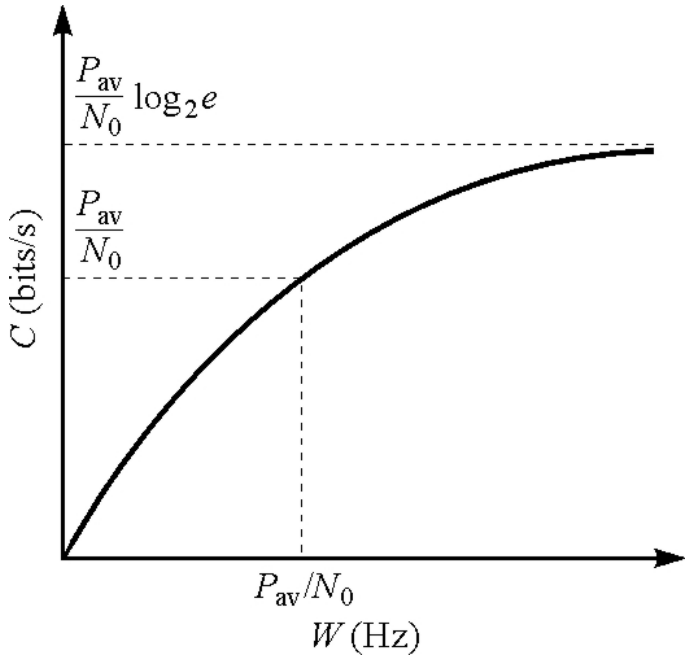
The continuous Gaussian channel

- In the high SNR regime, for a fixed W, the capacity can be doubled by doubling P_{av}.

- What happens if we fix instead P_{av} and vary W in $C_{\mathrm{GC}}/[\mathrm{bit/s}] = W\mathrm{ld}\left(1+\frac{P_{\mathrm{av}}}{WN_0}\right)$ (e.g. in power limited systems)? We obtain

$$\begin{aligned}
C_\infty &= \lim_{W\to\infty} C_{\mathrm{GC}} \\
&= \lim_{W\to\infty} W\mathrm{ld}\left(1+\frac{P_{\mathrm{av}}}{WN_0}\right) \\
&= \lim_{\omega\to 0} \frac{\mathrm{ld}\left(1+\frac{P_{\mathrm{av}}\omega}{N_0}\right)}{\omega} \\
&= \lim_{\omega\to 0} \frac{\ln\left(1+\frac{P_{\mathrm{av}}\omega}{N_0}\right)\mathrm{ld(e)}}{\omega} \\
&= \lim_{\omega\to 0} \frac{P_{\mathrm{av}}}{N_0}\frac{\mathrm{ld(e)}}{1+\frac{P_{\mathrm{av}}\omega}{N_0}} \\
&= \frac{P_{\mathrm{av}}}{N_0}\mathrm{ld(e)} = \frac{P_{\mathrm{av}}}{N_0\ln(2)} \ \ \mathrm{bit/s}.
\end{aligned}$$

The continuous Gaussian channel



Channel capacity as a function of bandwidth W with a fixed transmitted average power P_{av}.

$$C_{\text{GC}}/[\text{bit/s}] = W\text{ld}\left(1 + \frac{P_{\text{av}}}{WN_0}\right).$$

The continuous Gaussian channel

- Let us consider the capacity in [bit/s/Hz] as a function of the SNR per bit. If C is the rate in [bit/s], we obviously have the identity $P_{\mathrm{av}} = C\varepsilon_{\mathrm{b}}$, where ε_{b} is the energy per bit.

- As as result, we have

$$\frac{C}{W} = \mathrm{ld}\left(1 + \frac{C}{W}\frac{\varepsilon_{\mathrm{b}}}{N_0}\right) \qquad \text{or, resp.,} \qquad \frac{\varepsilon_{\mathrm{b}}}{N_0} = \frac{2^{C/W} - 1}{C/W}.$$

- Consider three cases:

  1. $C/W = 1$: We obtain directly $\varepsilon_{\mathrm{b}}/N_0 = 1$.
  2. $C/W \to \infty$: For large $C/W \gg 1$, we can write

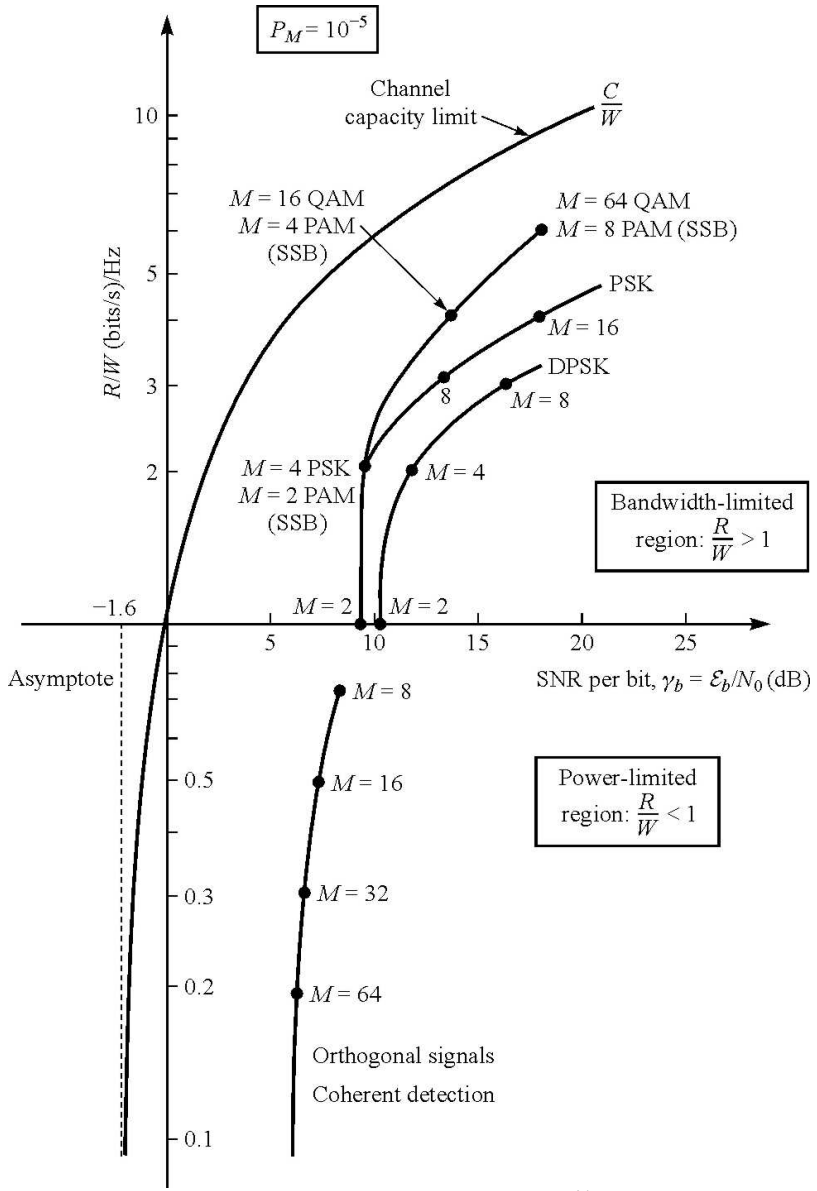
$$\frac{\varepsilon_{\mathrm{b}}}{N_0} \approx \frac{2^{C/W}}{C/W} = \exp\left(\frac{C}{W}\ln(2) \; - \ln\left(\frac{C}{W}\right)\right).$$

     Thus, $\varepsilon_{\mathrm{b}}/N_0$ increases exponentially as $C/W \to \infty$.

  3. $C/W \to 0$:

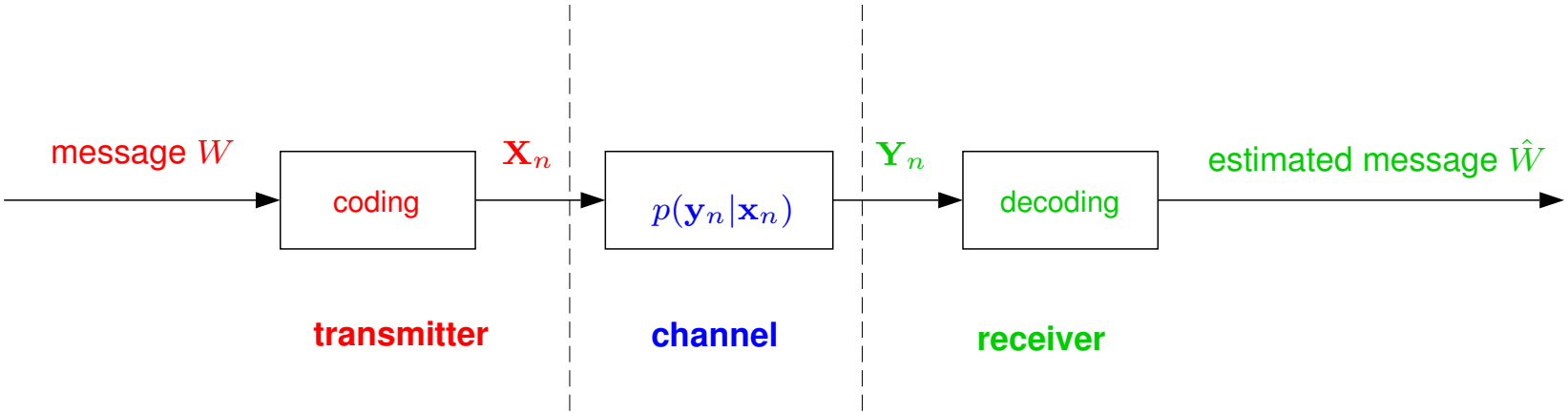
$$\frac{\varepsilon_{\mathrm{b}}}{N_0} = \lim_{C/W \to 0} \frac{2^{C/W} - 1}{C/W} = \ln(2) = -1.6 \text{ dB}.$$

The continuous Gaussian channel



Comparison of several modulation methods at 10^{-5} symbol error probability.

The channel coding theorem (Shannon, 1948)



For $W \in \{1, 2, \ldots, M\}$ let $R = \frac{\mathrm{ld}(M)}{n}$ bit/use define the rate of a (M, n) code.

Shannon: Information can be transmitted reliably over the channel if and only if $R < C$.

Example: the DMC

For a sequence of *independent identically distributed (i.i.d.)* random variables, their *mean* $\bar{X}$ converges in probability to the expectation $\mathbf{E}\{X\}$, i.e.

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \;\; \rightarrow \mathbf{E}\{X\} \text{ in prob.}$$

Consider now i.i.d. random variables $Y_i = -\mathrm{ld}(p_X(X_i)) \sim p_Y(y)$:

$$\begin{aligned}\bar{Y} &= \frac{1}{n}\sum_{i=1}^{n} Y_i = -\frac{1}{n}\sum_{i=1}^{n} \mathrm{ld}(p_X(X_i)) \\ &= -\frac{1}{n}\mathrm{ld}\big(p_{X_1,\ldots,X_n}(X_1,\ldots,X_n)\big) \;\; \rightarrow \mathbf{E}\{Y\} = \mathbf{E}\{-\mathrm{ld}(p_X(X))\} = H(X) \text{ in prob.}\end{aligned}$$

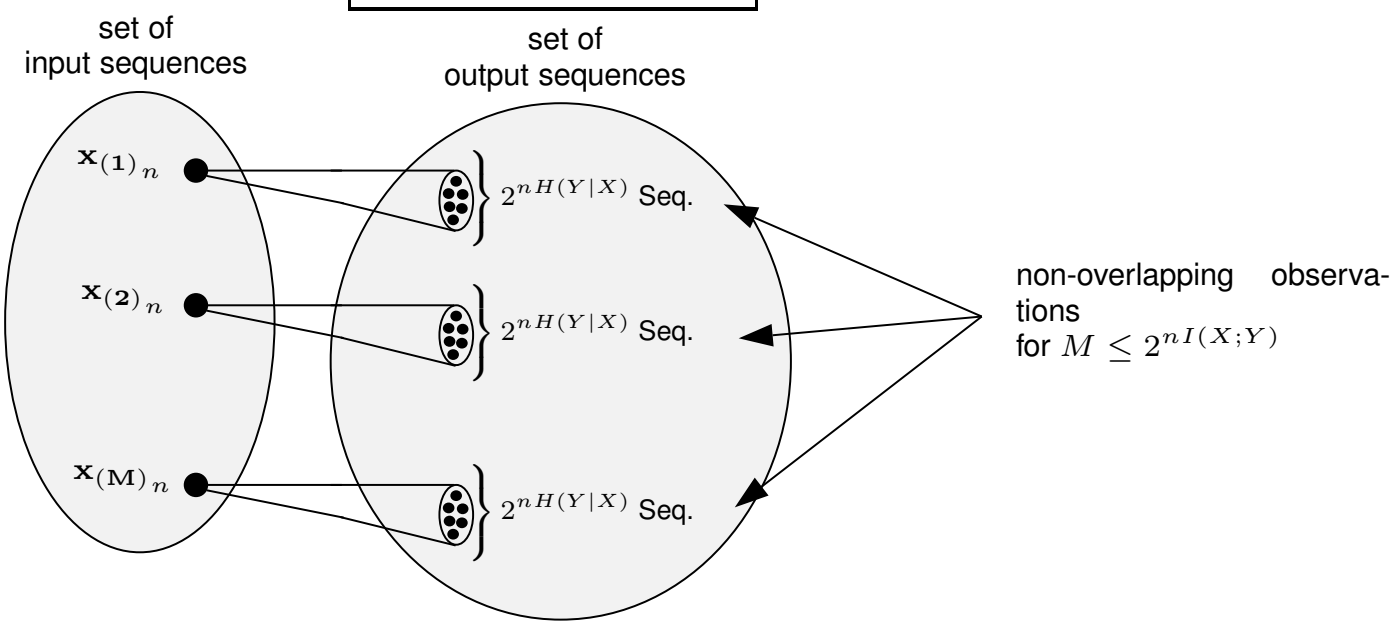
Thus, for sufficiently large values n, we only have so-called typical sequences with

$$p(x_1,\ldots,x_n) \approx 2^{-nH(X)}$$

For sufficiently large values n

- a total of $2^{nH(X)}$ typical sequences exists
- all (typical) sequences are equally probable
- thus, we have a situation like for the noisy typewriter.

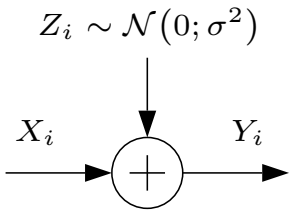
Example: the DMC



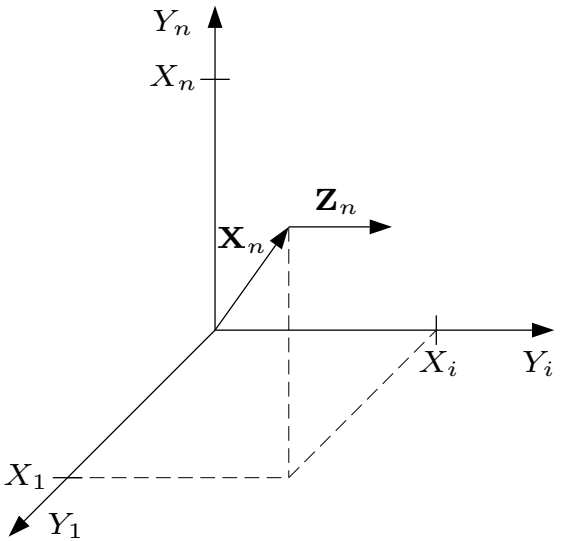
- approach as in the case of the noisy typewriter: for each (typical) sequence, there are $2^{nH(Y|X)}$ possible equally probable output sequences
- partition the total $2^{nH(Y)}$ output sequences into $M = 2^{nH(Y)}/2^{nH(Y|X)} = 2^{nI(X;Y)}$ groups, where each group can be associated uniquely with a certain input sequence
- thus, the maximum rate is given by $R = \mathrm{ld}(M)/n = C_{\mathrm{DMC}}$, above which the overlapping observations associated with different input sequences let the probability of error increase.

Information transmission over a continuous Gaussian channel

- Code word $\mathbf{X}_n = X_1, X_2, \ldots, X_n$.
- Sequence of independent disturbances $\mathbf{Z}_n = Z_1, Z_2, \ldots, Z_n$.



- Consider received sequence $\mathbf{Y}_n = Y_1, \ldots, Y_n$ in n-dimensional vector space.

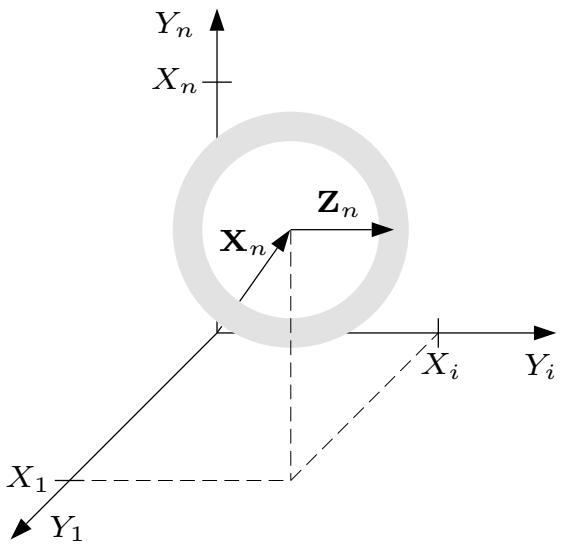


Information transmission over a continuous Gaussian channel

- For sufficiently large n, we have

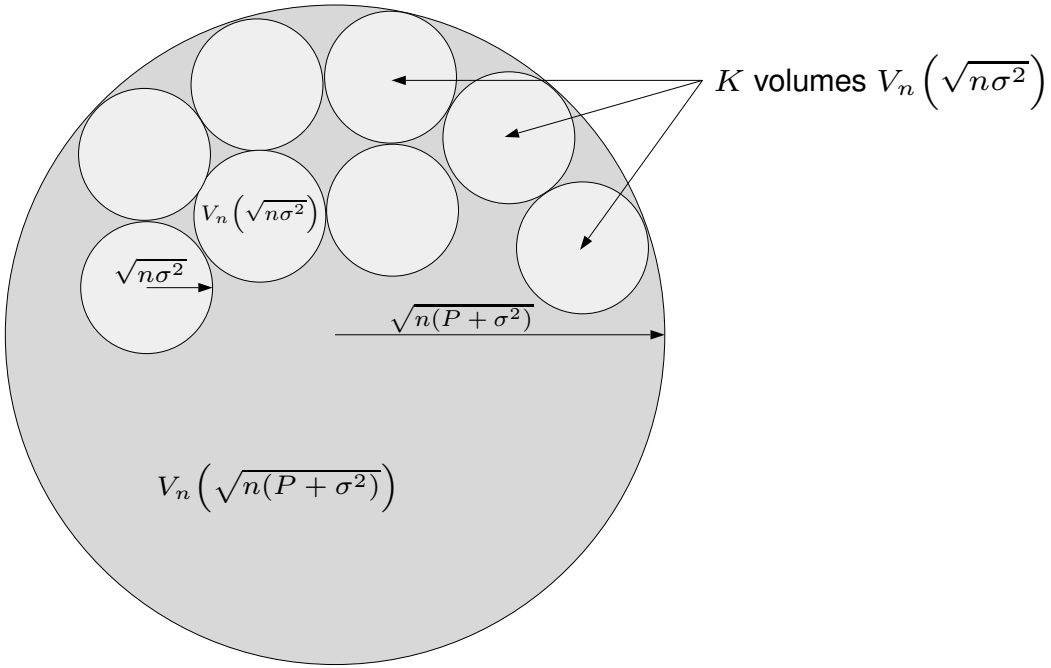
$$\|\mathbf{Z}_n\| = \sqrt{\sum_{i=1}^{n} Z_i^2} \approx \sqrt{n\mathbf{E}\{Z^2\}} = \sqrt{n\sigma^2}$$

and thus, the received sequences lie on an n-dimensional sphere surface with centre $\mathbf{X}_n$ and radius $r = \sqrt{n\sigma^2}$.



- Volume of a ball is given by $V_n(r) = A_n r^n$ with $A_n = \pi^{n/2}/\Gamma(1 + n/2)$.

Information transmission over a continuous Gaussian channel

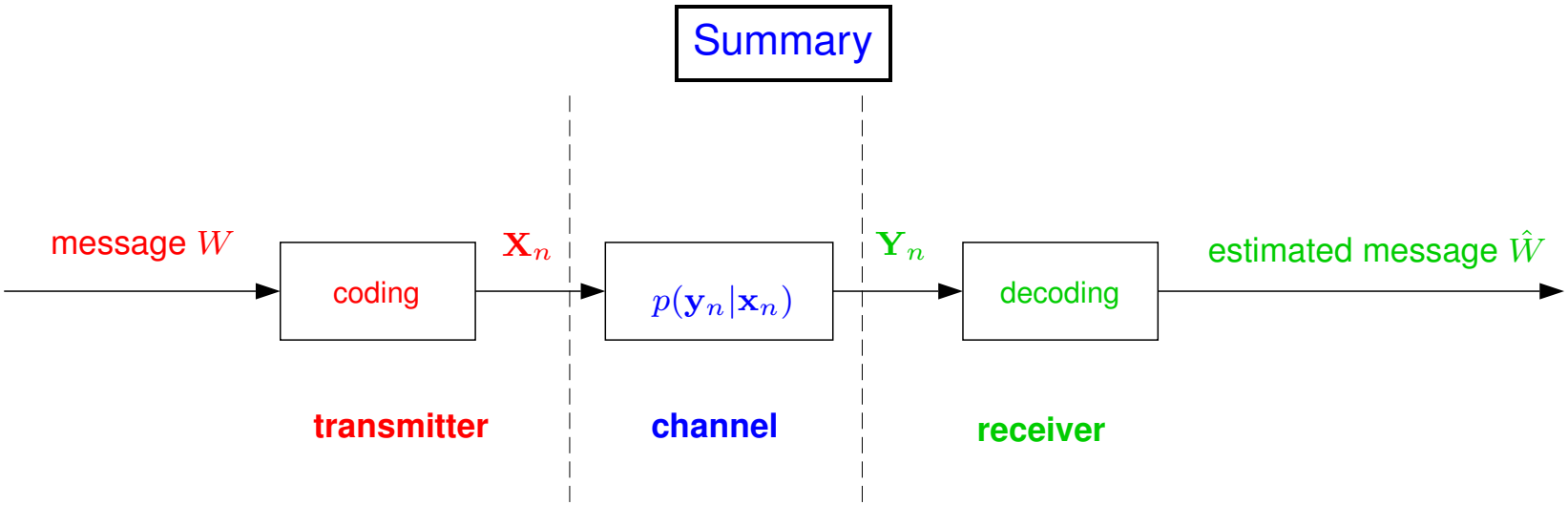


- Number of K non-overlapping volumes (in analogy to noise typewriter):

$$K \leq M = \frac{V_n\left(\sqrt{n(P+\sigma^2)}\right)}{V_n\left(\sqrt{n\sigma^2}\right)} = \frac{A_n\left(n\left(P+\sigma^2\right)\right)^{n/2}}{A_n(n\sigma^2)^{n/2}} = 2^{\frac{n}{2}\mathrm{ld}\left(1+\frac{P}{\sigma^2}\right)}.$$

- Maximum rate $= \frac{\mathrm{ld}(\max K)}{n} = \frac{1}{2}\mathrm{ld}\left(1+\frac{P}{\sigma^2}\right) = C_{\mathrm{GC}}$ bit/use.

How can information be transmitted efficiently over a channel?



information $\Rightarrow$ (1) *What* is information (of the receiver by the transmitter)?
Information is reduction of uncertainty (difference of entropies).

efficient $\Rightarrow$ (2) *How much* information can be transmitter over the channel?
The maximum information rate is given by the channel capacity C.

transmission $\Rightarrow$ (3) *How* can information be transmitted?
With a rate R being less than C.

Table of Contents

Channel Coding

- While the channel capacity theorem prescribes the maximum possible rate which can, in principal, be supported with zero error probability, it does not give a straightforward design criterion to find suitable codes.
- The usual way of encoding a stream of information bits to be transmitted over a noisy or fading channel is either a block code or a convolutional code or combinations thereof.
- Subsequently, we will consider
  - **block codes**
  - **convolutional codes**
- Coded modulation will not be considered.

Block Codes

- A block code consists of a set of fixed-length vectors called code words of length n.
- The elements of a code word are selected from an alphabet of q symbols or elements.
- When there are only $q = 2$ elements, namely 0 and 1, the code is called binary and the elements are so-called bits.
- There are 2^n possible code words in a binary block code of length n. From these 2^n code words, we may select $M = 2^k$ code words with $k < n$ to form a code. Thus, a block of k bits is mapped into a code word of length n and we refer to the resulting block code as an (n, k) code with code rate $R_\mathrm{c} = k/n$.
- Except for R_c, the weight of a code word, i.e. the number of nonzero elements it contains, is an important parameter. The set of *all* weights in a code constitutes the weight distribution of the code.
- When all code words have equal weight, the code is called a fixed-weight code or a constant-weight code.
- The encoding and decoding functions involve addition and multiplication operations defined on the q elements. The set of elements with the operations defined on them forms a field F.

Addition

1. The set F is closed under addition, i.e., if $a, b \in F$, then $a + b \in F$.
2. Addition is associative, i.e., if a, b and $c \in F$, then $a + (b + c) = (a + b) + c$.
3. Addition is commutative, i.e., $a + b = b + a$.
4. The set contains an element called *zero* that satisfies the condition $a + 0 = a$.
5. Every element in the set has its own negative element. Hence, if b is an element, its negative is denoted by $-b$. The subtraction of two elements, such as $a - b$, is defined ab $a + (-b)$.

Multiplication

1. The set F is closed under multiplication, i.e., if $a, b \in F$, then $ab \in F$.
2. Multiplication is associative, i.e., $a(bc) = (ab)c$.
3. Multiplication is commutative, i.e., $ab = ba$.
4. Multiplication is distributive over addition, i.e., $(a + b)c = ac + bc$.
5. The set F contains an element called the *identity* that satisfies the condition $a(1) = a$ for any element $a \in F$.
6. Every element of F, except zero, has an inverse. Hence, if $b \in F$ $(b \neq 0)$, then its inverse is defined by b^{-1} and $bb^{-1} = 1$. The division of two elements, such as $a \div b$, is defined ab ab^{-1}.

Examples:

- The field of real numbers (infinite number of elements).
- The field of complex numbers (infinite number of elements).

Further properties:

- Codes are constructed from fields with a finite number of elements. A finite field with q elements is generally called a Galois field and denoted by $\mathrm{GF}(q)$.
- Every field must have a zero element and a one element. Hence, the simplest field is $\mathrm{GF}(2)$.
- When q is a prime, we can construct the field $\mathrm{GF}(q)$ consisting of the elements $\{0, 1, 2, \ldots, q-1\}$. The major reason for requiring q to be a prime is the required existence of an inverse for every element except zero.
- The addition and multiplication operations on the elements of $\mathrm{GF}(q)$ are defined modulo q. Note that q is assumed to be a prime number. For example, if we had $q = 6$, we could not find the inverse of the element 2.
- Except for the case where q is a prime, a field can also be constructed if $q = p^m$ where p is a prime and m is any positive integer. In this case, we extend the field $\mathrm{GF}(p)$ to the field $\mathrm{GF}(p^m)$ being called extension field of $\mathrm{GF}(p)$. Multiplication and addition of the elements in the extension field are based on modulo-p arithmetic.

+	0	1
0	0	1
1	1	0

·	0	1
0	0	0
1	0	1

Addition and multiplication tables for $\mathrm{GF}(2)$.

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

·	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Addition and multiplication tables for $\mathrm{GF}(5)$.

Suppose that we have two code words $\mathbf{C}_i$ and $\mathbf{C}_j$. What is a good distance measure to discriminate these words?

Example (ML-Decoding): Consider a BSC with input X and output Y and

$$P(y|x) = \begin{cases} 1-p_{\mathrm{F}} & \text{if } y = x \\ p_{\mathrm{F}} & \text{if } y \neq x. \end{cases}$$

with cross-over probablity p_{F}. When a BSC is used without feedback to transmit a block of n binary digits, then

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}) &= p_{\mathrm{F}}^{d(\mathbf{x},\mathbf{y})}(1-p_{\mathrm{F}})^{n-d(\mathbf{x},\mathbf{y})} \\ &= (1-p_{\mathrm{F}})^n \left(\frac{p_{\mathrm{F}}}{1-p_{\mathrm{F}}}\right)^{d(\mathbf{x},\mathbf{y})}, \end{aligned}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ is the transmitted block, $\mathbf{y} = [y_1, y_2, \ldots, y_n]$ is the received block and $d(\mathbf{x}, \mathbf{y})$ is the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$, i.e. the number of positions in which the vectors $\mathbf{x}$ and $\mathbf{y}$ differ.

Since

$$0 < p_{\mathrm{F}} < \frac{1}{2} \quad \Leftrightarrow \quad 0 < \frac{p_{\mathrm{F}}}{1-p_{\mathrm{F}}} < 1,$$

it follows that choosing i to maximize $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}_i)$ is equivalent to choosing i to minimize $d(\mathbf{x}_i, \mathbf{y})$.

- Suppose that $\mathbf{C}_i$ and $\mathbf{C}_j$ are any two code words in an (n, k) block code. A measure of the difference between the code words is the Hamming distance denoted as d_{ij}.
- Clearly, we have $0 \leq d_{ij} \leq n$.
- The smallest value of the set $\{d_{ij}\}$ with $i \neq j$ is called the minimum distance $d_{\min}$.
- Linear codes:
  - Suppose that $\mathbf{C}_i$ and $\mathbf{C}_j$ are any two code words in an (n, k) block code and let α_1 and α_2 be any two elements selected from the alphabet.
  - Then, the code is said to be linear iff $\alpha_1 \mathbf{C}_i + \alpha_2 \mathbf{C}_j$ is also a code word. This implies (by choosing $\alpha_1 = \alpha_2 = 0$) that a linear code contains the all-zero code word. Consequently, a constant-weight code is non-linear.
  - Let $\mathbf{C}_i$ with $i = 1, \ldots, M$ denote the M code words of a linear code with $\mathbf{C}_1 = [00 \cdots 0]$ denoting the all-zero code word. Furthermore, w_r is the weight of the r-th code word which is obviously the Hamming distance between the code words $\mathbf{C}_r$ and $\mathbf{C}_1$, i.e., $d_{1r} = w_r$.
  - In general, the distance d_{ij} between code words $\mathbf{C}_i$ and $\mathbf{C}_j$ is the weight of the difference between $\mathbf{C}_i$ and $\mathbf{C}_j$.
  - Since the code is linear, the difference between $\mathbf{C}_i$ and $\mathbf{C}_j$ is also a code word with weight included in the set $\{w_r\}$.
  - Show that $d_{\min} = \min_{r, r \neq 1} w_r$.

The generator and parity check matrices

Linear codes:

- Let $\mathbf{X}_m = [x_{m1}, x_{m2}, \ldots, x_{mk}]$ denote the row vector of k information bits entering the encoder. The output is the code word defined by the n-dimensional row vector

$$\mathbf{C}_m = [c_{m1}, c_{m2}, \ldots, c_{mn}] = \mathbf{X}_m \mathbf{G}$$

with the generator matrix of the code defined by

$$\mathbf{G} = \begin{bmatrix} \leftarrow & \mathbf{g}_1 & \rightarrow \\ \leftarrow & \mathbf{g}_2 & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{g}_k & \rightarrow \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}.$$

- The code words can be written as linear combinations of the row vectors $\{\mathbf{g}_i\}$ of $\mathbf{G}$:

$$\mathbf{C}_m = x_{m1}\mathbf{g}_1 + x_{m2}\mathbf{g}_2 + \ldots + x_{mk}\mathbf{g}_k.$$

- Since the linear (n, k) code with 2^k code words is a subspace of dimension k, the set of vectors $\{\mathbf{g}_1, \ldots, \mathbf{g}_k\}$ must be linearly independent to form a basis for the (n, k) code. Thus, $\mathrm{rank}(\mathbf{G}) = k$.
- The set of basis vectors is not unique.

- Any generator matrix of an (n, k) code can be reduced by Gaussian elimination, i.e. row operations and column permutations to the systematic form:

$$\mathbf{G} = [\mathbf{I}_k | \mathbf{P}] = \left[\begin{array}{ccccc|cccc} 1 & 0 & 0 & \dots & 0 & p_{11} & p_{12} & \dots & p_{1\,n-k} \\ 0 & 1 & 0 & \dots & 0 & p_{21} & p_{22} & \dots & p_{2\,n-k} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 & p_{k1} & p_{k2} & \dots & p_{k\,n-k} \end{array}\right].$$

  Here, $\mathbf{I}_k$ and $\mathbf{P}$ denote a $(k \times k)$-dimensional identity matrix and a $(k \times n-k)$-dimensional matrix containing the $n-k$ redundant bits called parity check bits, respectively.

- Note that a generator matrix of the systematic form generates a linear block code in which the first k bits of each code word are identical to the information bits to be transmitted. The remaining $n-k$ bits of each code word are linear combinations of the k information bits.

- The resulting (n, k) code is called a systematic code. A code being not a systematic one which, however, can be obtained from a systematic one by elementary row operations and column permutations, is said to be equivalent to the systematic code. Thus, every linear (n, k) code is equivalent to a systematic (n, k) code.

Example: Consider a $(7, 4)$ code with generator matrix

$$\mathbf{G} = [\mathbf{I}_4 | \mathbf{P}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

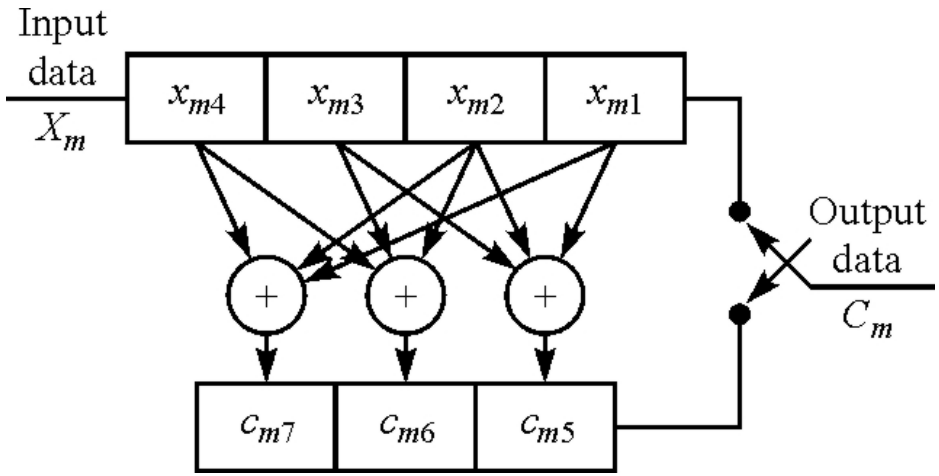
A code word may be expressed as

$$\mathbf{C}_m = [x_{m1}\, x_{m2}\, x_{m3}\, x_{m4}\, c_{m5}\, c_{m6}\, c_{m7}],$$

where the $\{x_{mj}\}$ represent the four information bits (arising unchanged in the code word) and the $\{c_{mj}\}$ represent the three parity check bits given by

$$\begin{aligned} c_{m5} &= x_{m1} + x_{m2} + x_{m3} \\ c_{m6} &= x_{m2} + x_{m3} + x_{m4} \\ c_{m7} &= x_{m1} + x_{m2} + x_{m4}. \end{aligned}$$

Implementation of a linear systematic binary block encoder



A linear shift register for generating a $(7, 4)$ binary code.

- First, we calculate the bits in the lower delay line, afterwards release the connections between the upper and the lower delay lines keeping the lower values.
- Then, the bits in the top delay line are read out.
- Then, the switch is turned downwards to read out the bits in the lower delay line.

- Associated with a linear (n, k) code is the dual code of dimension $n - k$.
- The dual code is a linear $(n, n - k)$ code with 2^{n-k} vectors which is the null space of the (n, k) code.
- The generator matrix for the dual code, denoted by $\mathbf{H}$, consists of $n - k$ linearly independent code vectors selected from the null space.
- As a result, any code word $\mathbf{C}_m$ of the (n, k) code is orthogonal to any code word in the dual code. Put differently, any code word $\mathbf{C}_m$ of the (n, k) code is orthogonal to every row of $\mathbf{H}$, i.e.

$$\mathbf{C}_m \mathbf{H}^{\mathrm{T}} = \mathbf{0},$$

  where $\mathbf{0}$ denotes an all-zero row vector with $n - k$ elements.
- Choosing the code words $[1\,0\,0\,\ldots\,0]$, $[0\,1\,0\,\ldots\,0]$, etc., we conclude that we have

$$\mathbf{G}\mathbf{H}^{\mathrm{T}} = \mathbf{0},$$

  where $\mathbf{0}$ is now an all-zero matrix of dimension $k \times (n - k)$.
- In the case of a systematic (n, k) code and a generator matrix $\mathbf{G}$ in the systematic form, we obtain from $\mathbf{G}\mathbf{H}^{\mathrm{T}} = \mathbf{0}$ that

$$\mathbf{H} = \left[-\mathbf{P}^{\mathrm{T}} \,\middle|\, \mathbf{I}_{n-k} \right].$$

- The negative sign in $-\mathbf{P}^{\mathrm{T}}$ may be dropped when dealing with binary codes (modulo-2 subtraction is identical to modulo-2 addition).

Example: We reconsider the $(7, 4)$ code with generator matrix

$$\mathbf{G} = [\mathbf{I}_4 | \mathbf{P}] = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right].$$

Then, the matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \left[-\mathbf{P}^{\mathrm{T}} \middle| \mathbf{I}_3\right] = \left[\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right],$$

i.e., the identity matrix $\mathbf{I}_3$ is applied to the parity check bits. The product $\mathbf{C}_m\mathbf{H}^{\mathrm{T}} = \mathbf{0}$ yields three equations given by

$$\begin{aligned} 0 &= x_{m1} + x_{m2} + x_{m3} + c_{m5} \\ 0 &= x_{m2} + x_{m3} + x_{m4} + c_{m6} \\ 0 &= x_{m1} + x_{m2} + x_{m4} + c_{m7}. \end{aligned}$$

Contrast this with the equations for the parity check bits:

$$\begin{aligned} c_{m5} &= x_{m1} + x_{m2} + x_{m3} \\ c_{m6} &= x_{m2} + x_{m3} + x_{m4} \\ c_{m7} &= x_{m1} + x_{m2} + x_{m4}. \end{aligned}$$

- Thus, the product $\mathbf{C}_m\mathbf{H}^{\mathrm{T}}$ is equivalent to adding the parity check bits to the corresponding linear combinations of the information bits used to compute c_{mj} for $j = 5, 6, 7$.
- If we have a received code word $\mathbf{Y}$ instead of $\mathbf{C}_m$, we can use the matrix $\mathbf{H}$ to check that $\mathbf{Y}$ satisfies the condition $\mathbf{Y}\mathbf{H}^{\mathrm{T}} = \mathbf{0}$.
- Therefore, it is appropriate to call $\mathbf{H}$ the parity check matrix associated with the (n, k) code.
- Since $\mathbf{C}_m\mathbf{H}^{\mathrm{T}} = \mathbf{0}$, the column vectors of $\mathbf{H}$ are linearly dependent.
- Suppose $\mathbf{C}_j$ denotes the minimum weight code word of a linear (n, k) code satisfying (as all other code words) $\mathbf{C}_j\mathbf{H}^{\mathrm{T}} = \mathbf{0}$. Since the minimum weight is equal to the minimum distance, it follows that $d_{\min}$ of the columns of $\mathbf{H}$ are linearly dependent.
- Alternatively, $d_{\min} - 1$ columns of $\mathbf{H}$ are linearly independent. Since the rank of $\mathbf{H}$ is at most $n - k$, we have $n - k \geq d_{\min} - 1$ and obtain an upper bound on the minimum distance:

$$d_{\min} \leq n - k + 1.$$

- We can used certain modifications (e.g. extension and shortening of codes) discussed below.

Code extension

- Given a linear binary (n, k) code with minimum distance $d_{\min}$, we can construct a linear binary $(n+1, k)$ code by appending one additional parity check bit to each code word.
- The check bit is usually selected to be a check bit on all the bits in the code word. Thus, the added check bit is a 0 if the original code word has an even number of ones and it is 1 if the original code word has an odd number of ones.
- Consequently, if the minimum weight and, hence, the minimum distance of the code is odd, the added parity check bit increases the minimum distance by 1.
- We call the $(n+1, k)$ code an extended code. Its parity check matrix is

$$\mathbf{H}_e = \begin{bmatrix} & & & & 0 \\ & & & & 0 \\ & & \mathbf{H} & & \vdots \\ & & & & 0 \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix},$$

where $\mathbf{H}$ is the parity check matrix of the original code.

Code shortening

- A systematic (n, k) can also be shortened by setting a number of information bits to zero. In this way, a linear (n, k) code consisting of k information bis and $n - k$ check bits can be shortened into an $(n - \ell, k - \ell)$ linear code by setting the first ℓ bits to zero. Clearly, these ℓ bits are not transmitted.

- The $n - k$ check bits are computed in the usual manner, as in the original code. Since for the original code, we have $\mathbf{C}_m = \mathbf{X}_m \mathbf{G}$, the effect of setting the first ℓ bits of $\mathbf{X}_m$ to zero is equivalent to removing the first ℓ rows of $\mathbf{G}$.

- Equivalently, since for the original code, we have $\mathbf{C}_m \mathbf{H}^{\mathrm{T}} = \mathbf{0}$, we may remove the first ℓ columns of $\mathbf{H}$.

- The shortened $(n - \ell, k - \ell)$ code consists of $2^{k-\ell}$ code words. The minimum distance of these $2^{k-\ell}$ code words is at least as large as the minimum distance of the original (n, k) code.

Specific linear block codes: Hamming codes

- We only consider binary Hamming codes with

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

  where m is any positive integer. For example, when $m = 3$, we have a $(7, 4)$ code.
- The parity check matrix $\mathbf{H}$ of a Hamming code has a special simple structure and consists of all non-zero column vectors of the corresponding dimension.

Example (Hamming code in systematic form):

$$\mathbf{H} = \left[-\mathbf{P}^{\mathrm{T}} \middle| \mathbf{I}_3 \right] = \left[\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array} \right].$$

- No two columns of $\mathbf{H}$ are linearly dependent. However, for $m > 1$, it is possible to find three columns of $\mathbf{H}$ that add to zero. Thus, $d_{\min} = 3$ for an (n, k) Hamming code.

Specific linear block codes: Hadamard codes

- A Hadamard code is obtained by selecting as code words the rows of a Hadamard matrix.
- A Hadamard matrix is an $n \times n$ matrix (where n is an even integer) of ones and zeros with the property that any row differs from any other row in exactly $n/2$ positions. One row of the matrix contains all zeros. The other rows contain $n/2$ zeros and $n/2$ ones.
- For $n = 2$, the Hadamard matrix is

$$\mathbf{M}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

- Furthermore, from $\mathbf{M}_n$ we can generate the Hadamard matrix $\mathbf{M}_{2n}$ according to the relation

$$\mathbf{M}_{2n} = \begin{bmatrix} \mathbf{M}_n & \mathbf{M}_n \\ \mathbf{M}_n & \bar{\mathbf{M}}_n \end{bmatrix},$$

  where $\bar{\mathbf{M}}_n$ denotes the complement (zeros and ones exchanged) of $\mathbf{M}_n$.
- Using this relation, we obtain

$$\mathbf{M}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Specific linear block codes: Hadamard codes

- The complement of $\mathbf{M}_4$ is

$$\bar{\mathbf{M}}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

- The rows of $\mathbf{M}_4$ and $\bar{\mathbf{M}}_4$ form a linear binary code of block length $n = 4$ having $2n = 8 = 2^k$ code words. The minimum distance of the code is $d_{\min} = n/2 = 2$.

- By repeated application we can generate Hadamard codes with block length $n = 2^m$, $k = \mathrm{ld}(2n) = 1 + \mathrm{ld}(n) = m + 1$ and $d_{\min} = n/2 = 2^{m-1}$, where m is a positive integer.

- In addition to the important special cases where $n = 2^m$, Hadamard codes of other block lengths are possible, but the codes are not linear.

Cyclic codes

Cyclic codes are a subset of the class of linear codes that satisfy the following

Cyclic shift property: If $\mathbf{C} = [c_{n-1}\, c_{n-2}\, \cdots c_1\, c_0]$ is a code word of a cyclic code, then $[c_{n-2}\, c_{n-3}\, \cdots c_0\, c_{n-1}]$, obtained by a cyclic shift of the elements of $\mathbf{C}$, is also a code word.

- As the result of the cyclic property, the codes possess a considerable amount of structure which can be exploited in the encoding and decoding operations.
- In dealing with cyclic codes, it is convenient to associate with a code word $\mathbf{C} = [c_{n-1}\, c_{n-2}\, \cdots c_1\, c_0]$ a polynomial $C(p)$ of degree $\leq n-1$, defined as

$$C(p) = c_{n-1}p^{n-1} + c_{n-2}p^{n-2} + \cdots + c_1 p + c_0.$$

We will consider only binary codes, i.e., each of the coefficients of the polynomial is either zero or one. Furthermore, the summation of terms ap^m and bp^m with equal powers m results in a term cp^m where c is the modulo-2 summation of a and b.

Cyclic codes

- Now suppose we do a left shift of the polynomial by multiplying it by p, i.e., we form the polynomial

$$pC(p) = c_{n-1}p^n + c_{n-2}p^{n-1} + \cdots + c_1p^2 + c_0p.$$

  This polynomial cannot represent a code word, since its degree may be equal to n (if $c_{n-1} = 1$).

- However, if we divide $pC(p)$ by $p^n + 1$, we obtain

$$\begin{aligned}
\frac{pC(p)}{p^n+1} &= \frac{c_{n-1}p^n + c_{n-2}p^{n-1} + \cdots + c_1p^2 + c_0p}{p^n+1} \\
&= \frac{c_{n-1}(p^n+1) + c_{n-2}p^{n-1} + \cdots + c_1p^2 + c_0p + c_{n-1}}{p^n+1} \\
&= c_{n-1} + \frac{C_1(p)}{p^n+1}.
\end{aligned}$$

- Note that the polynomial $C_1(p)$ represents the code word $\mathbf{C}_1 = [c_{n-2}\, c_{n-3}\, \cdots c_0\, c_{n-1}]$, which is just the code word $\mathbf{C}$ shifted cyclically by one position to the left.

Cyclic codes

- Since $C_1(p)$ is the remainder obtained by dividing $pC(p)$ by $p^n + 1$, we say that

$$C_1(p) = pC(p) \mod (p^n + 1).$$

- Thus, if $C(p)$ represents a code word, then $p^i C(p) \mod (p^n + 1)$ is also a code word of the cyclic code:

$$p^i C(p) = Q(p)(p^n + 1) + C_i(p),$$

  where the remainder polynomial $C_i(p)$ represents a code word of the cyclic code and $Q(p)$ is the quotient.

- We can generate a cyclic code by using a generator polynomial $g(p)$ of degree $n - k$. The generator polynomial of an (n, k) cyclic code is a **factor of** $p^n + 1$ and has the general form

$$g(p) = p^{n-k} + g_{n-k-1}p^{n-k-1} + \cdots + g_1 p + 1,$$

  where $g_i \in \{0, 1\}$ for $i = 1, \ldots, n - k - 1$.

- We also define a message polynomial $X(p)$ as

$$X(p) = x_{k-1}p^{k-1} + x_{k-2}p^{k-2} + \cdots + x_1 p + x_0,$$

  where $[x_{k-1}\, x_{k-2}\, \cdots\, x_1\, x_0]$ represent the k information bits.

Cyclic codes

- Clearly, the polynomial $X(p)g(p)$ of degree $\leq n-1$ may represent a code word. In total, there are 2^k polynomials $\{X_i(p)\}$ and hence there are 2^k possible code words that can be formed from a given $g(p)$.
- Suppose we denote these code words as

$$C_m(p) = X_m(p)g(p), \qquad \text{for } m = 1, 2, \ldots, 2^k.$$

- We want to show that these code words satisfy the cyclic shift property. To this end, a cyclic shift of $C_m(p)$ produces

$$C_n(p) = pC_m(p) + c_{n-1}(p^n + 1),$$

and, since $g(p)$ divides both $p^n + 1$ **and** $C_m p$, it also divides $C_n(p)$. Thus, $C_n(p)$ can be represented as

$$C_n(p) = X_n(p)g(p).$$

- Therefore, a cyclic shift of any code word $C_m(p)$ generated by $C_m(p) = X_m(p)g(p)$ yields another code word.

Cyclic codes

- In summary, code words possessing the cyclic property can be generated by multiplying the 2^k message polynomials with a unique polynomial $g(p)$ called the generator polynomial of the (n, k) cyclic code, which divides $p^n + 1$ and has degree $n - k$.
- The cyclic code generated in this manner is a subspace S_c of the vector space S. The dimension of S_c is k. Thus, we have a certain analogy to the generator matrices discussed before.

Example:

- Consider a code with block length $n = 7$. The polynomial $p^7 + 1$ has the following factors:

$$p^7 + 1 = (p + 1)\left(p^3 + p^2 + 1\right)\left(p^3 + p + 1\right).$$

- Since $g(p)$ must divide $p^7 + 1$ and there are three irreducible factors, we have 2^3 choices for $g(p)$:

1. $g(p) = 1$: Obviously, this generator polynomial creates all possible code words $\mathrm{GF}(2)^7$.
2. $g(p) = p + 1$: This generator polynomial results in a parity check code.
3. $g(p) = p^3 + p + 1$: gives a $(7, 4)$ code (Hamming code)
4. $g(p) = p^3 + p^2 + 1$: gives a $(7, 4)$ code (Hamming code being equivalent to 3.)

Cyclic codes

TABLE 8.1–2
(7, 4) Cyclic code
Generator polynomial: $g_1(p) = p^3 + p^2 + 1$

Information bits				Code words						
p^3	p^2	p^1	p^0	p^6	p^5	p^4	p^3	p^2	p^1	p^0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1	0	1
0	0	1	0	0	0	1	1	0	1	0
0	0	1	1	0	0	1	0	1	1	1
0	1	0	0	0	1	1	0	1	0	0
0	1	0	1	0	1	1	1	0	0	1
0	1	1	0	0	1	0	1	1	1	0
0	1	1	1	0	1	0	0	0	1	1
1	0	0	0	1	1	0	1	0	0	0
1	0	0	1	1	1	0	0	1	0	1
1	0	1	0	1	1	1	0	0	1	0
1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	1	0	1	1	1	0	0
1	1	0	1	1	0	1	0	0	0	1
1	1	1	0	1	0	0	0	1	1	0
1	1	1	1	1	0	0	1	0	1	1

Cyclic codes

5. $g(p) = (p+1)\left(p^3+p+1\right) = p^4+p^3+p^2+1$: gives a $(7,3)$ code (dual of the $(7,4)$ Hamming code, cf. discussion of dual codes below)
6. $g(p) = (p+1)\left(p^3+p^2+1\right) = p^4+p^2+p+1$: gives a $(7,3)$ code (dual of the $(7,4)$ Hamming code)
7. $g(p) = \left(p^3+p+1\right)\left(p^3+p^2+1\right) = p^6+p^5+p^4+p^3+p^2+p+1$: this gives a repetition code
8. $g(p) = (p+1)\left(p^3+p+1\right)\left(p^3+p^2+1\right) = p^7+1$: in this case, $X(p) = 0$ and we obtain the all-zero code word

In general, we can factor the polynomial p^n+1 according to

$$p^n + 1 = g(p)h(p),$$

where $g(p)$ denotes the generator polynomial for the (n,k) cyclic code and $h(p)$ denotes the parity polynomial that has degree k. The latter may be used to generate the dual code.

Cyclic codes

- The reciprocal polynomial of $h(p)$ is defined by

$$\begin{aligned} p^k h\left(p^{-1}\right) &= p^k\left(p^{-k} + h_{k-1}p^{-k+1} + h_{k-2}p^{-k+2} + \cdots + h_1 p^{-1} + 1\right) \\ &= 1 + h_{k-1}p + h_{k-2}p^2 + \cdots + h_1 p^{k-1} + p^k. \end{aligned}$$

Problem

Show that the reciprocal polynomial is a factor of $p^n + 1$.

- Since $p^k h\left(p^{-1}\right)$ is a factor of $p^n + 1$ and of degree k, it is the generator polynomial of an $(n, n-k)$ cyclic code.
- This cyclic code is the dual code to the (n, k) code generated from $g(p)$, i.e., the $(n, n-k)$ dual code constitutes the null space of the (n, k) cyclic code.

Example:

- The parity polynomial for the $(7, 4)$ cyclic code generated from $g(p) = p^3 + p^2 + 1$ is given by $h(p) = (p+1)\left(p^3 + p + 1\right) = p^4 + p^3 + p^2 + 1$, as mentioned before. The resulting reciprocal polynomial is given by $p^4 h\left(p^{-1}\right) = 1 + p + p^2 + p^4$.
- One can show that the code words of the dual code resulting from the reciprocal polynomial are orthogonal to the code words in the $(7, 4)$ cyclic code. Note that neither the $(7, 4)$ code nor the $(7, 3)$ code are systematic.

Cyclic codes

TABLE 8.1–3
(7, 3) Dual code
Generator polynomial: $p^4 h_1(p^{-1}) = p^4 + p^2 + p + 1$

Information bits			Code words						
p^2	p^1	p^0	p^6	p^5	p^4	p^3	p^2	p^1	p^0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1	0
0	1	1	0	1	1	1	0	0	1
1	0	0	1	0	1	1	1	0	0
1	0	1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0	1	0
1	1	1	1	1	0	0	1	0	1

Cyclic codes

- It is desirable to show how a generator matrix can be obtained from the generator polynomial of a cyclic (n, k) code.
- To that end, we have to find k linearly independent code words. These can be easily generated by the linearly independent polynomials

$$\left\{p^{k-1}g(p), p^{k-2}g(p), p^{k-3}g(p), \ldots, pg(p), g(p)\right\}.$$

- The code words associated with these polynomials form a basis of dimension k for the (n, k) cyclic code.

Example:

- The four rows of the generator matrix for the $(7, 4)$ cyclic code with generator polynomial $g_1(p) = p^3 + p^2 + 1$ are obtained from the polynomials

$$p^i g_1(p) = p^{3+i} + p^{2+i} + p^i \qquad \text{for } i = 3, 2, 1, 0.$$

- We obtain the following generator matrix:

Cyclic codes

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Similarly, the generator matrix for the $(7, 4)$ cyclic code generated by the polynomial $g_2(p) = p^3 + p + 1$ is

$$\mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

The parity check matrices for $\mathbf{G}_1$ and $\mathbf{G}_2$ can be constructed in the same manner using the respective reciprocal polynomials.

Problem

Construct the parity check matrices $\mathbf{H}_1$ and $\mathbf{H}_2$ for the above generator matrices $\mathbf{G}_1$ and $\mathbf{G}_2$ using the corresponding reciprocal polynomials and constructing a simple basis for the null spaces of the cyclic codes (mind the dimension of $\mathbf{H}_1$ and $\mathbf{H}_2$).

For the systematic form of the generator matrix of a cyclic code $\Rightarrow$ J.G. Proakis.

Optimum soft-decision decoding of linear block codes

- Consider the transmission of a linear block code over an AWGN channel where the receiver uses optimum i.e. unquantized soft-decision decoding.
- We only consider BPSK modulation with coherent detection.
- Let ε denote the transmitted signal energy per code word with n bit per code word.
- Thus, $\varepsilon = n\varepsilon_c$ with ε_c denoting the energy to transmit a single coded bit (i.e. one element of the code word).
- Since each code word conveys k bit of information, we have for the energy per information bit

$$\varepsilon_b = \frac{\varepsilon}{k} = \frac{n\varepsilon_c}{k} = \frac{\varepsilon_c}{R_c},$$

  where $R_c = k/n$ denotes the code rate.
- The code words are assumed equally likely a priori with prior probability $1/M$.
- We assume a correlator for the calculation of the decision variable of each bit followed by a combiner calculating a correlation metric for each of the M hypotheses (i.e. the M code words).

Optimum soft-decision decoding of linear block codes

- Let r_j, $j = 1, 2, \ldots, n$, represent the n sampled outputs of the matched filter for any particular code word.
- Since we assume BPSK modulation with coherent detection, the output r_j may be expressed either as

$$r_j = \sqrt{\varepsilon_c} + n_j,$$

when the jth bit of a code word is a one, or as

$$r_j = -\sqrt{\varepsilon_c} + n_j,$$

when the jth bit of a code word is a zero.
- The variables $\{n_j\}$ represent additive white Gaussian noise at the sampling instants. Each n_j has zero-mean and variance $N_0/2$.
- Equipped with the knowledge of the M possible code words, upon reception of the n samples r_j, $j = 1, 2, \ldots, n$, the receiver calculates the M correlation metrics

$$CM_i = C(\mathbf{r}, \mathbf{C}_i) = \sum_{j=1}^{n} (2c_{ij} - 1) r_j, \qquad i = 1, 2, \ldots, M,$$

where c_{ij} denotes the bit in the jth position of the ith code word.

Optimum soft-decision decoding of linear block codes

- Note that $2c_{ij} - 1 = 1$ if $c_{ij} = 1$ and $2c_{ij} - 1 = -$ if $c_{ij} = 0$.
- As a result, the correlation metric of the actual transmitted code word has a mean value $\sqrt{\varepsilon_c} n$, the other metrics have smaller mean values.
- Since we have a linear block code, the probability of error for the transmission of the mth code word in a binary-input symmetric channel such as the AWGN channel with soft-decision decoding is the same for all m.
- Hence, we assume for simplicity that the all-zero code word $\mathbf{C}_1$ is transmitted.
- For correct decoding of $\mathbf{C}_1$, the correlation metric CM_1 must exceed all other $M - 1$ correlation metrics CM_j with $j = 2, \ldots, M$.
- All correlation metrics are Gaussian distributed. The mean value of CM_1 is $\sqrt{\varepsilon_c} n$ while the mean values of CM_m is $\sqrt{\varepsilon_c} n\big(1 - 2\frac{w_m}{n}\big)$, where w_m denotes the weight of the mth code word.
- The variance of each decision variable is $N_0/2$.
- Unlike in the case of detecting an uncoded bit for BPSK transmission over an AWGN channel, the exact expression of the probability of error depends on the correlations among the M code words. The cross-correlation coefficients between $\mathbf{C}_1$ and $\mathbf{C}_m$ is given by

$$\rho_m = 1 - \frac{2w_m}{n} \in [-1, 1], \qquad m = 2, \ldots, M.$$

Optimum soft-decision decoding of linear block codes

- Instead of calculating the exact error probability, we resort to a union bound.
- The probability that $CM_m > CM_1$ is

$$P_2(m) = Q\left(\sqrt{\frac{\varepsilon}{N_0}(1-\rho_m)}\right),$$

where $\varepsilon = k\varepsilon_b$ is the transmitted energy per waveform.

- Substitution into the equation provides

$$P_2(m) = Q\left(\sqrt{\frac{2\varepsilon_b}{N_0}R_c w_m}\right) = Q\left(\sqrt{2\gamma_b R_c w_m}\right),$$

where γ_b is the SNR per bit and R_c is the code rate.

- We can now find an upper (union) bound on the probability of a word error according to

$$P_M \leq \sum_{m=2}^{M} P_2(m) = \sum_{m=2}^{M} Q\left(\sqrt{2\gamma_b R_c w_m}\right).$$

- This bound obviously requires knowledge of the weight distribution of the code. Clearly, we can exploit the fact that $w_m \geq \min_m w_m = d_{\min}$ and the monotonicity of the Q-function. Upon insertion, we obtain $\Rightarrow$

Optimum soft-decision decoding of linear block codes

$$Q\left(\sqrt{2\gamma_b R_c w_m}\right) \leq Q\left(\sqrt{2\gamma_b R_c d_{\min}}\right) < \exp\left(-\gamma_b R_c d_{\min}\right),$$

where we have used $Q\left(\sqrt{x}\right) < \exp\left(-x/2\right)$ for $x \geq 0$.

- Furthermore, since $M - 1 < M = 2^k \Leftrightarrow \ln(M) < k\ln(2)$, we obtain

$$\begin{aligned} P_M &\leq \sum_{m=2}^{M} Q\left(\sqrt{2\gamma_b R_c w_m}\right) \leq (M-1) Q\left(\sqrt{2\gamma_b R_c d_{\min}}\right) \\ &< M \exp\left(-\gamma_b R_c d_{\min}\right) = \exp\left(-\gamma_b R_c d_{\min} + k\ln(2)\right). \end{aligned}$$

- In order to compare the *code word error* probability P_M with *bit-error* probabilities P_b, we would have to take into account the, in general, complicated relation between the both error measures.
- To obtain a rough idea about the improvement by coding, we assume instead $P_b \approx P_M/2$. Since for uncoded BPSK transmission, we have $P_b < \frac{1}{2}\exp\left(-\gamma_b\right)$, we obtain for $P_b < \frac{1}{2}\exp\left(-\gamma_b(R_c d_{\min} - k\ln(2)/\gamma_b)\right)$ an approximate coding gain of $10\log\left(R_c d_{\min} - k\ln(2)/\gamma_b\right)$.
- The coding gain depends on the code parameters as well as on the SNR per bit γ_b.

Hard-decision decoding of linear block codes

- The soft-decision decoding procedure described above is optimum, it shows a large complexity, in particular when the number of code words M is large, since we have to compute M correlation metrics

$$CM_i = C(\mathbf{r}, \mathbf{C}_i) = \sum_{j=1}^{n} (2c_{ij} - 1) r_j, \qquad i = 1, 2, \ldots, M.$$

  If $M > 2^{10}$, this is highly complex.

- To reduce the computational burden, the analog samples can be quantized and the decoding operations are then performed digitally. That is, a hard decision is made as to whether each transmitted bit in a code word is a zero or a one.
- As a result, we have a BSC with cross-over probability of p. For BPSK, we obtain

$$p = Q\left(\sqrt{\frac{2\varepsilon_c}{N_0}}\right) = Q\left(\sqrt{2\gamma_b R_c}\right).$$

- As discussed on slide #85, the maximum-likelihood decoder results in a minimum probability of a code word error for the BSC.

Hard-decision decoding of linear block codes

- A simple (but computationally inefficient) method for decoding is to first add (modulo 2) the received code word vector to all M possible transmitted code words $\mathbf{C}_i$ to obtain the error vectors $\mathbf{e}_i$.
- Clearly, the number of errors in transforming $\mathbf{C}_i$ into the received code word is just equal to the number of ones in $\mathbf{e}_i$. The ML decoding scheme requires to decide in favor of the smallest weight error vector.
- A more efficient method for hard-decision decoding makes use of the parity check matrix $\mathbf{H}$. Assume that we transmit the code word $\mathbf{C}_m$ and the received vector at the output of the detector (with binary valued components) is $\mathbf{Y}$.
- If an error occurs, we can write $\mathbf{Y} = \mathbf{C}_m + \mathbf{e}$.
- The parity check equations are given by

$$\begin{aligned} \mathbf{Y}\mathbf{H}^{\mathrm{T}} &= (\mathbf{C}_m + \mathbf{e})\mathbf{H}^{\mathrm{T}} \\ &= \mathbf{C}_m\mathbf{H}^{\mathrm{T}} + \mathbf{e}\mathbf{H}^{\mathrm{T}} \\ &= \mathbf{e}\mathbf{H}^{\mathrm{T}} = \mathbf{S}, \end{aligned}$$

where the $(n-k)$-dimensional vector $\mathbf{S}$ is called the syndrome of the error pattern $\mathbf{e}$.

Hard-decision decoding of linear block codes

- The vector $\mathbf{S}$ has components that are zero for all parity check equations that are satisfied and nonzero for all parity check equations that are not satisfied.
- Thus, $\mathbf{S}$ contains the pattern of failures in the parity checks.
- Note that the syndrome $\mathbf{S}$ is a characteristic of the error pattern and not of the transmitted code word.
- There are 2^n possible error patterns, but only 2^{n-k} syndromes. Consequently, different error patterns result in the same syndrome. How to deal with this fact?
- Construction of a so-called standard array:

$$\begin{array}{llll}
\mathbf{C}_1 & \mathbf{C}_2 & \mathbf{C}_3 & \dots \quad \mathbf{C}_{2^k} \\
\mathbf{e}_2 & \mathbf{C}_2 + \mathbf{e}_2 & \mathbf{C}_3 + \mathbf{e}_2 & \dots \quad \mathbf{C}_{2^k} + \mathbf{e}_2 \\
\mathbf{e}_3 & \mathbf{C}_2 + \mathbf{e}_3 & \mathbf{C}_3 + \mathbf{e}_3 & \dots \quad \mathbf{C}_{2^k} + \mathbf{e}_3 \\
\vdots & \vdots & \vdots & \vdots \\
\mathbf{e}_{2^{n-k}} & \mathbf{C}_2 + \mathbf{e}_{2^{n-k}} & \mathbf{C}_3 + \mathbf{e}_{2^{n-k}} & \dots \quad \mathbf{C}_{2^k} + \mathbf{e}_{2^{n-k}}
\end{array}$$

- Each row is called a coset and the first (leftmost) code word (or error pattern) is called a coset leader.
- Therefore, a coset consists of all the possible received code words resulting from a particular error pattern (coset leader).

Hard-decision decoding of linear block codes

EXAMPLE 8.1–10. Let us construct the standard array for the (5, 2) systematic code with generator matrix given by

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

This code has a minimum distance $d_{\min} = 3$. The standard array is given in Table 8.1–7. Note that in this code, the coset leaders consist of the all-zero error pattern, five error patterns of weight 1, and two error patterns of weight 2. Although many more double error patterns exist, there is only room for two to complete the table. These were selected such that their corresponding syndromes are distinct from those of the single error patterns.

TABLE 8.1–7
Standard array for the (5, 2) code

Code words			
0 0 0 0 0	**0 1 0 1 1**	**1 0 1 0 1**	**1 1 1 1 0**
0 0 0 0 1	0 1 0 1 0	1 0 1 0 0	1 1 1 1 1
0 0 0 1 0	0 1 0 0 1	1 0 1 1 1	1 1 1 0 0
0 0 1 0 0	0 1 1 1 1	1 0 0 0 1	1 1 0 1 0
0 1 0 0 0	0 0 0 1 1	1 1 1 0 1	1 0 1 1 0
1 0 0 0 0	1 1 0 1 1	0 0 1 0 1	0 1 1 1 0
1 1 0 0 0	1 0 0 1 1	0 1 1 0 1	0 0 1 1 0
1 0 0 1 0	1 1 0 0 1	0 0 1 1 1	0 1 1 0 0

Hard-decision decoding of linear block codes

Problem

Show that the mapping from the error patterns in Table 8.1-7 to the corresponding syndromes is given by the following table:

TABLE 8.1–8
Syndrome table for the (5, 2) code

Syndrome	Error pattern
0 0 0	0 0 0 0 0
0 0 1	0 0 0 0 1
0 1 0	0 0 0 1 0
1 0 0	0 0 1 0 0
0 1 1	0 1 0 0 0
1 0 1	1 0 0 0 0
1 1 0	1 1 0 0 0
1 1 1	1 0 0 1 0

Hard-decision decoding of linear block codes

Clearly, not all error pattern can be corrected as is shown in

EXAMPLE 8.1–11. Consider the $(5, 2)$ code with the standard array given in Table 8.1–7. The syndromes versus the most likely error patterns are given in Table 8.1–8. Now suppose the actual error vector on the channel is

$$\mathbf{e} = [1 \quad 0 \quad 1 \quad 0 \quad 0]$$

The syndrome computed for the error is $\mathbf{S} = [0 \ 0 \ 1]$. Hence, the error determined from the table is $\hat{\mathbf{e}} = [0 \ 0 \ 0 \ 0 \ 1]$. When $\hat{\mathbf{e}}$ is added to $\mathbf{Y}$, the result is a decoding error. In other words the $(5, 2)$ code corrects all single errors and only two double errors, namely $[1 \ 1 \ 0 \ 0 \ 0]$ and $[1 \ 0 \ 0 \ 1 \ 0]$.

Subsequently, we will consider the decoding of systematic cyclic codes in greater detail.

Hard-decision decoding of linear block codes

Let us consider a cyclic code:

1. First, we want to find the systematic form of the code and see how the encoding can be done.
2. Second, we investigate how the encoding can be implemented using shift registers.
3. Third, we investigate how a syndrome decoding can be implemented for a systematic cyclic code.

Systematic form of cyclic codes

- Remember that the generator matrix of a systematic linear code is given by

$$\mathbf{G} = [\mathbf{I}_k | \mathbf{P}].$$

In terms of the polynomials discussed in the derivation of cyclic codes, this means that the ℓth row of $\mathbf{G}$ corresponds to a polynomial of the form

$$p^{n-\ell} + R_\ell(p), \qquad \ell = 1, 2, \ldots, k,$$

where $R_\ell(p)$ is a polynomial of degree less than $n - k$.

- This form can be obtained by dividing $p^{n-\ell}$ by $g(p)$:

$$\frac{p^{n-\ell}}{g(p)} = Q_\ell(p) + \frac{R_\ell(p)}{g(p)}, \qquad \ell = 1, 2, \ldots, k.$$

with $Q_\ell(p)$ denoting the quotient polynomial.

Systematic form of cyclic codes

- This is equivalent to

$$p^{n-\ell} = Q_\ell(p)g(p) + R_\ell(p), \qquad \ell = 1, 2, \ldots, k.$$

  Adding $R_\ell(p)$ on both sides provides

$$p^{n-\ell} + R_\ell(p) = Q_\ell(p)g(p), \qquad \ell = 1, 2, \ldots, k,$$

  i.e., $p^{n-\ell} + R_\ell(p)$ is a code word of the cyclic code.

- As a result, the desired polynomial corresponding to the ℓth row of $\mathbf{G}$ is

$$p^{n-\ell} + R_\ell(p).$$

Systematic form of cyclic codes

Example: Consider the $(7, 4)$ cyclic code discussed on slide #113 with generator polynomial $g_2(p) = p^3 + p + 1$. We find

$$\begin{aligned} p^6 &= (p^3 + p + 1)g_2(p) + p^2 + 1 \\ p^5 &= (p^2 + 1)g_2(p) + p^2 + p + 1 \\ p^4 &= pg_2(p) + p^2 + p \\ p^3 &= g_2(p) + p + 1. \end{aligned}$$

As a result, we obtain the generator matrix in the systematic form as

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

with corresponding parity check matrix (obtained as in the case of usual linear codes)

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Encoding a cyclic code

- The aforementioned method for constructing the generator matrix of the systematic form can be used to create the systematic code words directly from the message polynomials $X(p)$ and the generator polynomial $g(p)$.
- To that end, suppose that we multiply $X(p)$ by p^{n-k} corresponding to a left shift of $X(p)$ by $n-k$ bits:

$$p^{n-k}X(p) = x_{k-1}p^{n-1} + x_{k-2}p^{n-2} + \cdots + x_1 p^{n-k+1} + x_0 p^{n-k}.$$

- We want to interpret this polynomial as the part describing the first k bits in a systematic code word $C(p)$.
- Obviously, we have $n-k$ degrees of freedom to choose the polynomial containing the coefficients for powers p^{n-k-1}, p^{n-k-2}, $\ldots$ p, p^0.
- To find this polynomial, we divide the above one by $g(p)$, i.e.

$$\frac{p^{n-k}X(p)}{g(p)} = Q(p) + \frac{r(p)}{g(p)},$$

  $Q(p)$ ($r(p)$) being a quotient (remainder) polynomial of degree $\leq k-1$ ($\leq n-k-1$).
- Equivalently, we have

$$p^{n-k}X(p) = Q(p)g(p) + r(p).$$

Encoding a cyclic code

- Note that $Q(p)g(p)$ is a code word of the cyclic code. Thus, upon adding $r(p)$ to both sides of $p^{n-k}X(p) = Q(p)g(p) + r(p)$, we obtain the desired systematic code word

$$p^{n-k}X(p) + r(p).$$

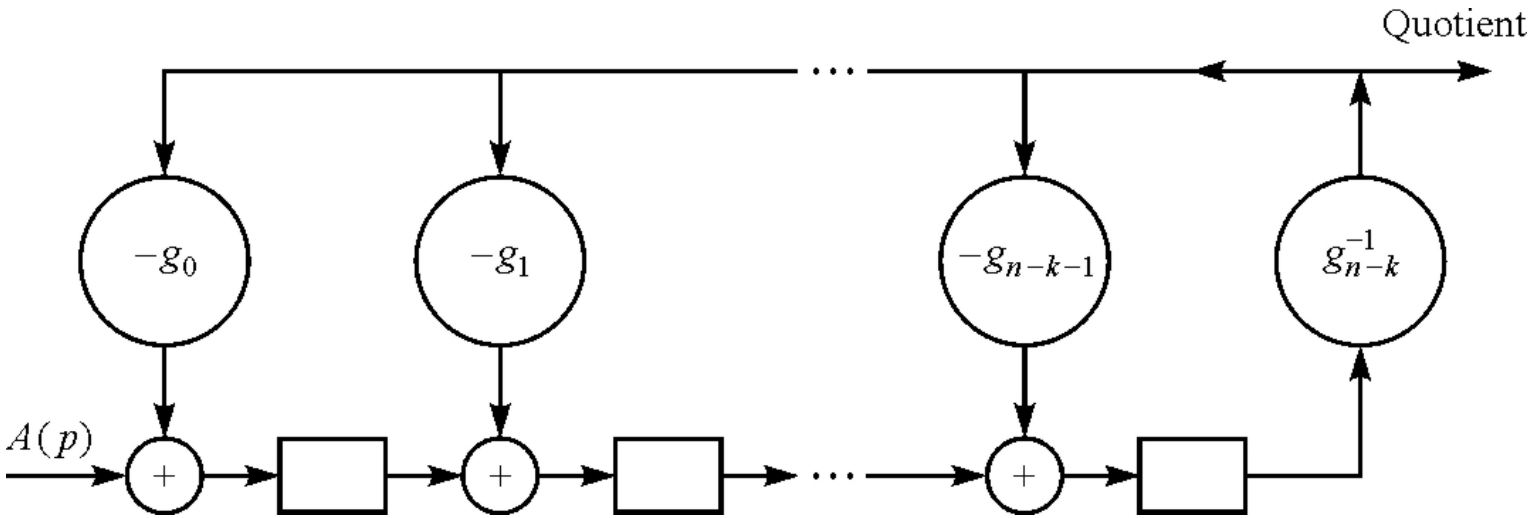
- In summary, the systematic cyclic code may be generated by
  1. multiplying the message polynomial $X(p)$ by p^{n-k} (left shift by k bits)
  2. dividing $p^{n-k}X(p)$ by $g(p)$ to obtain the remainder $r(p)$
  3. adding $r(p)$ to $p^{n-k}X(p)$.
- The first and the third steps are trivial operations, while the division by $g(p)$ is not. How can we implement the calculation of $r(p)$?

Encoding a cyclic code

Example:
Assume we are given a polynomial $A(p) = p^6 + p^5 + p^3$ of degree 6($= n-1$ with $n = 7$, later we will choose $A(p) = p^{n-k}X(p)$) and divide it by the polynomial $g(p) = p^3 + p + 1$ (of degree $n - k = 3$.

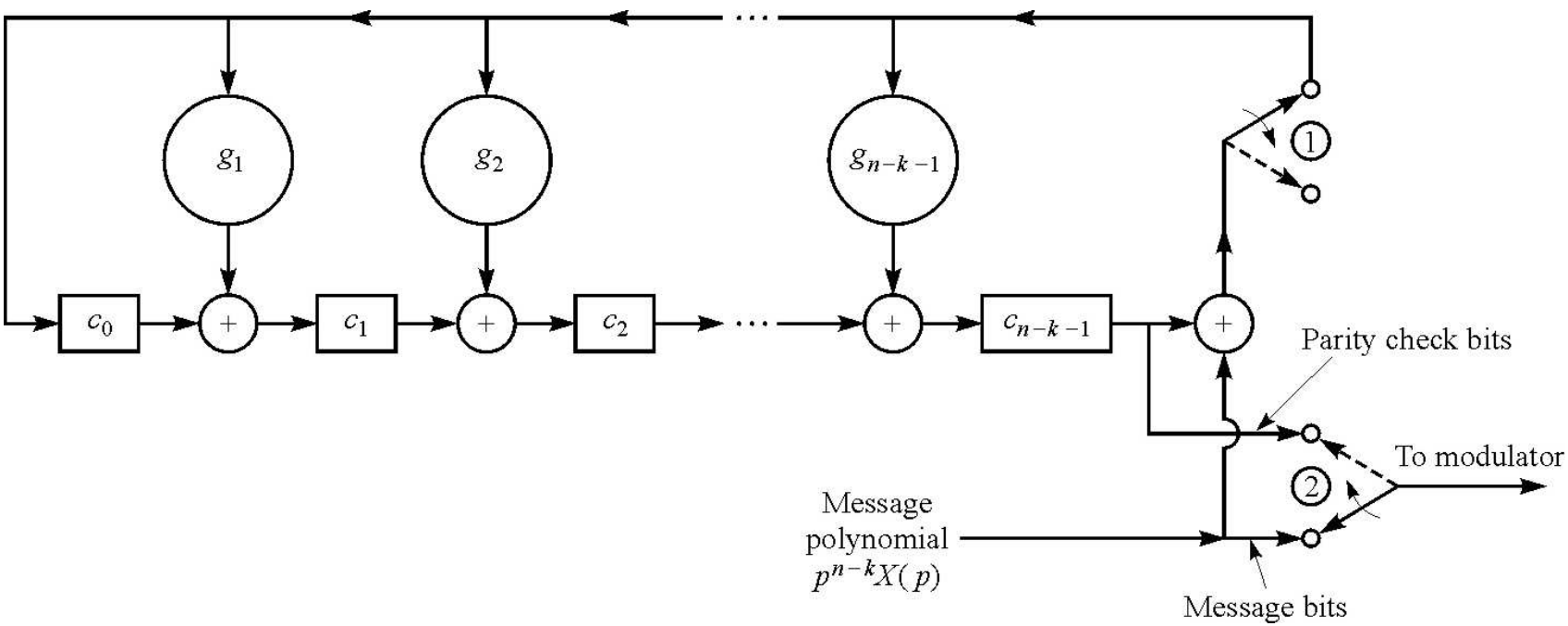
We obtain by polynomial division

$$\frac{A(p)}{g(p)} = \frac{p^6 + p^5 + p^3}{p^3 + p + 1} = p^3 + p^2 + p + 1 + \frac{1}{p^3 + p + 1}.$$



A feedback shift register for dividing $A(p)$ by $g(p)$.

Encoding a cyclic code



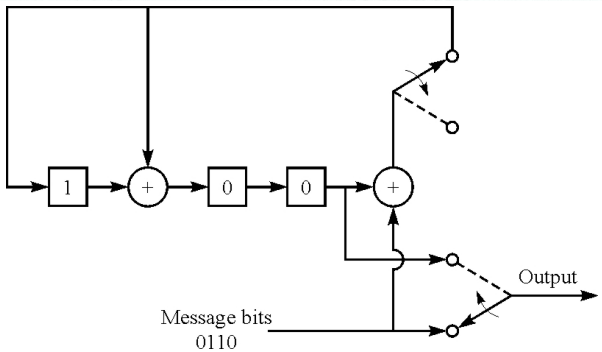
Encoding a cyclic code by use of the generator polynomial $g(p)$.

Encoding a cyclic code

EXAMPLE 8.1–8. The shift register for encoding the (7, 4) cyclic code with generator polynomial $g(p) = p^3 + p + 1$ is illustrated in Figure 8.1–4. Suppose the input message bits are 0110. The contents of the shift register are as follows:

Input	Shift	Shift register contents		
	0	0	0	0
0	1	0	0	0
1	2	1	1	0
1	3	1	0	1
0	4	1	0	0

Hence, the three parity check bits are 100, which correspond to the code bits $c_5 = 0$, $c_6 = 0$, and $c_7 = 1$.



Syndrome decoding of systematic cyclic codes

- Consider a systematic cyclic code, where a received vector $\mathbf{Y}$ is represented by the polynomial $Y(p)$.
- In general, $\mathbf{Y} = \mathbf{C} + \mathbf{e}$ with the transmitted code word $\mathbf{C}$ and the error vector $\mathbf{e}$.
- Hence, $Y(p) = C(p) + e(p) = X(p)g(p) + e(p)$. Upon division by $g(p)$, we obtain

$$\frac{Y(p)}{g(p)} = Q(p) + \frac{R(p)}{g(p)}$$

  or, equivalently,

$$Y(p) = Q(p)g(p) + R(p),$$

  where $R(p)$ is a polynomial of degree less than or equal to $n - k - 1$.
- Equating the both expressions for $Y(p)$, we have

$$e(p) = [X(p) + Q(p)]g(p) + R(p).$$

- Obviously, the remainder $R(p)$ obtained from dividing $Y(p)$ by $g(p)$ depends only on the error polynomial $e(p)$.
- Thus, $R(p) = S(p)$ is the syndrome associated with the error pattern $\mathbf{e}$.

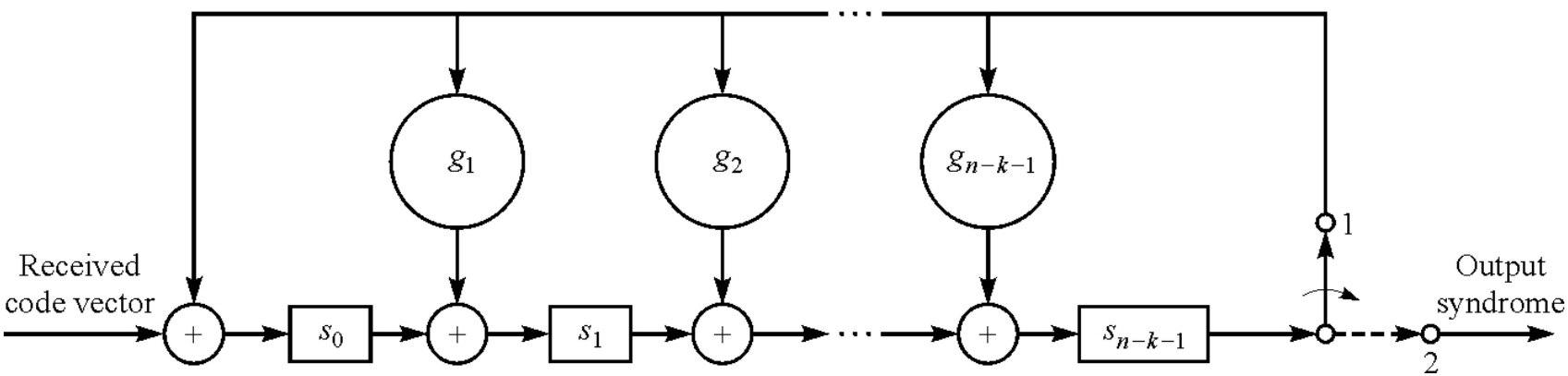
Syndrome decoding of systematic cyclic codes

- In analogy with the standard arrays for linear codes, we can write

$$Y(p) = Q(p)g(p) + S(p),$$

  where $S(p)$ is the syndrome polynomial of degree $\leq n-k-1$.

- If $g(p)$ divides $Y(p)$ exactly, then $S(p) = 0$ and the received decoded word is $\hat{\mathbf{C}}_m = \mathbf{Y}$. For a general error vector $\mathbf{e}$, we have $\hat{\mathbf{C}}_m = \mathbf{Y} + \mathbf{e}$.



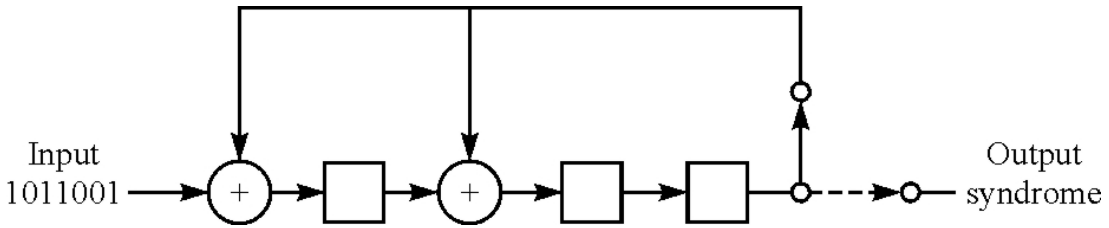
An $(n-k)$-stage shift register for computing the syndrome.

Syndrome decoding of systematic cyclic codes

EXAMPLE 8.1–12. Let us consider the syndrome computation for the $(7, 4)$ cyclic Hamming code generated by the polynomial $g(p) = p^3 + p + 1$. Suppose that the received vector is $\mathbf{Y} = [1\ 0\ 0\ 1\ 1\ 0\ 1]$. This is fed into the three-stage register shown in Figure 8.1–10. After seven shifts, the contents of the shift register are 110, which corresponds to the syndrom $\mathbf{S} = [0\ 1\ 1]$. The most probable error vector corresponding to this syndrome is $\mathbf{e} = [0\ 0\ 0\ 1\ 0\ 0\ 0]$ and, hence,

$$\hat{\mathbf{C}}_m = \mathbf{Y} + \mathbf{e} = [1\ 0\ 0\ 0\ 1\ 0\ 1]$$

The information bits are 1 0 0 0.



Error detection and error correction capability

Error detection:

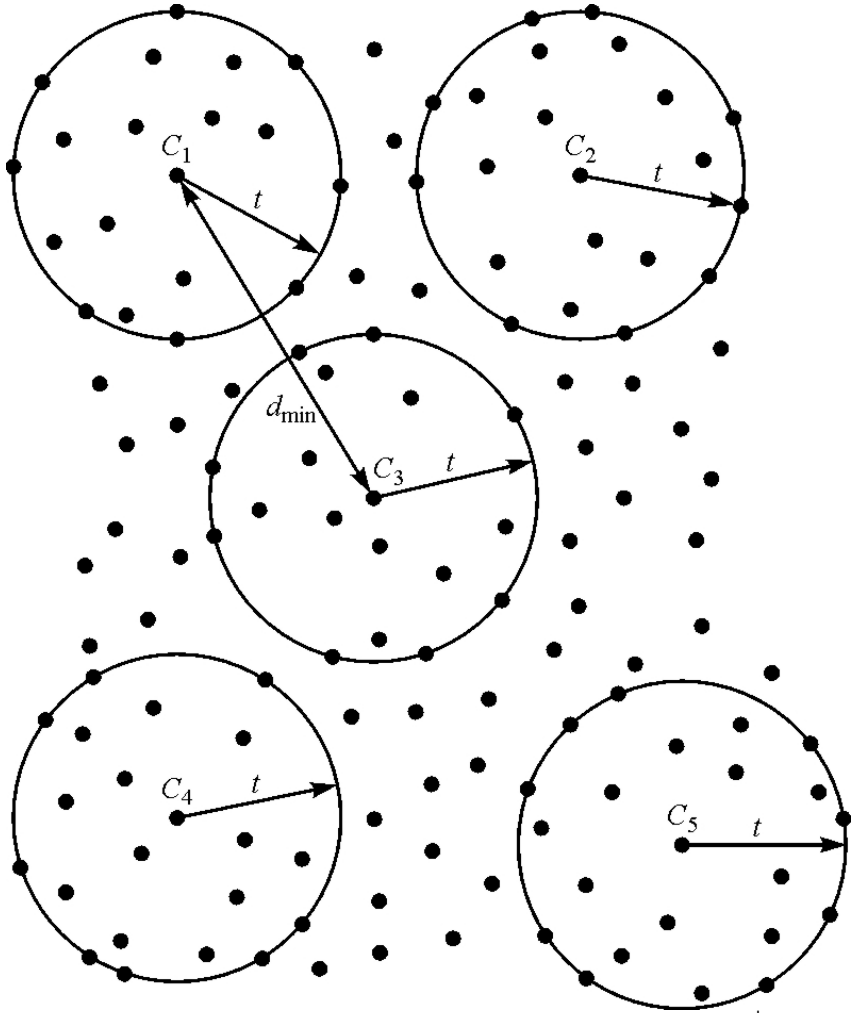
- From the discussion above, when the syndrome is the zero vector, the received word is one of the 2^k possible transmitted code words.
- Since the minimum separation between a pair of code words is $d_{\min}$, it is possible for an error pattern of weight $d_{\min}$ to transform one of these 2^k code words in the code into another code word.
- In this case, we have an undetected error.
- If the actual number of errors is less than $d_{\min}$, the syndrome will have a nonzero weight. When this occurs, we have detected the presence of one or more errors on the channel.
- Clearly, the (n, k) block code is capable of detecting $d_{\min} - 1$ errors. Note that detecting $d_{\min} - 1$ errors does not mean that the number of errors would be known at the receiver. It simply means that a non-valid code word has been recognized by the receiver.
- Error detection may be used in conjunction with an automatic repeat-request (ARQ) scheme for retransmission of the code word in error.

Error detection and error correction capability

Error correction:

- The error correction capability of the code also depends on the minimum distance. However, the number of correctable error patterns is clearly limited by the number of possible syndromes (or coset leaders in the standard array).
- We interpret the 2^k possible code words as elements in an n-dimensional vector space, where each code word is viewed as the centre of a sphere of radius t (all distance measures represent Hamming distances). The value of t is the largest value so that the spheres are non-overlapping.
- As a result, $t = \left\lfloor \frac{1}{2}(d_{\min} - 1) \right\rfloor$, where $\lfloor x \rfloor$ denotes the largest integer contained in x. Within each sphere lie all the possible received words of distance less than or equal to t from the valid code word.
- Consequently, any received code vector that falls within a sphere is decoded into the valid code word at the centre of the sphere.
- We say that the (n, k) code with minimum distance $d_{\min}$ is capable of correcting $t = \left\lfloor \frac{1}{2}(d_{\min} - 1) \right\rfloor$ errors.

Error detection and error correction capability



A representation of code words as centers of spheres of radius $t = \left\lfloor \frac{1}{2}(d_{\min} - 1) \right\rfloor$.

Error detection and error correction capability

- Clearly, to correct $t = \left\lfloor \frac{1}{2}(d_{\min} - 1) \right\rfloor$ errors, we first have to detect them. For example, for a code with $d_{\min} = 7$, we could define spheres of radius $t = 3$ around each code word. If a received code word falls within a certain sphere, we decode the centre of the sphere as the transmitted code word.
- Can we increase the error detection capability of the code? Obviously, if we keep $t = 3$, it is not possible to do so, since as soon as a received code word is inside a certain sphere, the syndrome in the corresponding set would decode the vector to the word in the sphere centre.
- However, if we reduce the radius of the spheres, there will be code words between the spheres which, if received, will make the receiver to ask for a retransmission. As an example, assume $d_{\min} = 7$ and $t = 2$. Obviously, around each sphere centre, we have *orbits* of radius 3 or radius 4 which do not overlap with the spheres around neighbored code words. If more than four errors occur they will not be detected if they fall into another sphere.
- Similarly, we can decrease the sphere radius to $t = 1$ and detect five errors.
- In general, a code with minimum distance $d_{\min}$ can detect e_d errors and correct e_c errors, where

$$e_d + e_c \leq d_{\min} - 1 \qquad \text{and} \qquad e_c \leq e_d.$$

Probability of error based on error correction

- Assume that we consider exclusively error correction for the case of a binary symmetric channel.
- The decoder will decode correctly if (but not necessarily only if) we choose a sphere radius of $t = \left\lfloor \frac{1}{2}(d_{\min} - 1) \right\rfloor$ and the number of errors in the code word is $\leq t$.
- The probability of m errors in a block of n bits is

$$P(m, n) = \begin{pmatrix} n \\ m \end{pmatrix} p^m (1-p)^{n-m}$$

and, therefore, the probability of a code word error is upper-bounded by the expression

$$P_M \leq \sum_{m=t+1}^{n} P(m, n).$$

- Equality in the previous equation holds if the linear block code is a perfect code.
- In order to relate t to the parameters n and k we note that each sphere around a code word contains the set of all code words of Hamming distance less than or equal to t from the code word. The number of code words in the sphere is thus

$$1 + \begin{pmatrix} n \\ 1 \end{pmatrix} + \begin{pmatrix} n \\ 2 \end{pmatrix} + \cdots + \begin{pmatrix} n \\ t \end{pmatrix} = \sum_{i=0}^{t} \begin{pmatrix} n \\ i \end{pmatrix}.$$

Probability of error based on error correction

- There are 2^k possible transmitted code words and, consequently, at most $M = 2^k$ non-overlapping spheres each having a radius t.
- At the same time, the total number of possible code words in all spheres cannot exceed the 2^n possible code words.
- Thus, a t-error correcting code must satisfy the inequality

$$2^k \sum_{i=0}^{t} \binom{n}{i} \leq 2^n$$

or, equivalently,

$$2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i}.$$

- A perfect code has the property that all spheres of Hamming distance $t = \left\lfloor \frac{1}{2}(d_{\min} - 1) \right\rfloor$ around the $M = 2^k$ possible code words are disjoint and every received code word falls in one of the spheres.
- Thus, every code word is at most at distance t from one of the possible transmitted code words and the above inequality holds with equality.

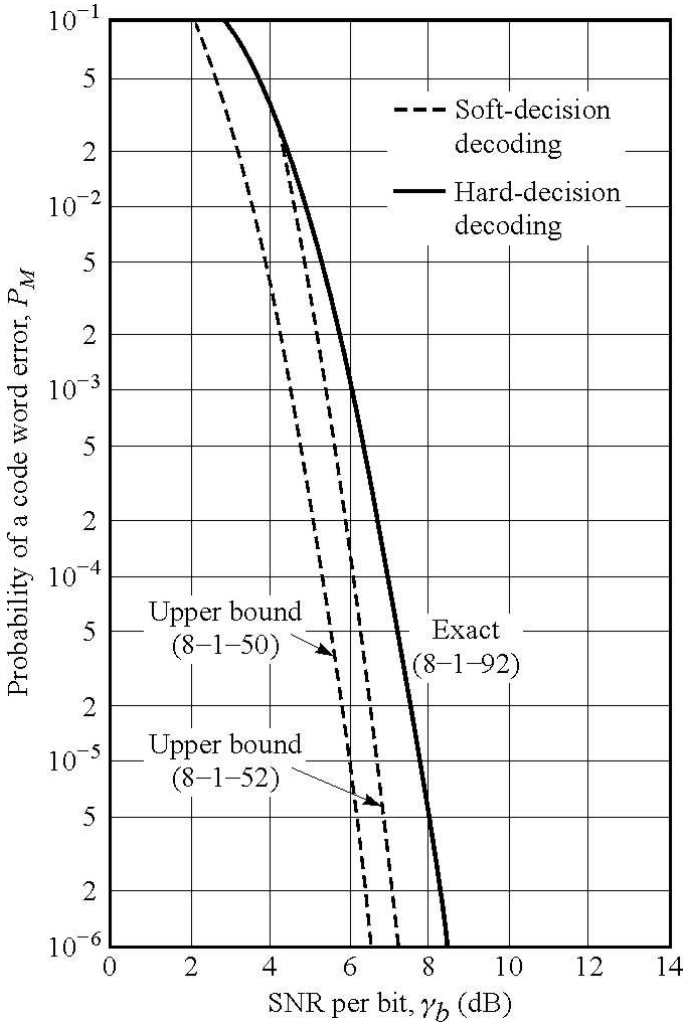
Probability of error based on error correction

- For a perfect code, all error patterns of weight $\leq t$ are corrected by the optimum (minimum distance) decoder. On the other hand, any error pattern of weight $t+1$ or greater cannot be corrected.
- Consequently, the probability of a code word error is

$$P_M = \sum_{m=t+1}^{n} P(m,n).$$

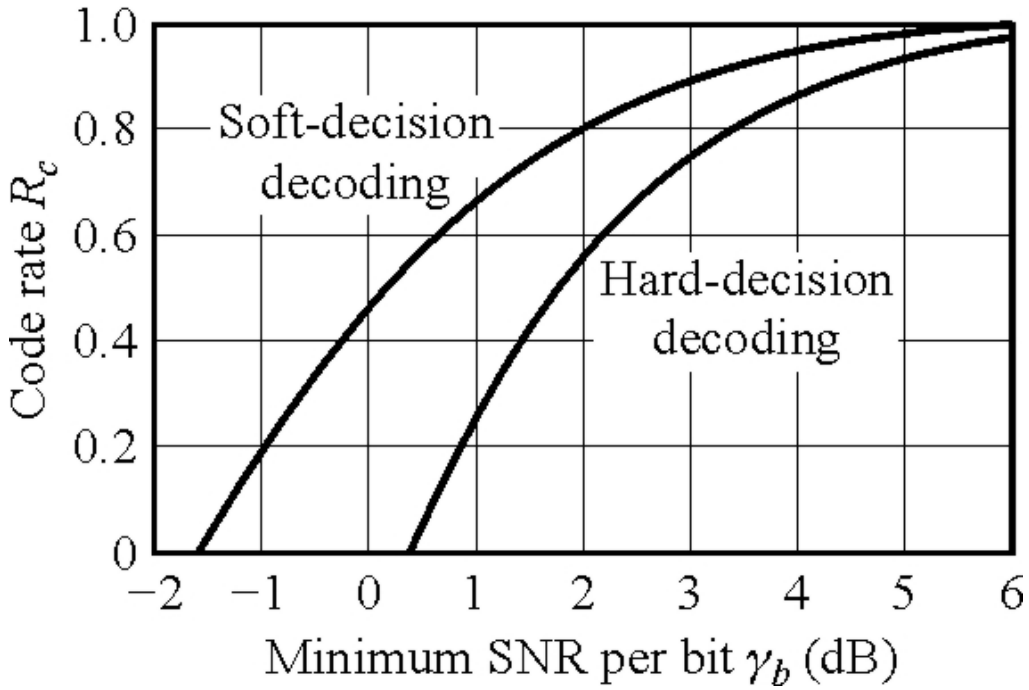
- There are only three different perfect codes:
  1. the Golay $(23, 12)$ code having $d_{\min} = 7$ and $t = 3$
  2. Hamming codes with $n = 2^{n-k} - 1$, $d_{\min} = 3$ and $t = 1$
  3. trivial codes with two code words of odd length n and $d_{\min} = n$.
- These codes are optimum in the BSC in the sense that they result in minimum error probability among all codes with the same block length and the same number of information bits.

Comparison of soft- and hard-decision



Comparison of soft-decision decoding with hard-decision decoding for the Golay $(23, 12)$ code.

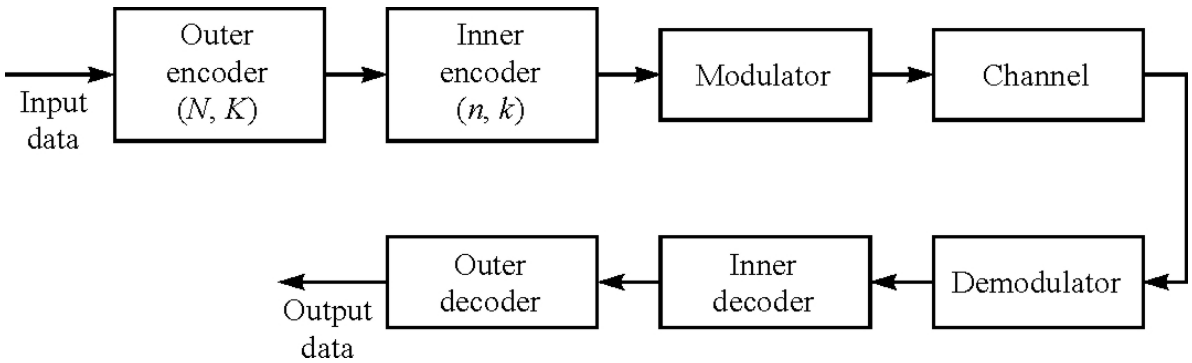
Comparison of soft- and hard-decision



Code rate as a function of the minimum SNR per bit for soft- and hard-decision decoding.

Concatenated Block Codes

- A way to increase the minimum distance of block codes is the so-called code concatenation.
- A concatenated code consists of two separate codes which are combined to form a larger code.
- Usually, one code is selected to be non-binary (N, K) outer code and the other is a binary (n, k) inner code.



Block diagram of a communication system employing a concatenated code.

- Code words are formed by subdividing a block of kK information bits into K groups, called symbols, where each symbol consists of k bits.

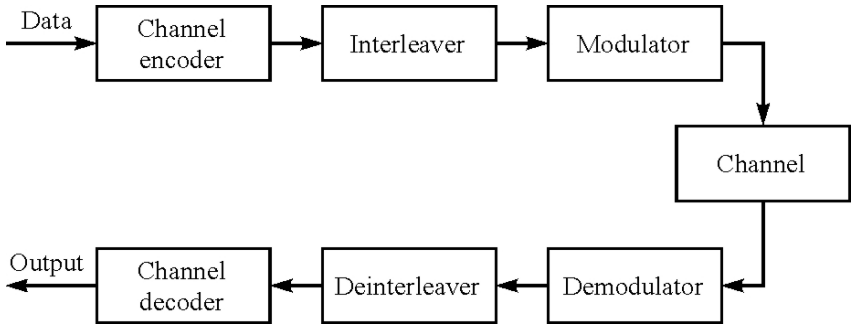
Concatenated Block Codes

- As usual, the K k-bit symbols are encoded into N k-bit symbols by the outer encoder.
- The inner coder takes each k-bit symbol and encodes it into a binary block code of length n. The result is an equivalent (Nn, Kk) long binary code.
- The minimum distance of the concatenated code is $D_{\min} d_{\min}$, where $D_{\min}$ is the minimum distance of the outer code and $d_{\min}$ is the minimum distance of the inner code.
- The overall code rate is Kk/Nn which is the product of the individual rates.
- The decoding can be done based on both soft- or hard-decision schemes. Usually, the inner decoder makes a decision on the k bits using a minimum-distance (= maximum-likelihood) decoding. When a block of N k-bit symbols is received from the inner decoder, the outer decoder makes a hard decision on the K k-bit symbols based on ML decoding.

Interleaving of coded data for channels with burst errors

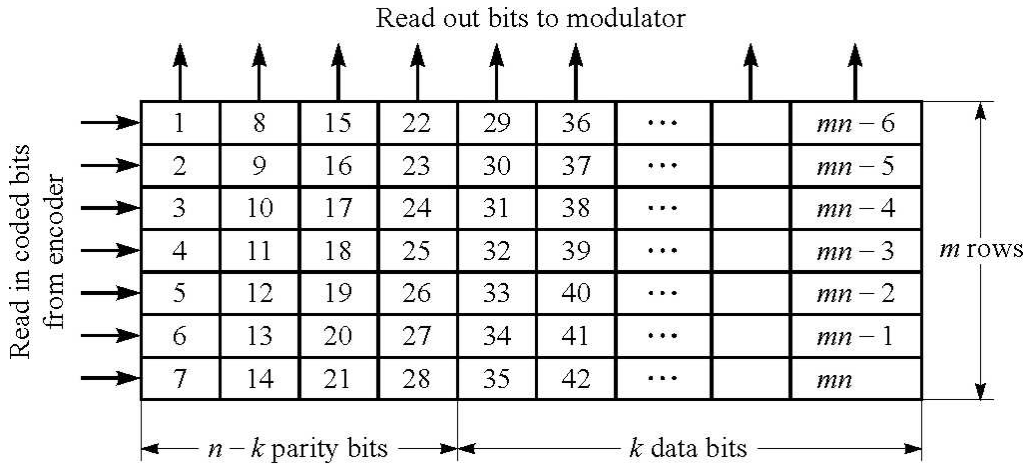
- Most of the well-known codes have been devised for increasing the reliability in the transmission of information over channels without memory where errors are statistically independent. An example of such a channel is the AWGN channel.
- However, there are important practical cases where this does not hold. An example is a fading channel where the SNR per bit γ_b is a time-variant parameter. If γ_b falls below a critical value, errors arise in the transmission and these errors are bursty in nature (sequences of errors) due to the fact that the signal level variations have memory.
- Usually, error bursts are not corrected by codes that are designed to perform well in channels with statistically independent errors.
- It can be shown that a systematic (n, k) code with $n - k$ parity check bits can correct bursts of length $b < \left\lfloor \frac{1}{2}(n-k) \right\rfloor$. Therefore, we are interested in breaking up error bursts in order to reduce the burst length.
- A standard way to reduce the burst length is by interleaving where the data are reordered prior to transmission over the channel.
- Upon deinterleaving at the receiver, error bursts are spread out in time so that errors within a code word seem to be independent.

Interleaving of coded data for channels with burst errors



Block diagram of a system with interleaving.

- The interleaver can take different forms, a block structure or a convolutional structure. Here, we only consider block interleavers.
- A block interleaver formats the encoded data in a rectangular array of m rows and n columns. Usually, each row of the array constitutes a code word of length n.



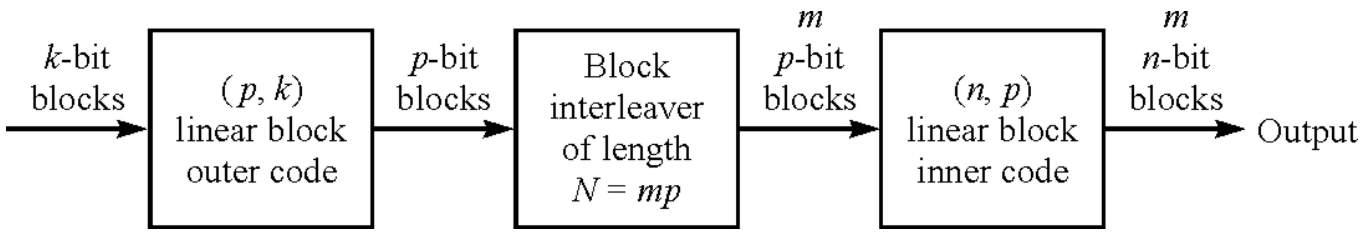
A block interleaver of degree m for coded data.

Interleaving of coded data for channels with burst errors

- An interleaver of degree m takes m code words (row-wise) of length n and transmits the bits in a column-wise order.
- At the receiver, the data are stored in a corresponding array from which they are read out row-wise again and fed to the decoder.
- In this way, a burst of errors of length $\ell = mb$ is broken up into m bursts of length b.
- Thus, an (n, k) code that can handle bursts of length $b < \left\lfloor \frac{1}{2}(n-k) \right\rfloor$ can be combined with an interleaver of degree m to create an interleaved (mn, mk) block code that can handle bursts of length mb.
- An interleaver can be used together with code concatenation in order to produce codes with extremely long code words as discussed next.

Serial and parallel concatenated block codes

- In a serial concatenated block code, the interleaver is inserted between two block encoders.

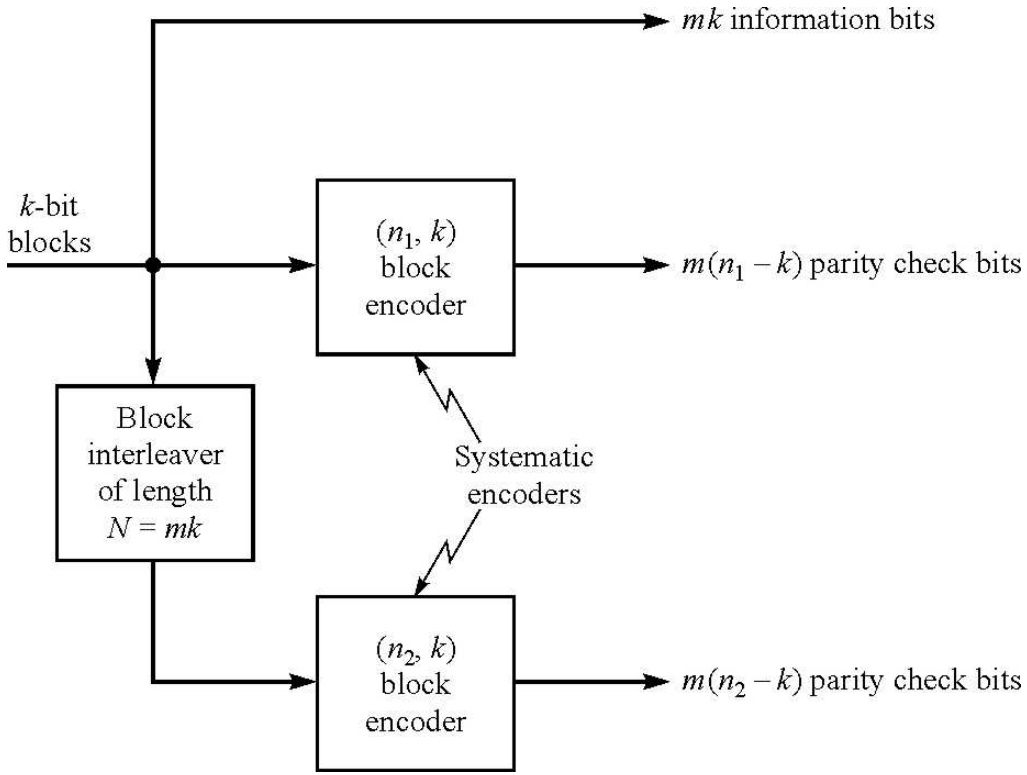


Serial concatenated block code with interleaver.

- The outer (p, k) code is combined with an inner (n, p) code. The block interleaver length is selected as $N = mp$, where m is the number of outer code words.
- Encoding: mk information bits are encoded by the outer encoder to produce mp coded bits. These $N = mp$ coded bits are permuted by the interleaver and fed to the inner encoder which outputs mn coded bits.
- As in the case without inserted interleaving, the overall code rate $R_c = \frac{k}{p}\frac{p}{n} = \frac{k}{n}$ is the product of the individual code rates. However, the block length of the code is nm which can be significantly larger than in the case without interleaving.

Serial and parallel concatenated block codes

- In a parallel concatenated block code, the interleaver is used to produce two block encoded streams which provide parity check bits in addition to the uncoded bit stream.



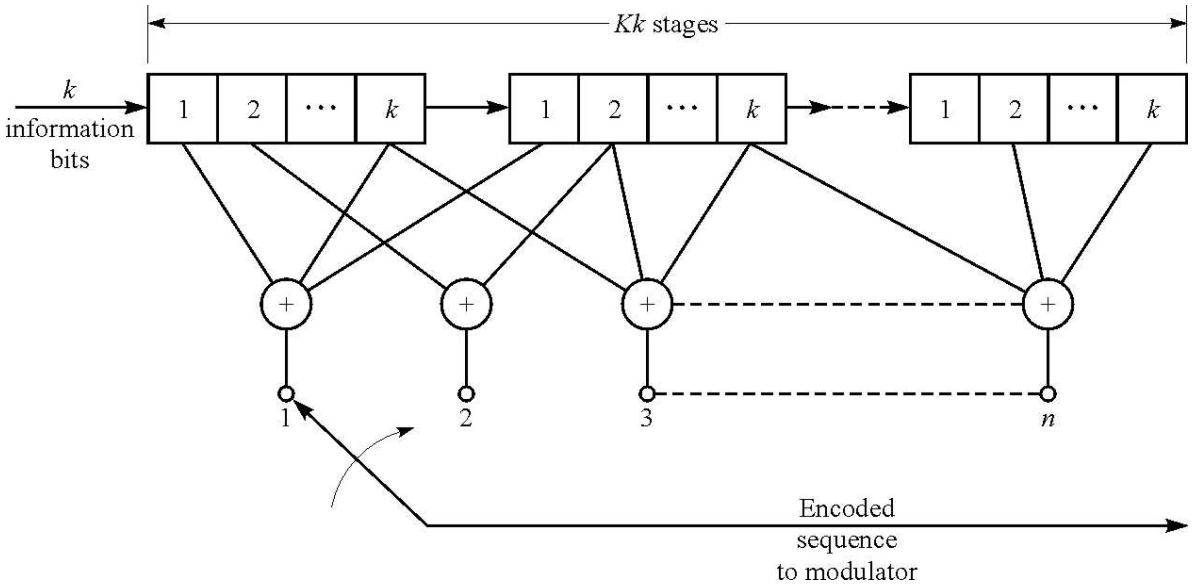
Parallel concatenated block code with interleaver.

Serial and parallel concatenated block codes

- The two encoders are systematic binary linear encoders providing an (n_1, k) and an (n_2, k) code, respectively. Note that the information is only transmitted once (uncoded bit stream), while the two encoders just output the parity check bits.
- In general, the interleaver has a length $N = mk$. Thus, mk information bits are encoded into $m(n_1 + n_2 - k)$ bits so that the rate of the code is $R_c = \frac{k}{n_1+n_2-k}$.
- The code words have a large length and are relatively sparse. Decoding is done in an iterative way using soft-in/soft-out maximum a posteriori probability (MAP) algorithms.
- The performance of the codes is very high (error rates of about $10^{-4} \ldots 10^{-5}$ near the Shannon limit.

Convolutional codes

- A convolutional code is generated by passing the information sequence to be transmitted through a linear finite-state shift register.
- In general, the shift register consists of K k-bit stages and n linear algebraic function generators.



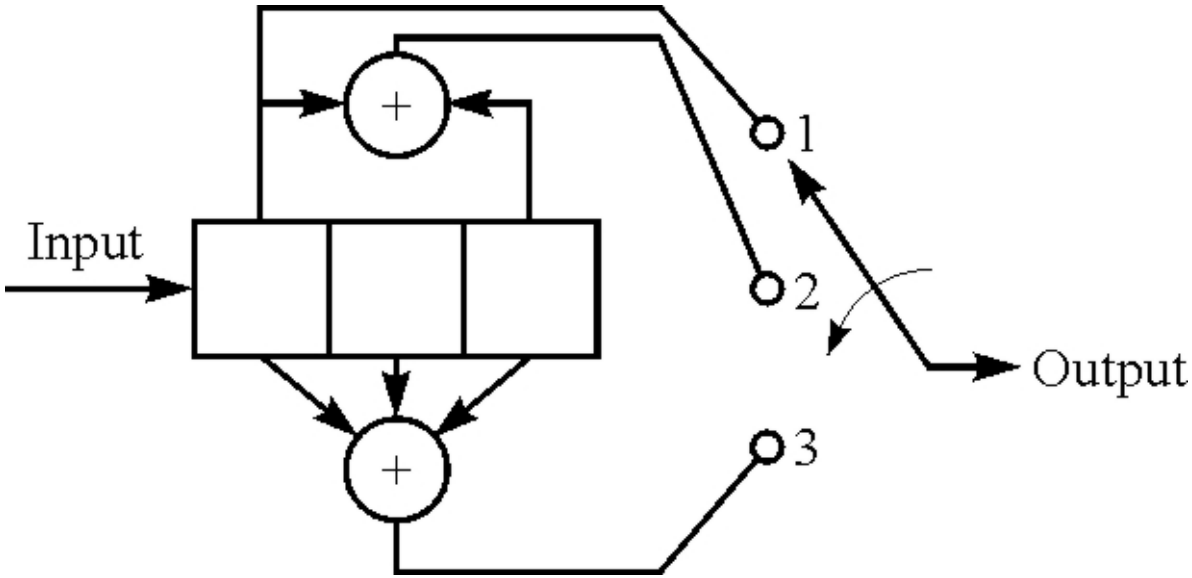
Convolutional encoder.

- The number of output bits for each k-bit input sequence is n, so that the code rate is defined as in the case of block codes by $R_c = k/n$.
- The parameter K is called the constraint length of the code. Often, the constraint length is given as the number of bits in the shift registers ($= Kk$ above) rather than as the number of k-bit bytes. In this case, the constraint length does not have to be a multiple of k.

- One method to describe a convolutional code is to give its generator matrix. A problem with this method is the fact that the input sequences are semi-infinite leading to semi-infinite dimensional generator matrices.

- An alternative description is based on the structure of the encoder itself. Here, we specify a set of n vectors, one for each of the n modulo-2 adders.

- Each vector has Kk dimensions and contains the connections of the shift registers to that modulo-2 adder (a one indicates a connection to the stage, a zero indicates no connection).



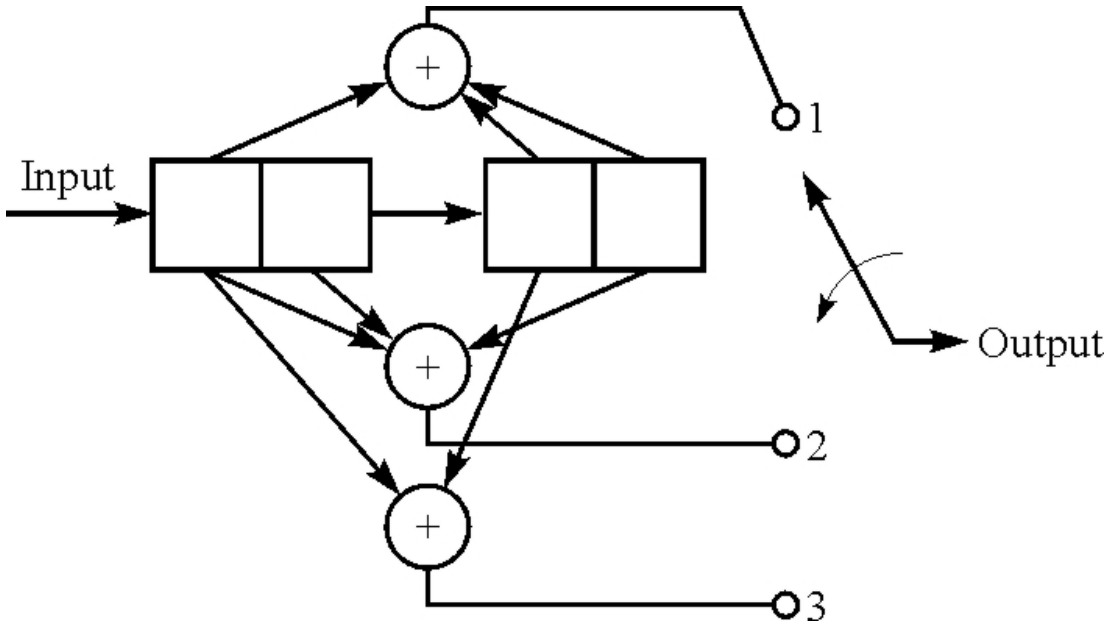
Convolutional encoder with $K = 3$, $k = 1$ and $n = 3$.

- Here, $K = 3$, $k = 1$ and $n = 3$. Initially, the shift register is in the all-zero state.

- The generator vectors are given by

$$\mathbf{g}_1 = [100], \qquad \mathbf{g}_2 = [101], \qquad \mathbf{g}_3 = [111].$$

- In most cases, the generators are given more compactly in octal form as $(4, 5, 7)$.



Convolutional encoder with $K = 2$, $k = 2$ and $n = 3$.

- The generator vectors are given by

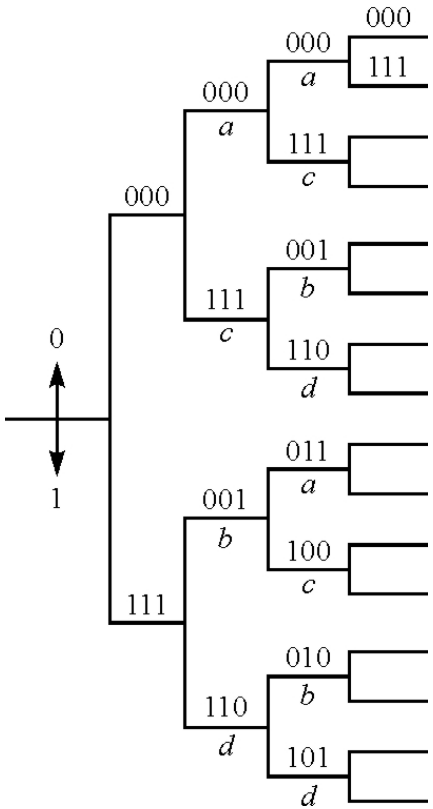
$$\mathbf{g}_1 = [1011], \qquad \mathbf{g}_2 = [1101], \qquad \mathbf{g}_3 = [1010]$$

or $(13, 15, 12)$ in octal representation.

The tree diagram

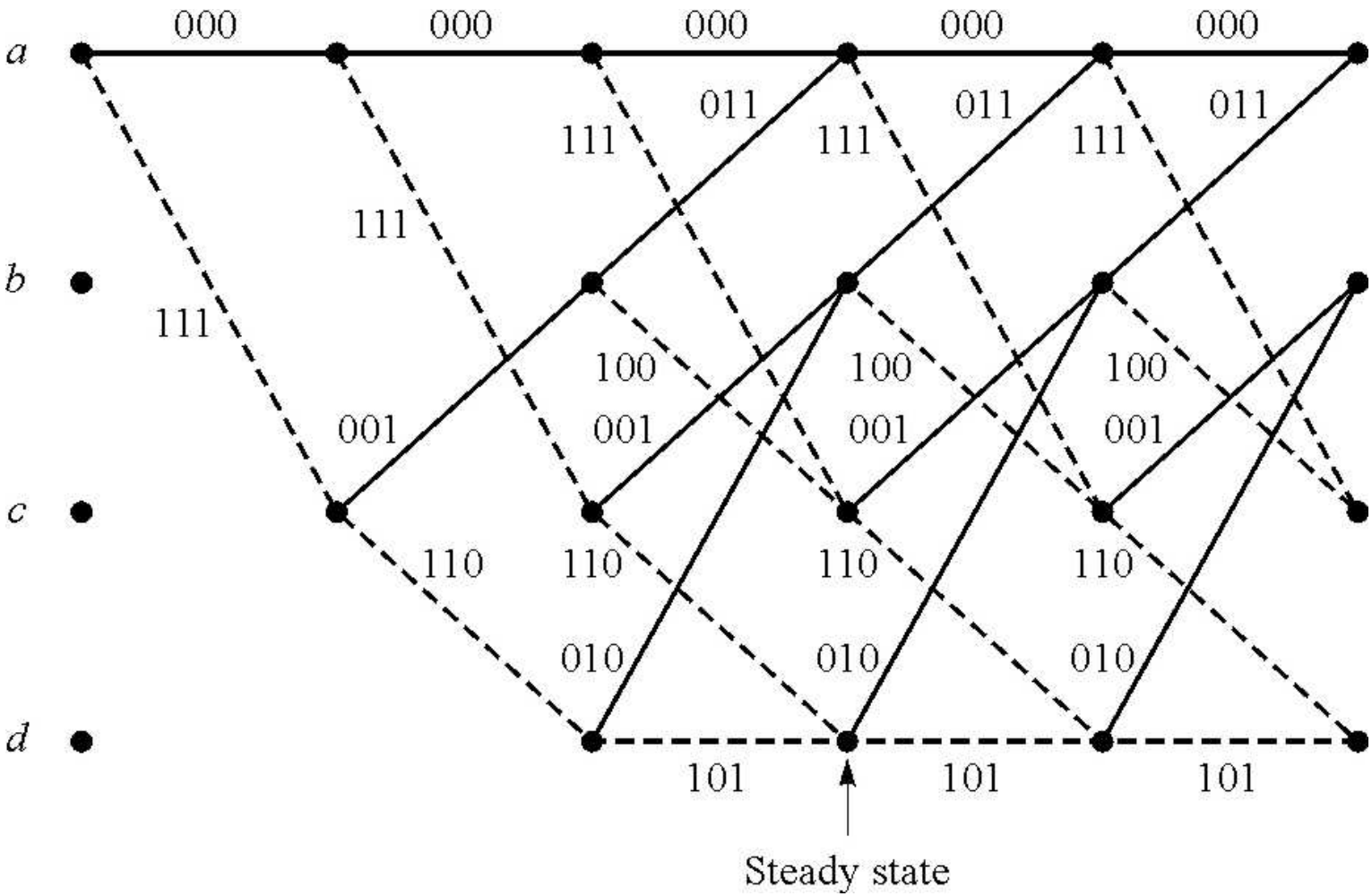
There are three alternative methods to describe a convolutional code:

1. the tree diagram
2. the trellis diagram
3. the state diagram.



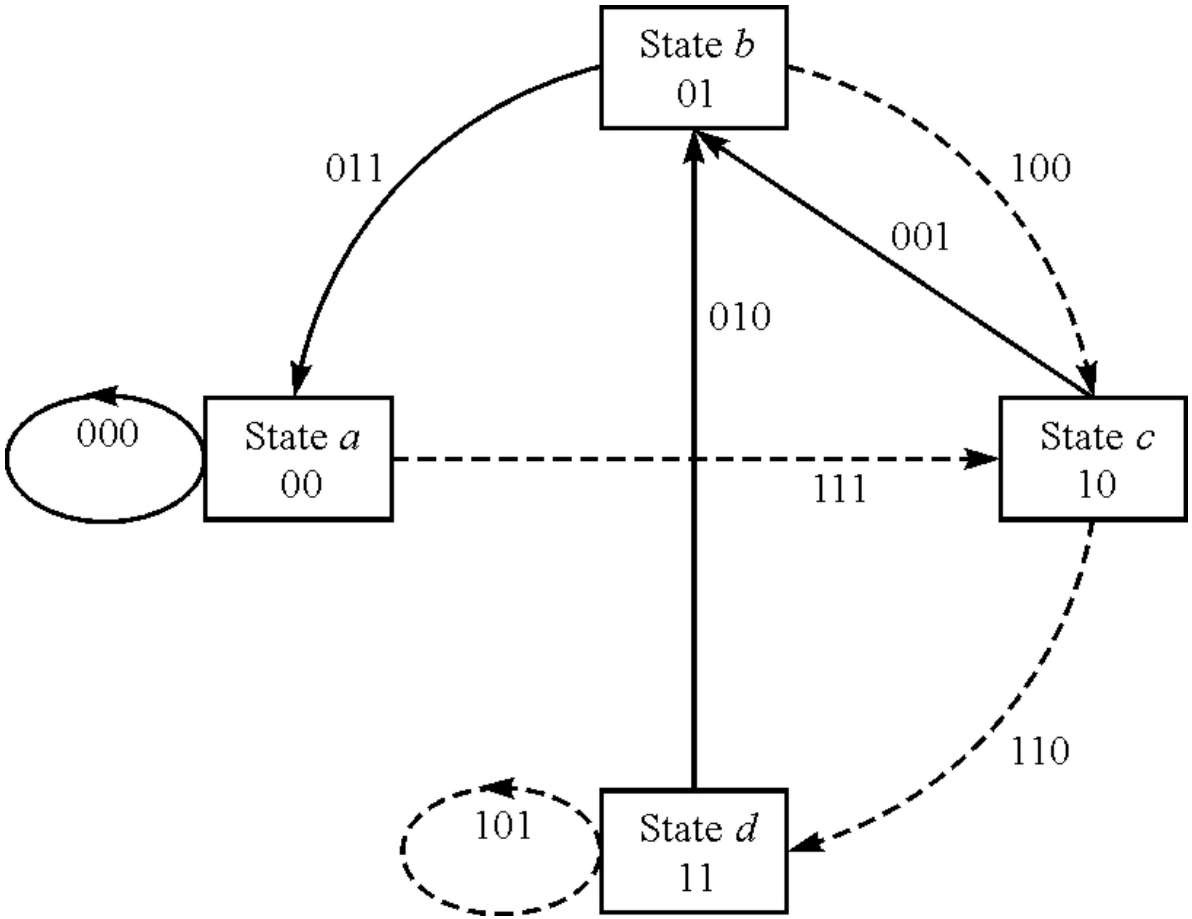
Tree diagram for a convolutional code with rate $R_c = 1/3$ and $K = 3$.

The trellis diagram



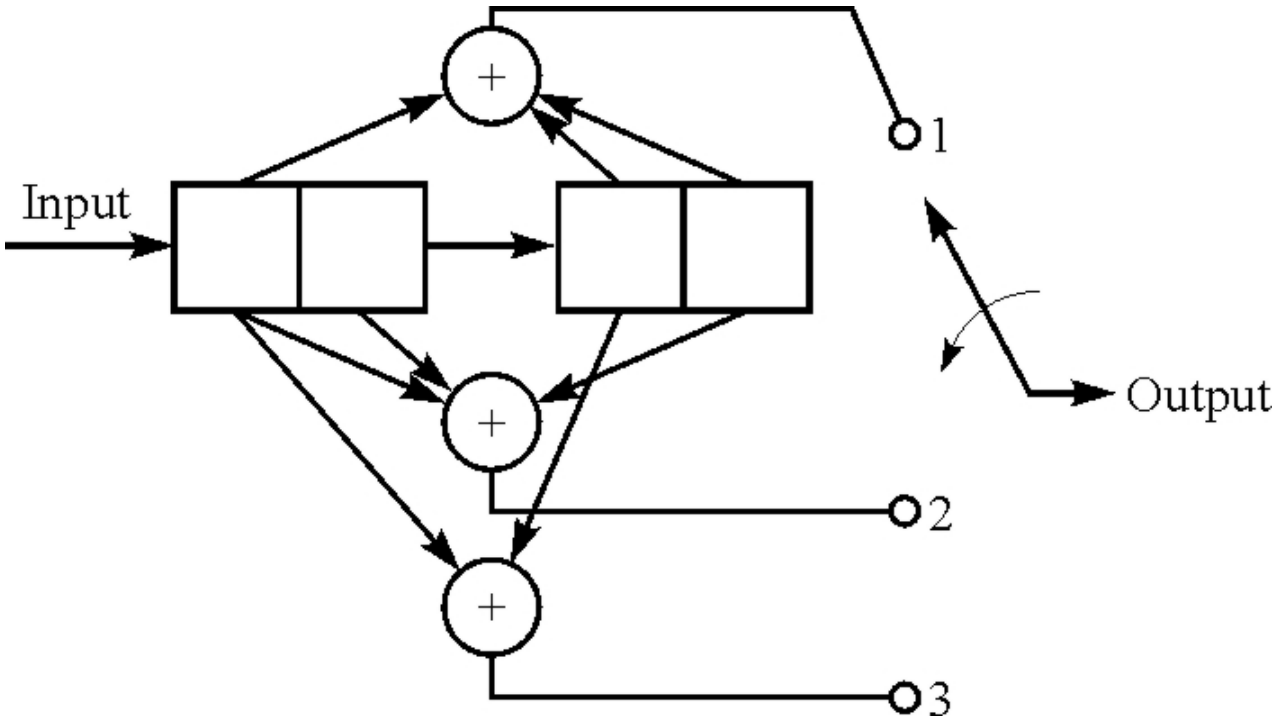
Trellis diagram for a convolutional code with rate $R_c = 1/3$ and $K = 3$.

The state diagram



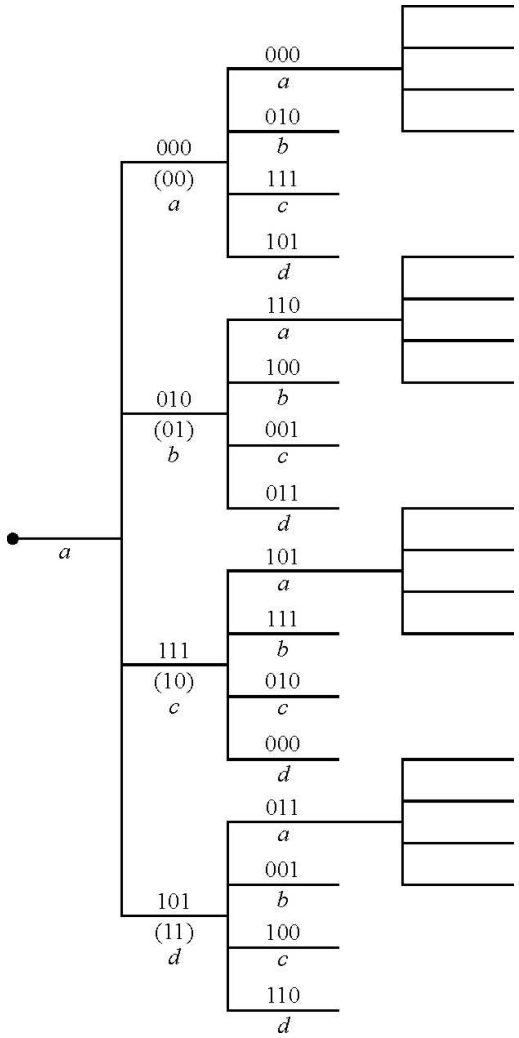
State diagram for a convolutional code with rate $R_C = 1/3$ and $K = 3$.

The aforementioned way of representing a convolutional encoder can be transferred to the case $k > 1$:



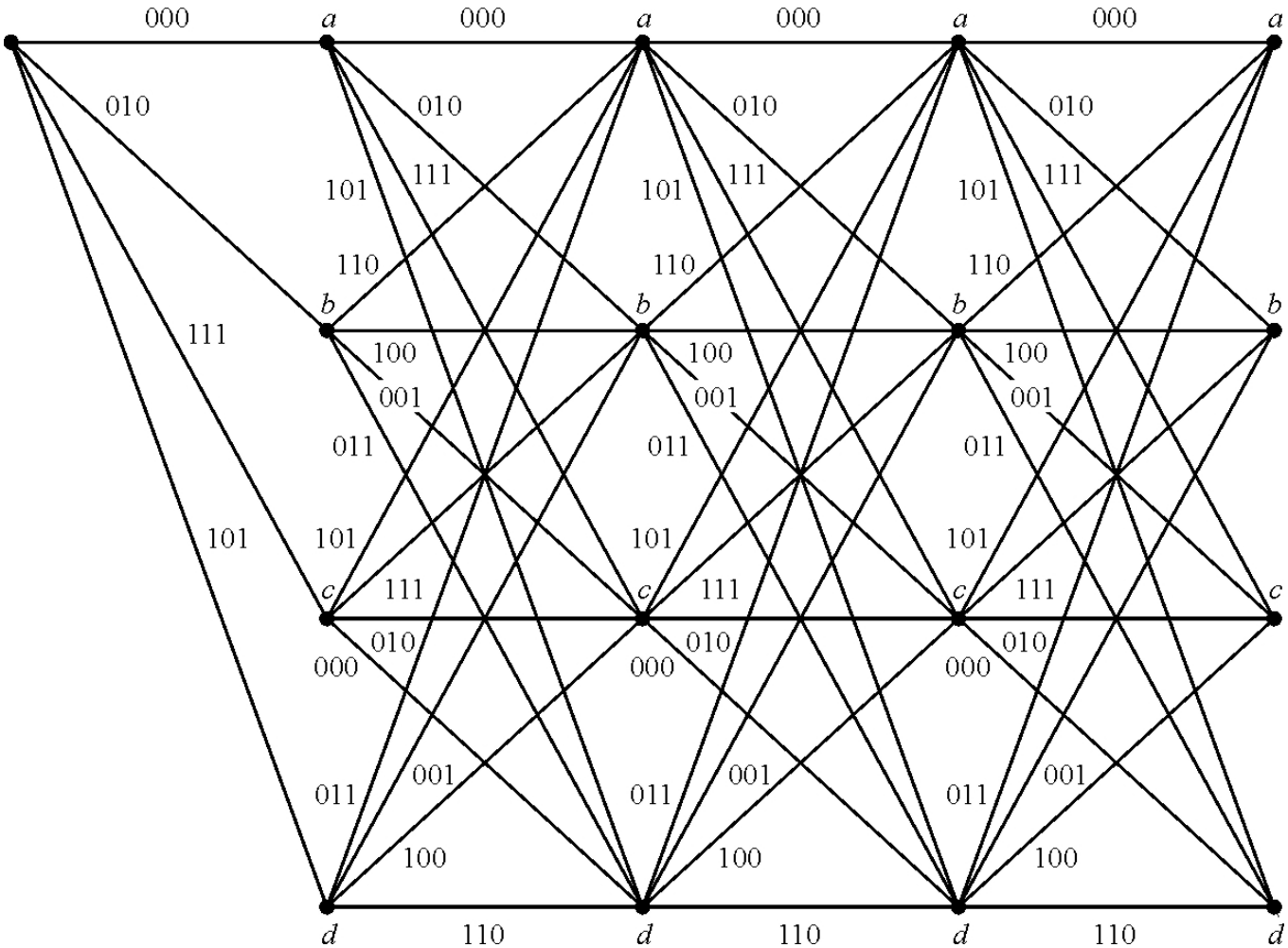
Convolutional encoder with $K = 2$, $k = 2$ and $n = 3$.

The tree diagram



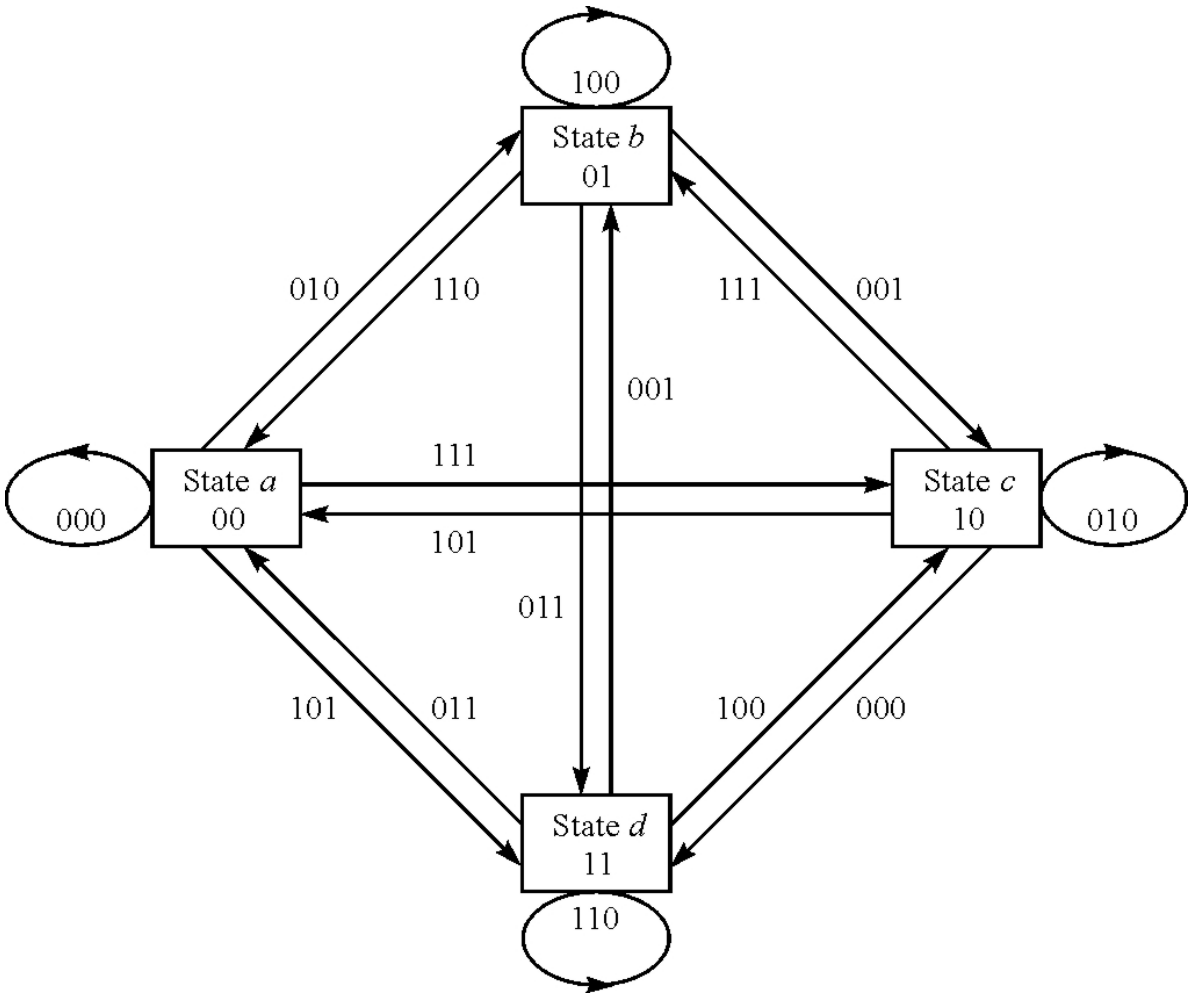
Tree diagram for a convolutional code with rate $R_C = 2/3$, $k = 2$, $n = 3$ and $K = 2$.

The trellis diagram



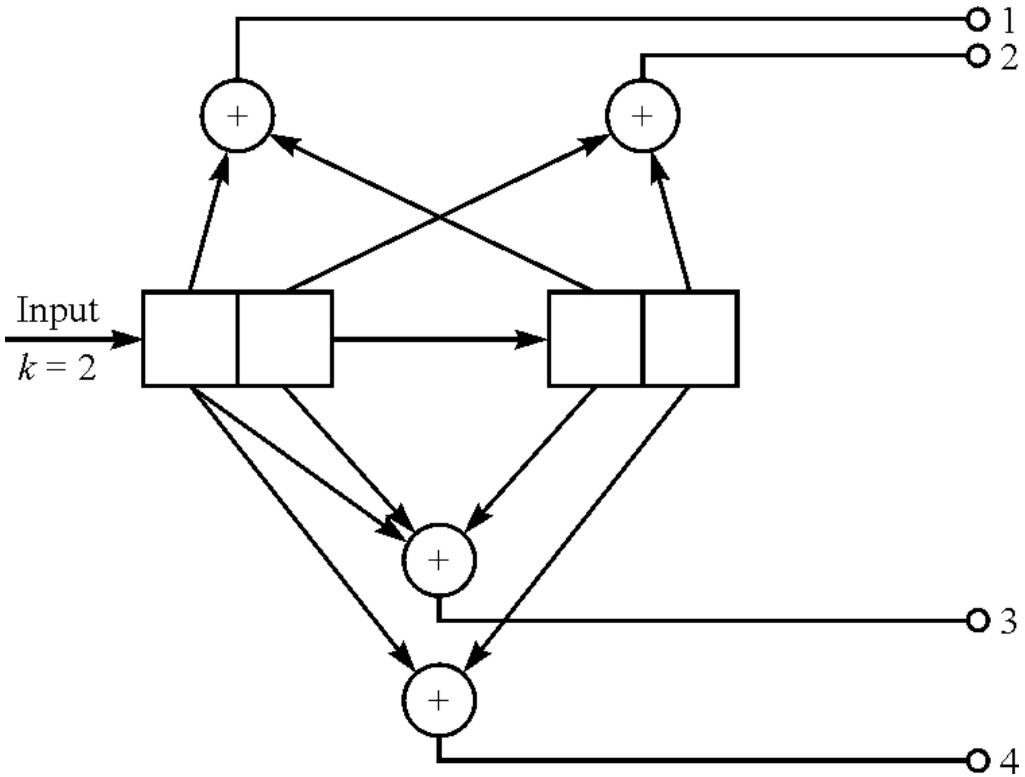
Trellis diagram for a convolutional code with rate $R_c = 2/3$, $k = 2$, $n = 3$ and $K = 2$.

The state diagram



State diagram for a convolutional code with rate $R_c = 2/3$, $k = 2$, $n = 3$ and $K = 2$.

- The three types of diagrams described above are also used to represent nonbinary convolutional codes.
- When the number of symbols in the code alphabet is $q = 2^k$, $k > 1$, the resulting nonbinary code may also be represented as an equivalent binary code.



Convolutional encoder with $K = 2$, $k = 2$ and $n = 4$.

EXAMPLE 8.2–3. Let us consider the convolutional code generated by the encoder shown in Figure 8.2–10. This code may be described as a binary convolutional code with parameters $K = 2$, $k = 2$, $n = 4$, $R_c = 1/2$, and having the generators

$$\mathbf{g}_1 = [1010], \qquad \mathbf{g}_2 = [0101], \qquad \mathbf{g}_3 = [1110], \qquad \mathbf{g}_4 = [1001]$$

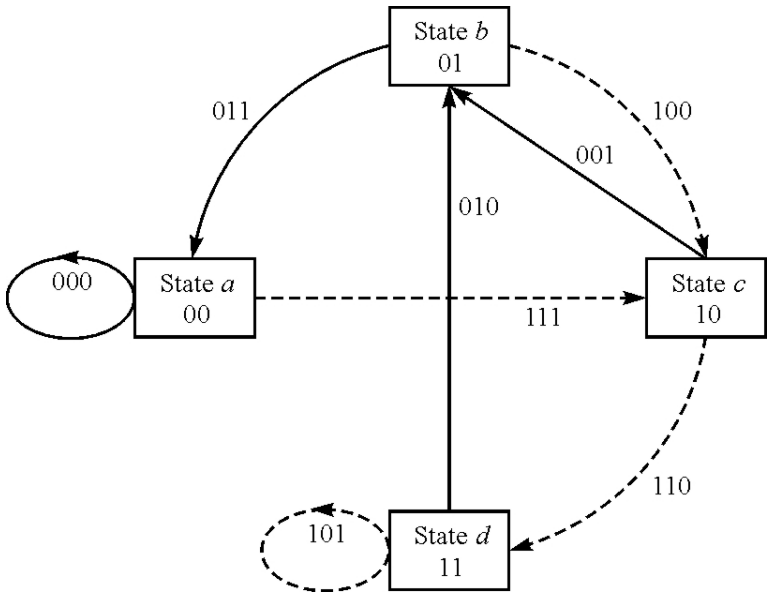
Except for the difference in rate, this code is similar in form to the rate $2/3$, $k = 2$ convolutional code considered in Example 8.2–1.

Alternatively, the code generated by the encoder in Figure 8.2–10 may be described as a nonbinary ($q = 4$) code with one quaternary symbol as an input and two quaternary symbols as an output. In fact, if the output of the encoder is treated by the modulator and demodulator as q-ary ($q = 4$) symbols that are transmitted over the channel by means of some M-ary ($M = 4$) modulation technique, the code is appropriately viewed as nonbinary.

In any case, the tree, the trellis, and the stage diagrams are independent of how we view the code. That is, this particular code is characterized by a tree with four branches emanating from each node, or a trellis with four possible states and four branches entering and leaving each state, or, equivalently, by a state diagram having the same parameters as the trellis.

The transfer function

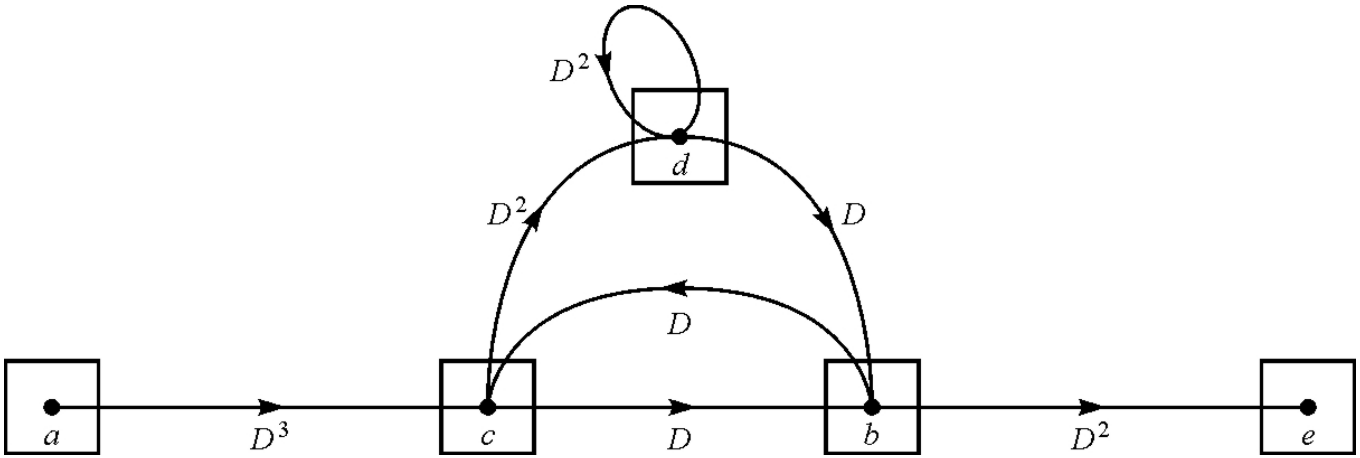
- The distance properties and the error rate performance of a convolutional code can be obtained from its state diagram.
- Similarly as in the case of a linear block code, a convolutional code is linear. Thus, the set of Hamming distances of the code sequences generated up to some stage in the tree, from the all-zero code sequence, is the same as the set of distances of the code sequences w.r.t. any other code sequence.
- Consequently, we assume without loss of generality that the all-zero code sequence is the input to the encoder.



State diagram for a convolutional code with rate $R_c = 1/3$ and $K = 3$.

The transfer function: distance properties

- First, we label the branches of the state diagram as either $D^0 = 1$, D^1, D^2 or D^3, where the exponent of D denotes the Hamming distance between the sequence of output bits corresponding to each branch and the sequence of output bits corresponding to the all-zero branch.
- The self-loop at node a can be eliminated since it contributes nothing to the distance properties of a code sequence relative to the all-zero code sequence.
- Furthermore, node a is split into two nodes, one of which represents the input and the other the output of the state diagram;



State diagram for a convolutional code with rate $R_c = 1/3$ and $K = 3$.

The transfer function: distance properties

- We use the diagram, which now consists of five nodes because node a was split into two, to write the four state equations:

$$\begin{aligned} X_c &= D^3 X_a + D X_b \\ X_b &= D X_c + D X_d \\ X_d &= D^2 X_c + D^2 X_d \\ X_e &= D^2 X_b. \end{aligned}$$

- The transfer function (for going from node a to node e, i.e. back to node a) is defined as

$$T(D) = X_e / X_a,$$

where additions, multiplications etc. of coefficients are to be taken in $\mathbb{Z}$.

Problem

By solving the state equations, show that

$$T(D) = \frac{D^6}{1-2D^2} = D^6 + 2D^8 + 4D^{10} + 8D^{12} + \cdots = \sum_{d=6}^{\infty} a_d D^d,$$

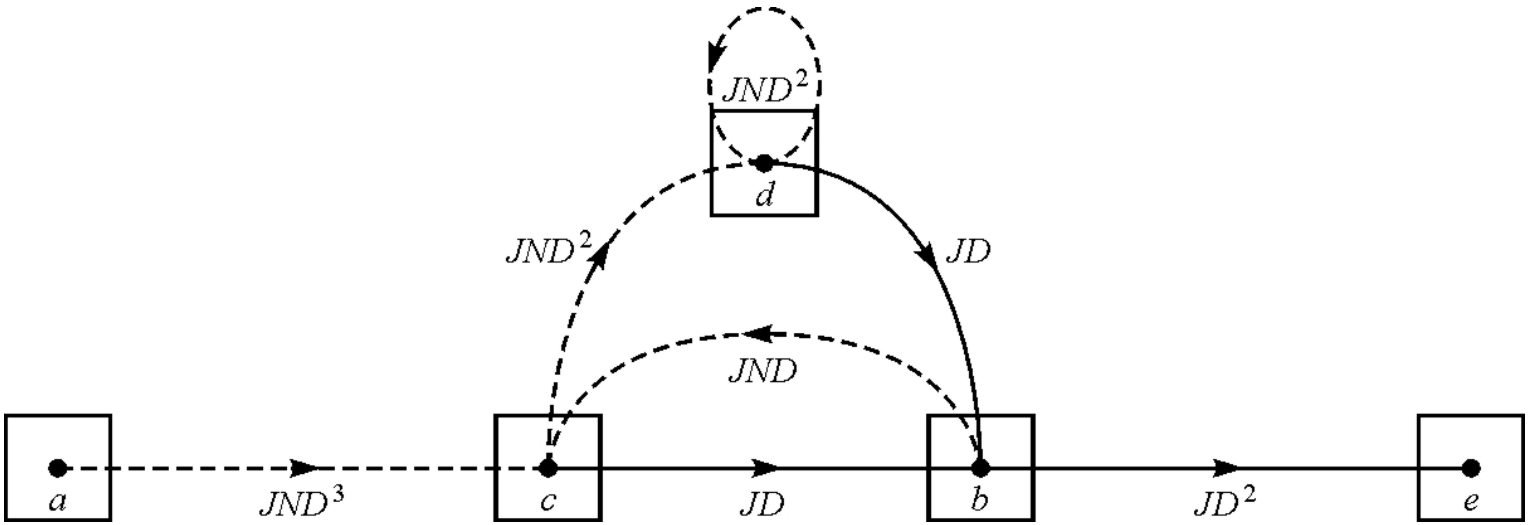
with $$a_d = \begin{cases} 2^{(d-6)/2} & \text{for even } d \\ 0 & \text{for odd } d \end{cases}.$$

The transfer function: distance properties

- The transfer function indicates that there is a single path or Hamming distance $d = 6$ from the all-zero path that merges with the all-zero path at a given node.
- From the state diagram or the trellis, it is observed that the $d = 6$ path is $acbe$, since there is no other path from node a to node e having a distance $d = 6$.
- The second term in $T(D)$ indicates that there are two paths from node a to node e having a distance $d = 8$.
- Again from the state diagram or the trellis, we observe that these paths are $acdbe$ and $acbcbe$.
- The third term in $T(D)$ indicates that there are four paths of distance $d = 10$ and so forth.
- As a result, the transfer function gives us the distance properties of the convolutional code.
- The minimum Hamming distance of the code is called the **minimum free distance** and denoted by d_{free}. In the example, $d_{\text{free}} = 6$.

The transfer function: other properties

- The transfer function can be used to provide more detailed information than just the distance of the various paths.
- Suppose we introduce a factor N into all branch transitions caused by the input bit 1. Thus, as each branch is traversed, the cumulative exponent on N increases by one onlyif that branch transition is due to an input bit 1.
- Futhermore, we introduce a factor of J into each branch of the state diagram so that the exponent of J will serve as a counting variable to indicate the number of branches in any given path from node a to node e.



State diagram for a convolutional code with rate $R_c = 1/3$ and $K = 3$.

The transfer function: other properties

Problem

Derive the four state equations and the transfer function $T(D, N, J)$ Show that we have

$$T(D, N, J) = J^3 N D^6 + J^4 N^2 D^8 + J^5 N^2 D^8 + J^5 N^3 D^{10} + 2J^6 N^3 D^{10} + J^7 N^3 D^{10} + \cdots$$

Interpretation:

- The path with distance $d = 6$ is of length 3 and of the three information bits one is a 1.
- The second and third terms indicate that of the two terms with $d = 8$, one is of length 4 and two of the five information bits in the path having length 5 are 1s.
- The exponent of the factor J indicates the length of the path that merges with the all-zero path for the first time.
- The exponent of the factor N indicates the number of 1s in the information sequence for that path.
- The exponent of the factor D indicates the distance of the sequence of encoded bits for that path from the all-zero sequence.
- If we want to hide one of the factors mentioned above, we just set the corresponding value to one, e.g. $J = 1$ in order to hide the number of bits.

The transfer function: other properties

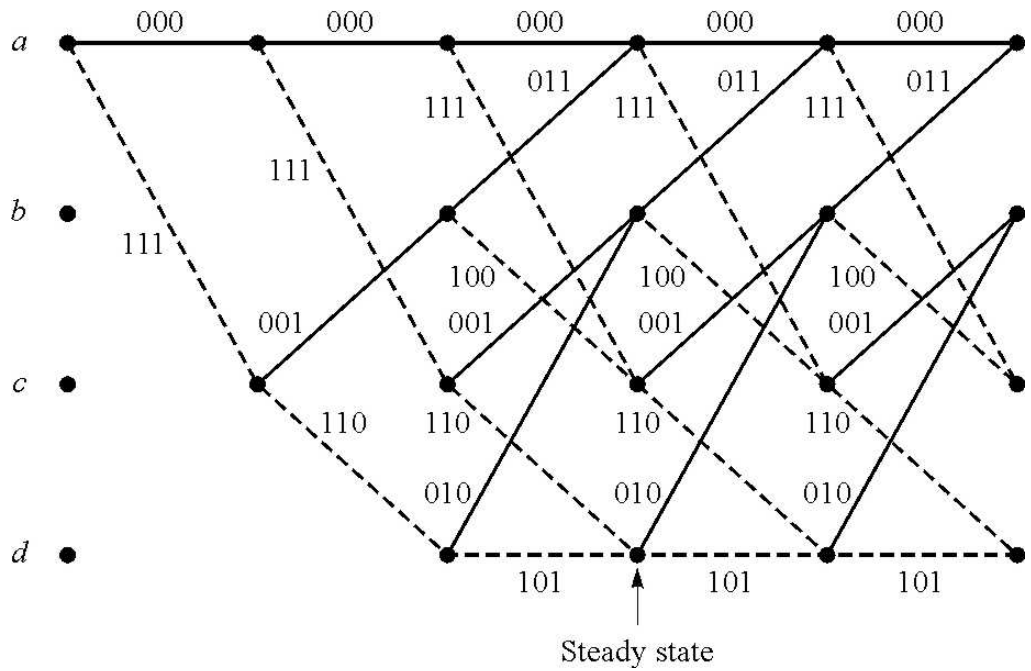
- Some convolutional codes exhibit a characteristic behavior that is called catastrophic error propagation.
- When a code that has this characteristic is used on a binary symmetric channel, it is possible for a finite number of channel errors to cause an infinite number of decoding errors.
- Such a code can be identified from its state diagram. It will contain a zero-distance path (i.e., a path with multiplier $D^0 = 1$) from some nonzero state back to the same state.
- This means that one can loop around this zero-distance path an infinite number of times without increasing the distance relative to the all-zero path.
- But, if this self-loop corresponds to the transmission of a 1, the decoder will make an infinite number of errors.
- Since such codes are easily recognized, they are easily avoided in practice.

Problem

Solve **problem 8.23** in Proakis' book (p. 544).

Optimum decoding: The Viterbi algorithm

- Optimum decoding of a convolutional code involves a search through the trellis for the most probable sequence.
- Depending on whether the detector following the demodulator performs hard or soft decisions, the corresponding metrics in the trellis search may be either a Hamming metric or a Euclidean metric, respectively.



Trellis diagram for a convolutional code with rate $R_c = 1/3$ and $K = 3$.

Optimum decoding: The Viterbi algorithm

- Consider the two paths in the trellis that begin at the initial state a and remerge at state a after three state transitions (three branches) corresponding to the two information sequences 000 and 100.
- We obtain as transmitted sequences 000 000 000 and 111 001 011, respectively.
- Below, we denote the transmitted bits by $\{c_{jm}, j = 1, 2, 3; m = 1, 2, 3\}$, where the index j indicates the jth branch and the index m the mth bit in that branch.
- Correspondingly, we define $\{r_{jm}, j = 1, 2, 3; m = 1, 2, 3\}$ as the output of the demodulator.
- If the decoder performs hard-decision decoding, the detector output for each transmitted bit is either 0 or 1.
- If the decoder performs soft-decision decoding for the case of a BPSK transmission, the input to the decoder is

$$r_{jm} = \sqrt{\varepsilon_c}(2c_{jm} - 1) + n_{jm},$$

where n_{jm} represents the additive noise and ε_c is the transmitted signal energy for each code bit.

Optimum decoding: The Viterbi algorithm

- A metric is defined for the jth branch of the ith path through the trellis as the logarithm of the joint probability of the sequence $\mathbf{R}_j = \{r_{jm}, m = 1, 2, 3\}$ conditioned on the transmitted sequence $\mathbf{C}_j^{(i)} = \left\{c_{jm}^{(i)}, m = 1, 2, 3\right\}$ for the ith path:

$$\mu_j^{(i)} = \ln P\left(\mathbf{R}_j | \mathbf{C}_j^{(i)}\right), \qquad j = 1, 2, 3, \ldots$$

- Furthermore, a metric for the ith path consisting of B branches through the trellis is defined as

$$PM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)}.$$

- The criterion for deciding between two paths through the trellis is to select the one having the larger path metric.

- This rule maximizes the probability of a correct decision or, equivalently, it minimizes the probability of error for the sequence of information bits.

Optimum decoding: The Viterbi algorithm with hard-decision

- For example, suppose that a hard-decision is performed by the detector yielding the received sequence $\{101\,000\,100\}$.
- Let $i = 0$ denote the three-branch all-zero path and $i = 1$ the second three-branch path that begins in the initial state a and remerges with the all-zero path at state a after three transitions.
- The metrics for these two paths are

$$\begin{aligned} PM^{(0)} &= 6\ln(1-p) + 3\ln(p) \\ PM^{(1)} &= 4\ln(1-p) + 5\ln(p), \end{aligned}$$

  where p denotes the probability of a bit error.
- Assuming that $p < 1/2$, we find that the metric $PM^{(0)}$ is larger than $PM^{(1)}$. This result is consistent with the observation that the all-zero path is at Hamming distance $d = 3$ from the received sequence, while the $i = 1$ path is at Hamming distance $d = 5$ from the received path. As discussed before, the Hamming distance is an equivalent metric for hard-decision decoding.

Optimum decoding: The Viterbi algorithm with soft-decision

- Suppose now that a soft-decision is performed and that the channel adds white Gaussian noise to the signal.
- The demodulator output is described by the PDF

$$p\left(r_{jm}|\, c_{jm}^{(i)}\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{\left(r_{jm} - \sqrt{\varepsilon_c}\left(2c_{jm}^{(i)} - 1\right)\right)^2}{2\sigma^2}\right\},$$

where $\sigma^2 = N_0/2$ is the variance of the additive Gaussian noise.

Problem

Show that a suitable branch metric for the jth branch of the ith path may be expressed as

$$\mu_j^{(i)} = \sum_{m=1}^{n} r_{jm}\left(2c_{jm}^{(i)} - 1\right),$$

where, in our example, $n = 3$.

- Thus, the correlation metrics for the two paths under consideration are

$$CM^{(0)} = \sum_{j=1}^{3}\sum_{m=1}^{3} r_{jm}\left(2c_{jm}^{(0)} - 1\right), \qquad CM^{(1)} = \sum_{j=1}^{3}\sum_{m=1}^{3} r_{jm}\left(2c_{jm}^{(1)} - 1\right).$$

Optimum decoding: The Viterbi algorithm

Viterbi algorithm:

- Consider the two paths described above which merge at state a after three transitions.
- Note that any particular path through the trellis that stems from this node will add identical terms to the path metrics $CM^{(0)}$ and $CM^{(1)}$. Consequently, if $CM^{(0)} > CM^{(1)}$ at the merged node a after three transitions, $CM^{(0)}$ will continue to be larger than $CM^{(1)}$ for any path that stems from node a.
- This means that the path corresponding to $CM^{(1)}$ can be discarded from further consideration. The path corresponding to the metric $CM^{(0)}$ is the survivor.
- Similarly, one of the two paths that merge at state b can be eliminated on the basis of the two corresponding metrics. This procedure is repeated at state c and state d.
- As a result, after the first three transitions, there are four surviving paths, one terminating at each state, and a corresponding metric for each survivor.
- This procedure is repeated at each stage of the trellis as new signals are received in subsequent time intervals.
- It was not noticed until several years after Viterbi introduced this decoding technique that the Viterbi algorithm can be viewed as the application of Bellman's dynamic programming method to the problem of decoding a trellis code.

Optimum decoding: The Viterbi algorithm

Computational complexity of the Viterbi algorithm:

- In general, when a binary convolutional code with $k = 1$ and constraint length K is decoded by means of the Viterbi algorithm, there are 2^{K-1} states.
- Hence, there are 2^{K-1} surviving paths at each stage and 2^{K-1} metrics, one for each surviving path.
- If, instead of $k = 1$, $k > 1$ bits are shifted at a time into an encoder that consists of K k-bit shift-register stages, the trellis has $2^{k(K-1)}$ states. As a result, the decoding of such a code requires keeping track of $2^{k(K-1)}$ surviving paths and $2^{k(K-1)}$ metrics.
- At each stage of the trellis, there are 2^k paths that merge at each node. Only one survives, namely the most-probable (i.e. minimum-distance) path.
- Thus, the number of computations in decoding performed at each stage increases exponentially with k and K.
- The exponential increase in computational burden limits the use of the Viterbi algorithm to relatively small values of k and K.

Optimum decoding: The Viterbi algorithm

Modifications of the Viterbi algorithm:

- The decoding delay in decoding a long information sequence that has been convolutionally encoded is usually too long for most practical applications.
- Moreover, as discussed before, the memory required to store the entire length of surviving sequences is large and expensive.
- A solution to this problem is to modify the Viterbi algorithm in a way which results in a fixed decoding delay without significantly affecting the optimal performance of the algorithm in terms of bit-error probability.
- The modification is to retain at any given time t only the most recent δ decoded information bits (symbols) in each surviving sequence.
- As each new information bit (symbol) is received, a final decision is made on the bit (symbol) received δ branches back in the trellis, by comparing the metrics in the surviving sequences and deciding in favor of the bit in the sequence having the largest metric.
- If δ is chosen sufficiently large, all surviving sequences will contain the identical decoded bit (symbol) δ branches back in time.
- It has been found by computer simulations that a delay $\delta \geq 5K$ results in a negligible degradation in the performance relative to the optimum Viterbi algorithm.

Optimum decoding: The Viterbi algorithm

Upper bounds on the bit-error probability (without proof):

- For soft-decision decoding and $k = 1$, we obtain assuming a BPSK transmission over an AWGN channel that the bit-error probability is upper-bounded by

$$P_b < \left.\frac{\mathrm{d}T(D,N)}{\mathrm{d}N}\right|_{N=1, D=\exp\left(-\gamma_b R_c\right)},$$

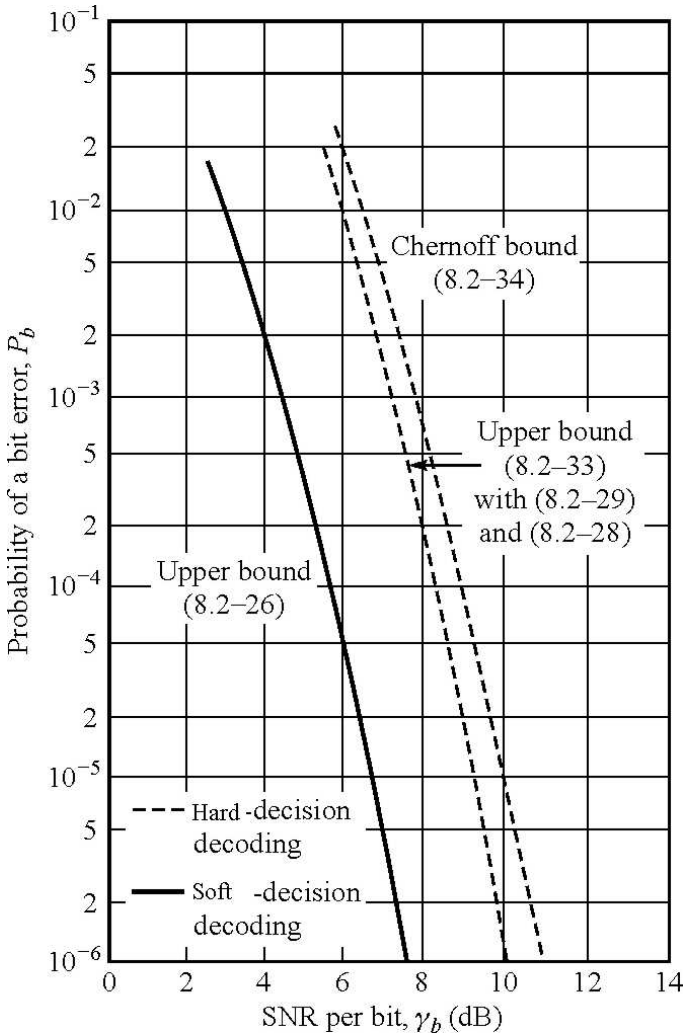
  where $T(D)$, γ_b and R_c denote the transfer function of the code, the SNR per bit and the code rate, respectively.

- For hard-decision decoding and $k = 1$, we obtain assuming a transmission over a BSC with a probability of error p that the bit-error probability is upper bounded by

$$P_b < \left.\frac{\mathrm{d}T(D,N)}{\mathrm{d}N}\right|_{N=1, D=2\sqrt{p(1-p)}},$$

  where $T(D,N)$ denotes the transfer function of the code.

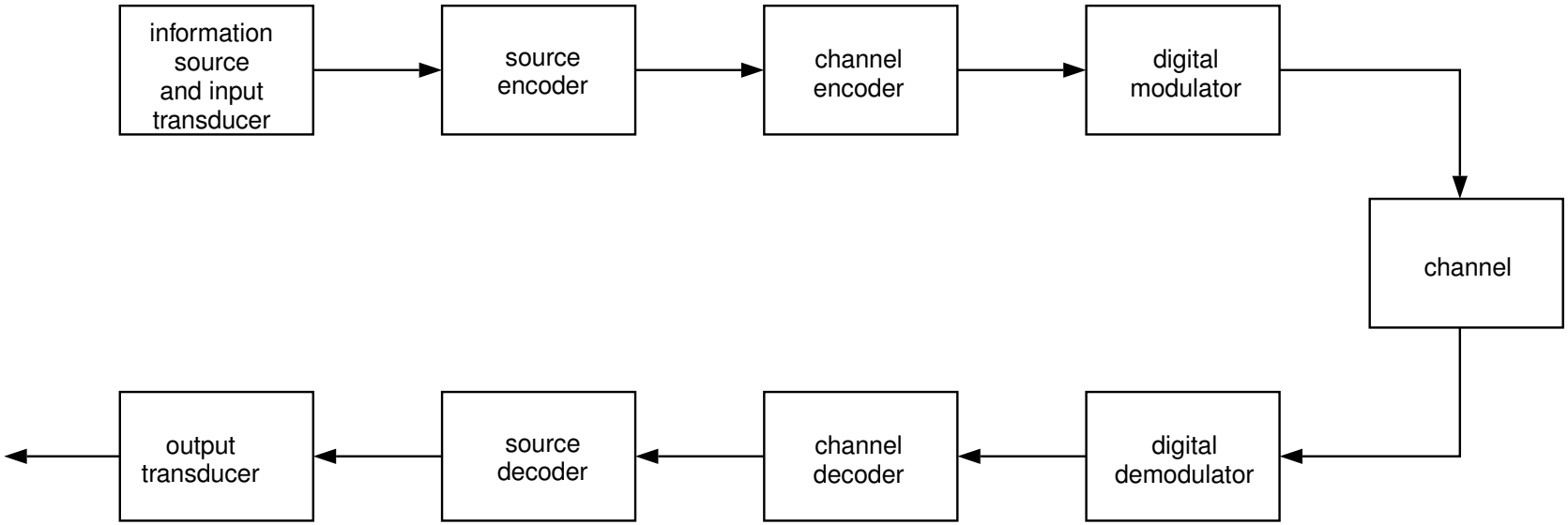
Optimum decoding: The Viterbi algorithm



Comparison of soft-decision and hard-decision decoding of a convolutional code with rate $K = 3$, $k = 1$ and $n = 3$.

Table of Contents

Source Coding



Generic model of a digital communication system.

- Until now, we have discussed the transmission of information, i.e. a representation of a random sequence with a given entropy by bit patterns, over a channel to a receiver.
- Here, we want to investigate how we should represent the information, i.e., how to encode the output of an information source with entropy $H(X)$ where X is the output of the source.
- A measure of efficiency of a source encoding scheme can be obtained by comparing the average number of binary digits per output letter from the source of entropy $H(X)$.

- The encoding of a discrete source having a finite alphabet size may appear, at first glance, to be a relatively simple problem. This is true, however, only when the source is memoryless, i.e., when successive symbols from the source are statistically independent and each symbol is encoded separately.
- The discrete memoryless source (DMS) is by far the simplest model that can be devised for a physical source. Only few sources, however, closely fit this mathematical model.
- For example, successive output letters from a machine printing English text are expected to be statistically dependent (you rarely say *yzxutyuwerg zzzzzzurwn fsdfiut kweoriu*, but you can say *information is sometimes hard to understand*).
- On the other hand, if the machine output is a computer program coded in FORTRAN, the sequence of output letters is expected to exhibit a much smaller dependence.
- In any case, we shall demonstrate that it is always more efficient to encode blocks of symbols instead of encoding each symbol separately.
- Intuitively, we would expect that by making the block size sufficiently large, the average number of binary digits per output letter from the source can be made arbitrarily close to the entropy of the source.

Coding for Discrete Memoryless Sources

Fixed-length code words

- Suppose that a DMS produces an output letter or symbol every τ_s seconds. Each symbol is selected from a finite alphabet of symbols x_i, $i = 1, 2, \ldots, L$, occuring with probabilities $P(x_i)$, $i = 1, 2, \ldots, L$. The entropy of a DMS in bits per source symbol is

$$H(X) = -\sum_{i=1}^{L} P(x_i)\mathrm{ld}(P(x_i)) \leq \mathrm{ld}(L),$$

  where equality holds when the symbols are equally probable.
- The average number of bits per source symbol is $H(X)$ and the source rate in bit/s is defined as $H(X)/\tau_s$.
- First, we consider a block encoding scheme that assigns a unique set of R binary digits to each symbol. Since there are L possible symbols, the number of binary digits per symbol required for unique encoding when L is a power of 2 is $R = \mathrm{ld}(L)$, and when L is not a power of 2, it is $R = \lfloor \mathrm{ld}(L) \rfloor + 1$, where again $\lfloor x \rfloor$ denotes the largest integer less than x.
- Since $H(X) \leq \mathrm{ld}(L)$, it follows that the code rate satisfies $R \geq H(X)$.
- The efficiency of the encoding for the DMS is defined as the ratio $H(X)/R$.

Fixed-length code words

- We observe that when L is a power of 2 and the source letters are equally probable, $R = H(X)$. Hence, a fixed-length code of R bits per symbol attains 100 percent efficiency.
- However, if L is not a power of 2 but the source symbols are still equally probable, R differs from $H(X)$ by at most one bit per symbol.
- When $\mathrm{ld}(L) \gg 1$, the efficiency of this encoding scheme is high.
- On the other hand, when L is small, the efficiency of the fixed-length code can be increased by encoding a sequence of J symbols at a time. To accomplish the desired encoding, we require L^J unique code words. By using sequences of N binary digits, we an accomodate 2^N possible code words.
- We must select N according to

$$N \geq J \,\mathrm{ld}(L).$$

- Hence, the minimum integer value of N required is

$$N = \lfloor J \,\mathrm{ld}(L) \rfloor + 1.$$

- Now the average number of bits per source symbol is $N/J = R$, thus, the inefficiency has been reduced by approximately a factor of $1/J$ relative to the symbol-by-symbol encoding described above. Making J large, the efficiency of the encoding scheme, measured by the ratio $JH(X)/N$, can be made as close to unity as desired.

Fixed-length code words

- The encoding schemes described above introduce no distortion since the encoding of source symbols (or blocks of symbols) into code words is unique (noiseless coding).
- Now suppose we reduce the code rate R by relaxing the condition that the encoding process be unique. For example, only a fraction of the L^J blocks of symbols is encoded uniquely.
- To be specific, select the $2^N - 1$ most probable J-symbol blocks and encode each of them uniquely, while the remaining $L^J - \left(2^N - 1\right)$ J-symbol blocks are represented by the single remaining code word. This procedure results in a decoding failure or (distortion) probability of error of every time a low probability block is mapped into this single code word.
- Let p_e denote this probability of error. Based on the block encoding procedure, Shannon proved the following source coding theorem.

Fixed-length code words

Source coding theorem I: Let X be the ensemble of letters from a DMS with finite entropy $H(X)$. Blocks of J symbols from the source are encoded into code words of length N from a binary alphabet. For any $\varepsilon > 0$, the probability p_e of a block decoding failure can be made arbitrarily small if

$$R \equiv \frac{N}{J} \geq H(X) + \varepsilon.$$

and J is sufficiently large. Conversely, if

$$R \leq H(X) + \varepsilon,$$

then p_e becomes arbitrarily close to 1 as J is made sufficiently large.

- The theorem says that the average number of bits per symbol required to encode the output of a DMS with arbitrarily small probability of decoding failure is lower bounded by the source entropy $H(X)$.
- On the other hand, if $R < H(X)$, the decoding failure rate approaches 100 percent as J is arbitrarily increased.

Variable-length code words

- Clearly, when the code words are not equally probable, one should encode letters occuring more frequently by short words, and rare letters by long words (principle of entropy coding).
- Suppose that a DMS with output letters a_1, a_2, a_3 and a_4 and corresponding probabilities $P(a_1) = 1/2$, $P(a_2) = 1/4$ and $P(a_3) = P(a_4) = 1/8$ is encoded as shown in the following table:

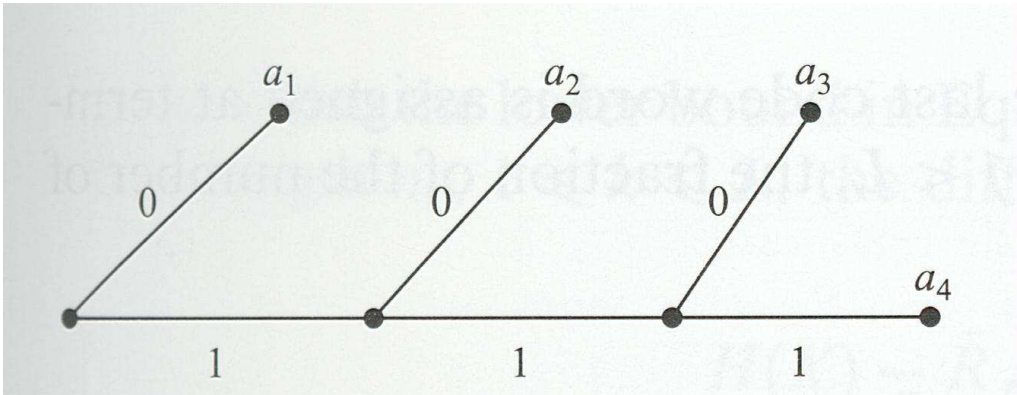
TABLE 3.3–1
Variable-length codes

Letter	$P(a_k)$	Code I	Code II	Code III
a_1	$\frac{1}{2}$	1	0	0
a_2	$\frac{1}{4}$	00	10	01
a_3	$\frac{1}{8}$	01	110	011
a_4	$\frac{1}{8}$	10	111	111

Variable-length code words

- **Code I** is a variable-length code that has a basic flaw which can be observed if we are given the sequence 001001.... Clearly, the first symbol corresponding to 00 is a_2. However, the next four bits are ambiguous (not uniquely decodable) (decode as either a_4a_3 or $a_1a_2a_1$). Perhaps, the ambiguity can be resolved by waiting for additional bits, but such a decoding delay is highly undesirable.
- We shall only consider codes that are decodable instantaneously, i.e. without any decoding delay.

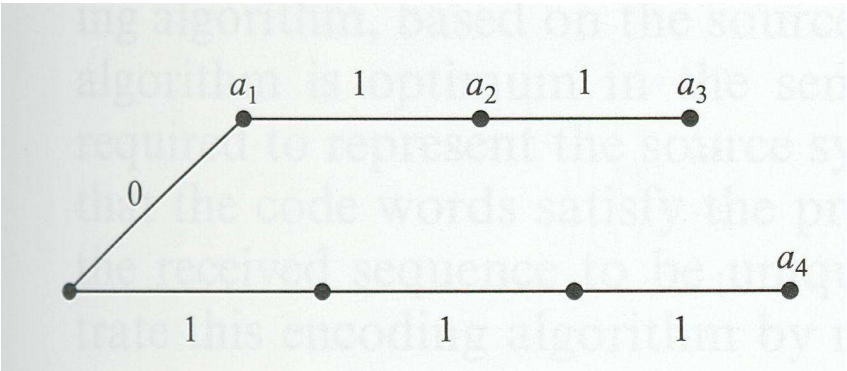
- **Code II** is uniquely decodable and instantaneously decodable.
- It is convenient to represent the code words in this code graphically as terminal nodes of a tree.



Code tree for code II.

Variable-length code words

- The digit 0 indicates the end of a code word for the first three code words. This characteristic plus the fact that no code word is longer than three binary digits makes this code instantaneously decodable.
- Note that no code word in this code is a prefix of any other code word.
- In general, the prefix condition requires that for a given code word $\mathbf{C}_k$ of length k having elements $(b_1, b_2, \ldots, b_k)$, there is no other code word of length $\ell < k$ with elements $(b_1, b_2, \ldots, b_\ell)$ for $1 \leq \ell \leq k - \ell$.
- In other words: there is no code word of length $\ell < k$ that is identical to the first ℓ binary digits of another code word of length $k > \ell$.
- This property makes the code word instantaneously decodable.
- **Code III** has the following tree structure:



Code tree for code III.

Variable-length code words

- Obviously, code III is uniquely decodable, but not instantaneously decodable. Clearly, the code does not satisfy the prefix condition.
- As a result, our design objective is to devise a systematic procedure for constructing uniquely decodable variable-length codes that are efficient in the sense that the average number of bits per source letter defined as the quantity

$$\bar{R} = \sum_{k=1}^{L} n_k P(a_k)$$

  is minimized where n_k is the number of bits for source letter k and $P(a_k)$ is the probability of a_k.
- Conditions for the existence of a code satisfying the prefix condition are given in the

Kraft inequality: A necessary and sufficient condition for the existence of a binary code with code words having lengths $n_1 \leq n_2 \leq \cdots \leq n_L$ that satisfy the prefix condition is $\sum_{k=1}^{L} 2^{-n_k} \leq 1$.

For a proof, cf. Proakis' book p. 93/94.

Variable-length code words

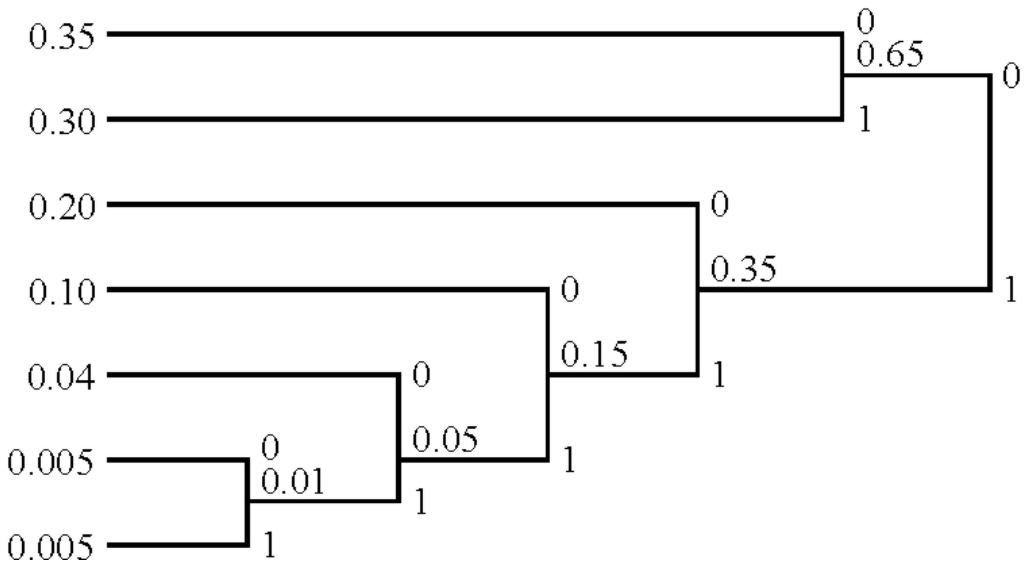
Source coding theorem II: Let X be the ensemble of letters from a DMS with finite entropy $H(X)$ and output letters x_k, $1 \leq k \leq L$, with corresponding probabilities of occurence p_k, $1 \leq k \leq L$. It is possible to construct a code that satisfies the prefix condition and has an average length $\bar{R}$ that satisfies the inequalities

$$H(X) \leq \bar{R} \leq H(X) + 1.$$

- We have now established that variable-length codes that satisfy the prefix condition are efficient source codes for any DMS with source symbols that are not equally probable.
- Next we describe an algorithm for constructing such codes.

Huffman coding

- Huffman (1952) devised a variable-length encoding algorithm based on the source letter probabilities $P(x_i)$, $i = 1, 2, \ldots, L$. This algorithms is optimum in the sense that the average number of binary digits required to represent the source symbols is a minimum subject to the constraint that the code words satisfy the prefix condition.
- To recap, the prefix condition ensures that the received sequence to be uniquely and instantaneously decodable.
- Consider two examples.



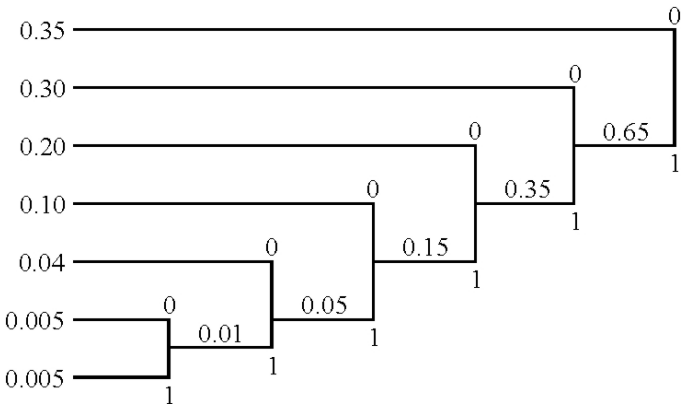
Example of variable-length source encoding for a DMS with $H(X) = 2.11$ and $\bar{R} = 2.21$.

Huffman coding

EXAMPLE 3.3–1. Consider a DMS with seven possible symbols $x_1, x_2, \ldots, x_7$ having the probabilities of occurrence illustrated in Figure 3.3–4. We have ordered the source symbols in decreasing order of the probabilities, i.e., $P(x_1) > P(x_2) > \cdots > P(x_7)$. We begin the encoding process with the two least probable symbols x_6 and x_7. These two symbols are tied together as shown in Figure 3.3–4, with the upper branch assigned a 0 and the lower branch assigned a 1. The probabilities of these two branches are added together at the node where the two branches meet to yield the probability 0.01. Now we have the source symbols $x_1, \ldots, x_5$ plus a new symbol, say x_6', obtained by combining x_6 and x_7. The next step is to join the two least probable symbols from the set $x_1, x_2, x_3, x_4, x_5, x_6'$. These are x_5 and x_6', which have a combined probability of 0.05. The branch from x_5 is assigned a 0 and the branch from x_6' is assigned a 1. This procedure continues until we exhaust the set of possible source letters. The result is a code tree with branches that contain the desired code words. The code words are obtained by beginning at the rightmost node in the tree and proceeding to the left. The resulting code words are listed in Figure 3.3–4. The average number of binary digits per symbol for this code is $\bar{R} = 2.21$ bits per symbol. The entropy of the source is 2.11 bits per symbol.

Huffman coding

- Obviously, the code word assignment in Huffman coding is not unique. For example, at the next to the last step in the encoding scheme, we have a tie between x_1 and x'_3, since these symbols are equally likely. In addition, zeros and ones can be changed arbitrarily.
- Above, we have chosen to pair x_1 with x_2. Alternatively, we can pair x_2 with x'_3.



Letter	Code
x_1	0
x_2	10
x_3	110
x_4	1110
x_5	11110
x_6	111110
x_7	111111

$\bar{R} = 2.21$

Huffman coding

- The variable-length Huffman coding algorithm generates a prefix code having a $\bar{R}$ that satisfies $H(X) \leq \bar{R} < H(X) + 1$.
- However, instead of encoding on a symbo-by-symbol basis, a more efficient scheme is to encode blocks of J symbols at a time. In such a case, the bounds in the source coding theorem II become

$$JH(X) \leq \bar{R}_J < JH(X) + 1,$$

since the entropy of a J-symbol block from a DMS is $JH(X)$ and $\bar{R}_J$ is the average number of bits per J-symbol blocks.

- Upon division, we obtain

$$H(X) \leq \frac{\bar{R}_J}{J} < H(X) + \frac{1}{J},$$

where $\bar{R} = \bar{R}_J/J$ is the average number of bits per source symbol. Hence, $\bar{R}$ can be made as close to $H(X)$ as desired by selecting J sufficiently large.

Huffman coding

EXAMPLE 3.3–3. The output of a DMS consists of *letters* x_1, x_2, and x_3 with probabilities 0.45, 0.35, and 0.20, respectively. The entropy of this source is $H(X) = 1.518$ bits per symbol. The Huffman code for this source, given in Table 3.3–2, requires $\bar{R}_1 = 1.55$ bits per symbol and results in an efficiency of 97.9 percent. If pairs of symbols are encoded by means of the Huffman algorithm, the resulting code is as given in Table 3.3–3. The entropy of the source output for pairs of letters is $2H(X) = 3.036$ bits per symbol pair. On the other hand, the Huffman code requires $\bar{R}_2 = 3.0675$ bits per symbol pair. Thus, the efficiency of the encoding increases to $2H(X)/\bar{R}_2 = 0.990$ or, equivalently, to 99.0 percent.

TABLE 3.3–2
Huffman code for Example 3.3–3

Letter	Probability	Self-information	Code
x_1	0.45	1.156	1
x_2	0.35	1.520	00
x_3	0.20	2.330	01

$H(X) = 1.518$ bits/letter
$\bar{R}_1 = 1.55$ bits/letter
Efficiency $= 97.9\%$

Huffman coding

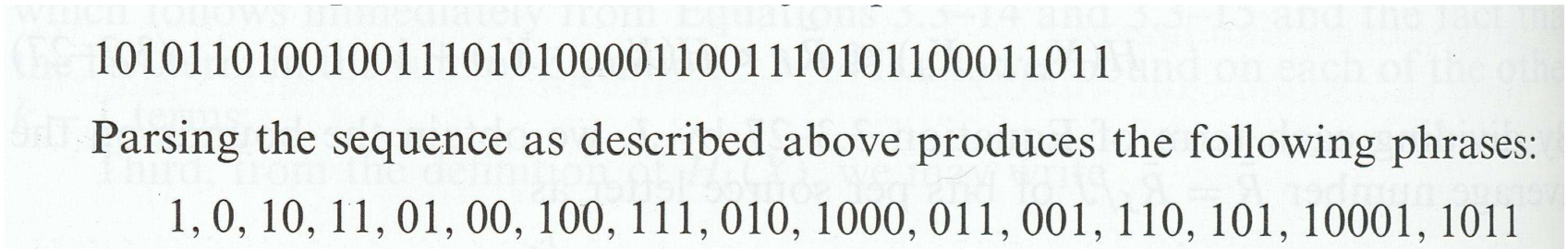
TABLE 3.3–3
Huffman code for encoding pairs of letters

Letter pair	Probability	Self-information	Code
x_1x_1	0.2025	2.312	10
x_1x_2	0.1575	2.676	001
x_2x_1	0.1575	2.676	010
x_2x_2	0.1225	3.039	011
x_1x_3	0.09	3.486	111
x_3x_1	0.09	3.486	0000
x_2x_3	0.07	3.850	0001
x_3x_2	0.07	3.850	1100
x_3x_3	0.04	4.660	1101

$2H(X) = 3.036$ bits/letter pair
$\bar{R}_2 = 3.0675$ bits/letter pair
$\frac{1}{2}\bar{R}_2 = 1.534$ bits/letter
Efficiency $= 99.0\%$

The Lempel-Ziv Algorithm

- In order to encode data using the Huffman code, we need
  - the probabilities of occurrence of all source letters in the case of a DMS
  - the joint probabilities of blocks of length $n \geq 2$ in the case of a discrete source with memory.
- These statistics, however, are rarely known in a practical system. They can be estimated, but this takes many data and a high complexity. Therefore, the Huffman coding is often impractical to be implemented.



1010110100100111010100001100111010110001101 1

Parsing the sequence as described above produces the following phrases:

1, 0, 10, 11, 01, 00, 100, 111, 010, 1000, 011, 001, 110, 101, 10001, 1011

The Lempel-Ziv Algorithm

TABLE 3.3–4
Dictionary for Lempel-Ziv algorithm

	Dictionary location	Dictionary contents	Code word
1	0001	1	00001
2	0010	0	00000
3	0011	10	00010
4	0100	11	00011
5	0101	01	00101
6	0110	00	00100
7	0111	100	00110
8	1000	111	01001
9	1001	010	01010
10	1010	1000	01110
11	1011	011	01011
12	1100	001	01101
13	1101	110	01000
14	1110	101	00111
15	1111	10001	10101
16		1011	11101

The Lempel-Ziv Algorithm

- The dictionary locations are numbered consecutively, starting with 1 and counting up to 16.
- The code words are determined by listing the dictionary location (in binary form) of the previous phrase that matched the new phrase in all but the last location.
- Then, the new output letter is appended to the dictionary location of the previous phrase.
- Initially, the location 0000 is used to encode a phrase that has not appeared previously.
- For the example, we have 44 source bits encoded in 16 code words of 5 bits each, resulting in 80 coded bits; inefficiency due to very short sequence.
- Code length might overflow and special measures are to be taken in order to avoid that (e.g. remove phrases from the dictionaries that are not useful and substitute new phrases in their place).
- The Lempel-Ziv algorithm is widely used in the compression of computer files. The **compress** and **uncompress** utilities under the UNIX operating system and numerous algorithms under the MS-DOS operating system are implementations of various versions of the algorithm.

Coding for Analog Sources

- Suppose an analog source emits a message waveform $x(t)$ that is a sample function of a stochastic process $X(t)$.
- When $X(t)$ is a band-limited stationary stochastic process, the sampling theorem allows us to represent $X(t)$ by a sequence of uniform samples taken at the Nyquist rate.
- Then, the continuous-valued samples are quantized in amplitude and encoded.
- A simple encoding represents the L discrete discrete amplitude levels by a sequence R of binary digits, where $R = \mathrm{ld}(L)$ if L is a power of 2 and $R = \lfloor \mathrm{ld}(L) \rfloor + 1$ when L is not a power of 2.
- If the levels are not equally probable and the probabilities of the output levels are known, we may use entropy coding (Huffman coding) to improve the efficiency of the encoding process.
- Quantization of the amplitudes of the sampled signals results in data compression, but it also introduces some distortion of the waveform or a loss of signal fidelity. Below, we consider the minimization of a suitably chosen distortion measure.

Rate-distortion function

- Distortion characterizes the difference between the actual source sampels $\{x_k\}$ and the corresponding quantized values $\tilde{x}_k$. The distortion is represented by the expression $d(x_k, \tilde{x}_k)$.
- A commonly used distortion measure is the squared-error distortion

$$d(x_k, \tilde{x}_k) = (x_k - \tilde{x}_k)^2.$$

- Other distortion measures are

$$d(x_k, \tilde{x}_k) = |x_k - \tilde{x}_k|^p \,,$$

  where p is a positive integer. The case $p = 2$ is often mathematically tractable and therefore chosen subsequently.
- If $d(x_k, \tilde{x}_k)$ is the distortion measure per letter, the distortion between a sequence of n samples $\mathbf{X}_n$ and the corresponding quantized version $\tilde{\mathbf{X}}_n$ is defined by the average distortion over the n source output samples, i.e.

$$d\left(\mathbf{X}_n, \tilde{\mathbf{X}}_n\right) = \frac{1}{n} \sum_{k=1}^{n} d(x_k, \tilde{x}_k).$$

Rate-distortion function

- The source output is a random process and, hence, the n samples in $\mathbf{X}_n$ are random variables. As a result, $d\left(\mathbf{X}_n, \tilde{\mathbf{X}}_n\right)$ is also a random variable.
- The expectation of $d\left(\mathbf{X}_n, \tilde{\mathbf{X}}_n\right)$ is defined as the distortion D, i.e.

$$D = E\left[d\left(\mathbf{X}_n, \tilde{\mathbf{X}}_n\right)\right] = \frac{1}{n}\sum_{k=1}^{n} E[d(x_k, \tilde{x}_k)] = E[d(x_k, \tilde{x}_k)],$$

  where the last equation results from our assumption of a stationary source output process.
- Suppose we have a memoryless source with a continuous-amplitude output $\mathbf{X}$ that has a PDF $p(x)$, a quantized amplitude output alphabet $\tilde{\mathbf{X}}$ and a per-letter distortion measure $d(x, \tilde{x})$, where $x \in \mathbf{X}$ and $\tilde{x} \in \tilde{\mathbf{X}}$. Then the minimum rate in bit per source output that is required to represent the output $\mathbf{X}$ of the memoryless source with a distortion less than or equal to D is called the rate-distortion function $R(D)$ and is defined as

$$R(D) = \min_{p(\tilde{x}|x): E\left[d\left(\mathbf{X}, \tilde{\mathbf{X}}\right)\right] \leq D} I\left(\mathbf{X}; \tilde{\mathbf{X}}\right),$$

  where $I\left(\mathbf{X}; \tilde{\mathbf{X}}\right)$ is the average mutual information between $\mathbf{X}$ and $\tilde{\mathbf{X}}$.

Rate-distortion function

- In general, the rate $R(D)$ decreases for increasing distortion D or, conversely, $R(D)$ increases for decreasing D.
- One interesting model of a continuous amplitude memoryless information source is the Gaussian source model. In this case, Shannon (1959) proved the following fundamental theorem on the rate-distortion function:

Theorem: Rate-distortion function for a memoryless Gaussian source

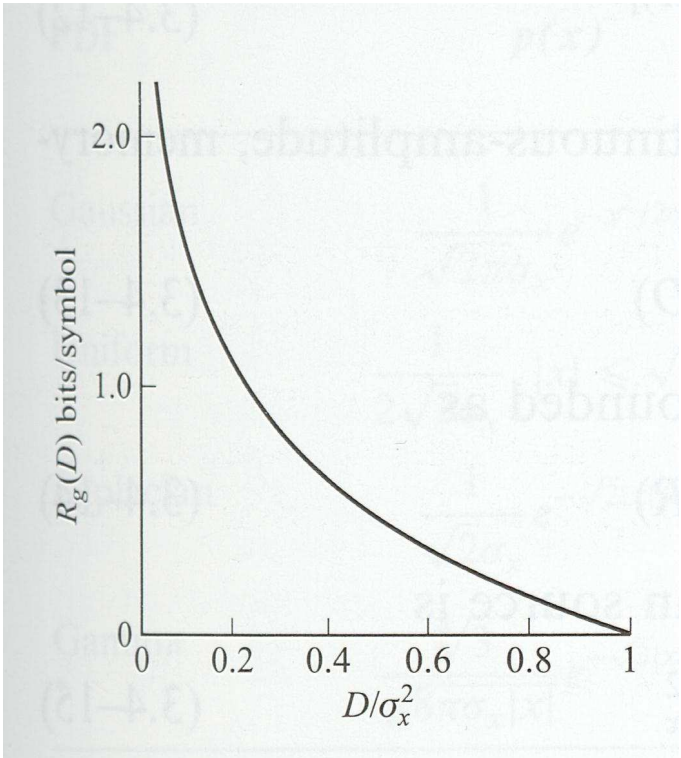
The minimum information rate necessary to represent the output of a discrete time continuous amplitude memoryless Gaussian source based on a mean-square-error distortion measure per symbol (single letter distortion measure) is

$$R_g(D) = \begin{cases} \frac{1}{2}\mathrm{ld}\left(\sigma_x^2/D\right) & \text{for } 0 \leq D \leq \sigma_x^2 \\ 0 & \text{for } D > \sigma_x^2 \end{cases},$$

where σ_x^2 is the variance of the Gaussian source output.

Rate-distortion function

- Obviously, no information need to be transmitted for $D \geq \sigma_x^2$.
- Specifically, $D = \sigma_x^2$ can be obtained by using zeros in the reconstruction of the signal. For $D > \sigma_x^2$, we can use statistically independent , zero-mean Gaussian noise samples with a variance of $D - \sigma_x^2$ for the reconstruction.



Rate-distortion function for a continuous-amplitude memoryless Gaussian source.

Rate-distortion function

- The rate-distortion function $R(D)$ of a source is associated with the following basic source coding theorem in information theory (Shannon, 1959):

Theorem: Source coding with a distortion measure
There exists an encoding scheme that maps the source output into code words such that for any given distortion D, the minimum rate $R(D)$ bits per symbol (sample) is sifficient to reconstruct the source output with an average distortion that is arbitrarily close to D.

- It is clear, therefore, that the rate-distortion function $R(D)$ for any source represents a lower bound on the source rate that is possible for a given level of distortion.
- Returning to the memoryless Gaussian source, we want to reverse the functional dependency in the expression

$$R_g(D) = \begin{cases} \frac{1}{2}\mathrm{ld}\left(\sigma_x^2/D\right) & \text{for } 0 \leq D \leq \sigma_x^2 \\ 0 & \text{for } D > \sigma_x^2 \end{cases}$$

and express D as a function of R.

Rate-distortion function

- We obtain the distortion-rate function for the discrete-time memoryless Gaussian source

$$D_g(R) = 2^{-2R}\sigma_x^2.$$

- When we express the distortion in decibel, we obtain

$$D_{g,\mathrm{dB}}(R) = 10\log_{10}\left(D_g(R)\right) = -6R + 10\log_{10}\left(\sigma_x^2\right).$$

- Note that the mean-square-error distortion decreases at the rate of 6dB/bit.

Theorem: Upper bound on $R(D)$ The rate-distortion function of a memoryless continuous-amplitude source with zero mean and finite variance σ_x^2 with respect to the mean-square-error distortion measure is upper-bounded as

$$R(D) \leq \frac{1}{2}\mathrm{ld}\left(\frac{\sigma_x^2}{D}\right), \qquad 0 \leq D \leq \sigma_x^2.$$

Rate-distortion function

- The theorem implies that the Gaussian source requires the maximum rate among all other sources for a specified level of mean-square-error distortion. Thus, the rate distortion $R(D)$ of any continuous-amplitude memoryless source with zero mean and finite variance σ_x^2 satisfies the condition $R(D) \leq R_g(D)$.
- Similarly, the distortion-rate function of the same source satisfies the condition

$$D(R) \leq D_g(R) = 2^{-2R}\sigma_x^2.$$

- A lower bound $R^*(D)$ on the rate-distortion function also exists (Shannon lower bound for a mean-square-error distortion measure:

$$R^*(D) = H(X) - \frac{1}{2}\mathrm{ld}(2\pi \mathrm{e} D),$$

  where $H(X)$ is the differential entropy of the continuous-amplitude memoryless source.
- The corresponding distortion-rate function reads

$$D^*(R) = \frac{1}{2\pi \mathrm{e}}\, 2^{-2(R-H(X))}.$$

Rate-distortion function

- In summary, we obtain the following inequalities:

$$\begin{array}{ccccc} R^*(D) & \leq & R(D) & \leq & R_g(D) \\ D^*(R) & \leq & D(R) & \leq & D_g(R). \end{array}$$

- It is well-known that the differential entropy of the memoryless Gaussian source is

$$H_g(X) = \frac{1}{2}\mathrm{ld}\left(2\pi \mathrm{e}\sigma_x^2\right),$$

so that we have $R^*(D) = R_g(D)$.

- If we express $D^*(R)$ in decibel and normalize it by setting $\sigma_x^2 = 1$ (or dividing $D^*(R)$ by σ_x^2), we obtain

$$D^*_{\mathrm{dB}}(R) = 10\log_{10}\left(D^*(R)\right) = -6R - 6(H_g(X) - H(X))$$

or equivalently

$$10\log_{10}\frac{D_g(R)}{D^*(R)} = 6[H_g(X) - H(X)]\,\mathrm{dB} = 6[R_g(D) - R^*(D)]\,\mathrm{dB}.$$

- These relations allos us to compare the lower bound in the distortion with the upper bound which is the distortion for the Gaussian source. Note that $D^*(R)$ also decreases at -6 dB/bit.

Rate-distortion function

PDF	$p(x)$	$H(X)$	$R_g(D) - R^*(D)$ (bits/sample)	$D_g(R) - D^*(R)$ (dB)
Gaussian	$\frac{1}{\sqrt{2\pi}\sigma_x} e^{-x^2/2\sigma_x^2}$	$\frac{1}{2}\log_2(2\pi e\sigma_x^2)$	0	0
Uniform	$\frac{1}{2\sqrt{3}\sigma_x}, \lvert x\rvert \leqslant \sqrt{3}\sigma_x$	$\frac{1}{2}\log_2(12\sigma_x^2)$	0.255	1.53
Laplacian	$\frac{1}{\sqrt{2}\sigma_x} e^{-\sqrt{2}\lvert x\rvert/\sigma_x}$	$\frac{1}{2}\log_2(2e^2\sigma_x^2)$	0.104	0.62
Gamma	$\frac{\sqrt[4]{3}}{\sqrt{8\pi\sigma_x\lvert x\rvert}} e^{-\sqrt{3}\lvert x\rvert/2\sigma_x}$	$\frac{1}{2}\log_2(4\pi e^{0.423}\sigma_x^2/3)$	0.709	4.25

Differential entropies and rate-distortion comparisons of four common PDFs for signal models.

Rate-Distortion Function

Finally, let us consider the example of a band-limited Gaussian source with power spectral density

$$\Phi(f) = \begin{cases} \sigma_x^2/w & \text{for } |f| \leq W \\ 0 & \text{for } |f| > W \end{cases}.$$

- When the output of this source is sampled at the Nyquist rate, the samples are uncorrelated and, by the Gaussian distributions, statistically independent. Hence, the equivalent discrete-time Gaussian source is memoryless.
- The rate-distortion function for the Gaussian source in bit/sample is

$$R_g(D) = \frac{1}{2}\mathrm{ld}\left(\sigma_x^2/D\right) \qquad \text{for } 0 \leq D \leq \sigma_x^2.$$

- Since we have for Nyquist sampling that the sampling time interval T_s satisfies $(1 \text{ sample})T_\mathrm{s}^{-1} = 2W \text{sample}$, we obtain the rate-distortion function for the Gaussian source in bit/s

$$R_g(D) = W\mathrm{ld}\left(\sigma_x^2/D\right) \qquad \text{for } 0 \leq D \leq \sigma_x^2.$$

- The corresponding distortion-rate function is $D_g(R) = 2^{-R/W}\sigma_x^2$, which, when expressed in decibel and normalized by σ_x^2, becomes

$$10\log_{10}\frac{D_g(R)}{\sigma_x^2} = -\frac{3R}{W}.$$

Coding Techniques For Analog Sources

- Having investigated the theoretical bounds of encoding analog source signals, we have to choose techniques for implementing suitable encoding schemes.
- Most of the existing schemes have been applied to encoding speech and image signals.
- Analog source encoding schemes can be classified into three categories:
  - **Temporal waveform coding:** In this type of encoding, the source encoder is designed to represent digitally the temporal characteristics of the source waveform
  - **Spectral waveform coding:** The signal waveform is usually subdivided into different frequency bands, and either the time waveform in each band or its spectral characteristics are encoded for transmission.
  - **Model-based coding:** Here, the encoding is done based on an explicit mathematical model of the source signal.

Temporal waveform coding

- There are several analog source coding techniques that are designed to represent the time-domain characteristics of the signal.
- Here, we will consider
  - pulse-code modulation (PCM)
  - differential pulse-code modulation (DPCM)
  - adaptive pulse-code modulation (APCM)
  - delta modulation (DM).

PCM

- Let $x(t)$ denote a sample function emitted by a source and let x_n denote the samples taken at a sampling rate $f_s \geq 2W$, where W is the highest frequency in the spectrum of $x(t)$.
- In PCM, each sample of the signal is quantized to one of 2^R amplitude levels where R is the number of binary digits used to represent each sample. Thus, the rate from the source is Rf_s bit/s.
- The quantized process may be modeled mathematically as

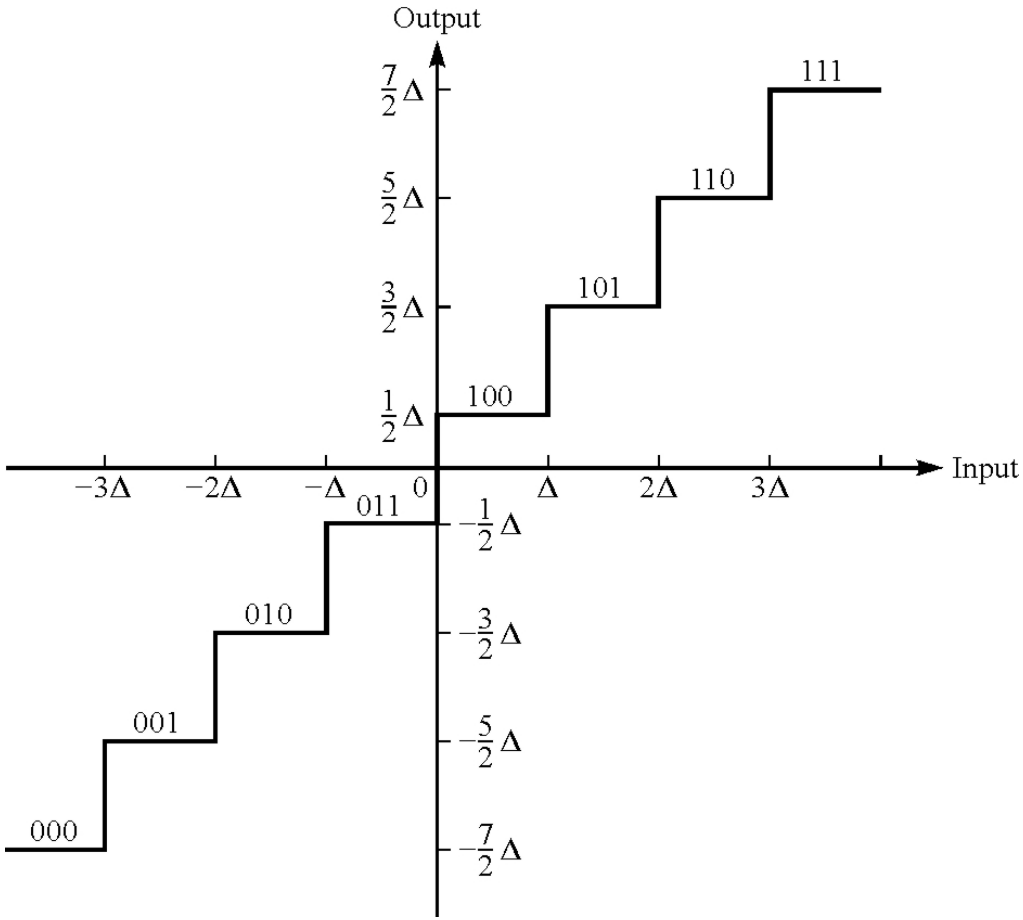
$$\tilde{x}_n = x_n + q_n,$$

  where $\tilde{x}_n$ represents the quantized value of x_n and q_n denotes the quantization error, which we treat as an additive noise.
- Assume that we use a uniform quantizer and use a mode with a correspondingly uniform PDF given by

$$p(q) = \frac{1}{\Delta}, \qquad -\frac{1}{2}\Delta \leq q \leq \frac{1}{2}\Delta,$$

  where the step size of the quantizer is $\Delta = 2^{-R}$.

PCM



Input-output characteristic for a uniform quantizer.

PCM

- **Problem** Show that the mean square value of the quantization error is

$$E[q^2] = \frac{\Delta^2}{12} = \frac{2^{-2R}}{12}$$

- Measured in decibels, the mean square value of the noise is

$$10\log_{10}\frac{\Delta^2}{12} = 10\log_{10}\frac{2^{-2R}}{12} = -6R - 10.8 \text{ dB}.$$

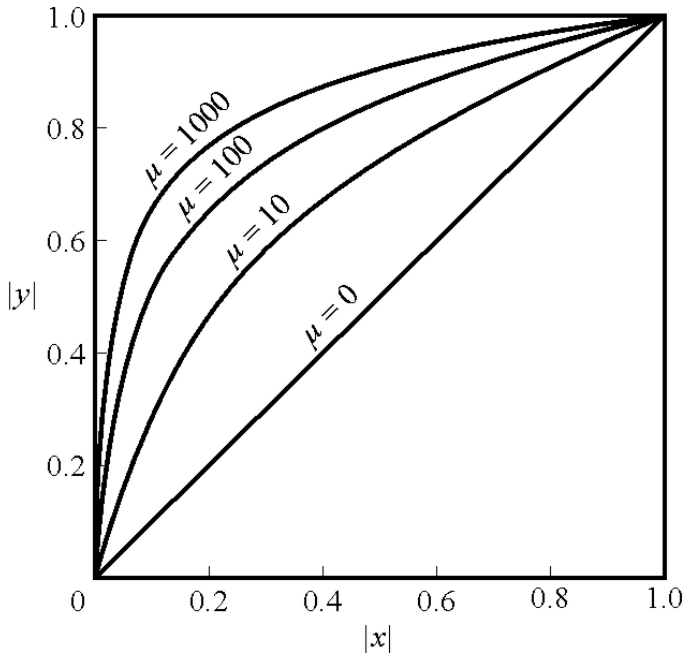
- We observe that the quantization noise decreases by 6 dB/bit used in the quantizer. For example, a 7-bit quantizer results in a quantization noise power of -52.8 dB.
- Many source signals (e.g. speech waveforms) have the characteristic that small signal amplitudes occur more frequently than large ones. However, a uniform quantizer provides the same spacing between successive levels throughout the entire dynamic range of the signal.
- Given that we know what this dynamic range is, a better approach is therefore to employ a nonuniform quantizer. The latter one is usually implemented in two steps:
  - pass the signal through a non-linear device that compresses the signal amplitude
  - the output of the nonlinearity is fed to a uniform quantizer.

PCM

An example of a compressor which can be configured by a parameter μ, is the logarithmic compressor:

$$|y| = \frac{\log(1 + \mu|x|)}{\log(1 + \mu)},$$

where $|x| \leq 1$ is the normalized magnitude of the input, $|y|$ is the magnitude of the output, and μ is a parameter to give the desired compression characteristic ($\mu = 0$ indicating no compression).



Input-output characteristic for a logarithmic compressor.

PCM

- In the encoding of speech waveforms, for example, the value of $\mu = 255$ has been adopted as a standard in the United States and Canada.
- This value results in about a 24-dB reduction in the quantization noise power relative to uniform quantization. Consequently, a 7-bit quantizer used in conjunction with a $\mu = 255$ logarithmic compressor produces a quantization noise power of approximately -77 dB compared with -53 dB for uniform quantization.
- Clearly, at the receiver which reconstructs the signal, we have to carry out an inverse logarithmic mapping in order to expand the signal amplitude. The combined compressor-expander pair is termed compandor.
- **Problem** Find the expander characteristic for the logarithmic compressor.

PCM

- In PCM, each sample of the waveform is encoded independently of all other samples.
- However, most source signals sampled at the Nyquist rate or faster exhibit significant correlation between successive samples. In other words: the average change in amplitude between successive samples is relatively small.
- Consequently, an encoding scheme that exploits the redundancy in the samples will result in a lower bit rate for the source output.
- A straightforward solution is to encode the differences between successive samples rather than the samples themselves. Since differences between samples are expected to be substantially smaller than the actual sampled amplitudes, fewer bits are required to represent the differences.
- A refinement of this general approach is to predict the current sample based on the previous p samples. Let x_n denote the current sample from the source and let $\hat{x}_n$ denote the predicted value of x_n defined as

$$\hat{x}_n = \sum_{i=1}^{p} a_i x_{n-i}$$

  with $\{a_i\}$ denoting the predictor coefficients. Clearly, $\hat{x}_n$ is a weighted average of the past p samples.
- We select the predictor coefficients to minimize some function of the error $e_n = x_n - \hat{x}_n$ between x_n and $\hat{x}_n$.

DPCM

- Using a mean square error (MSE) measure, we select $\{a_i\}$ to minimize $\varepsilon_p = E\left[e_n^2\right]$.
- **Problem** Assuming that the source output is wide-sense stationary, show that the minimum MSE measure provides the Yule-Walker equations

$$\sum_{i=1}^{p} a_i \phi(i-j) = \phi(j), \qquad j = 1, 2, \ldots, p,$$

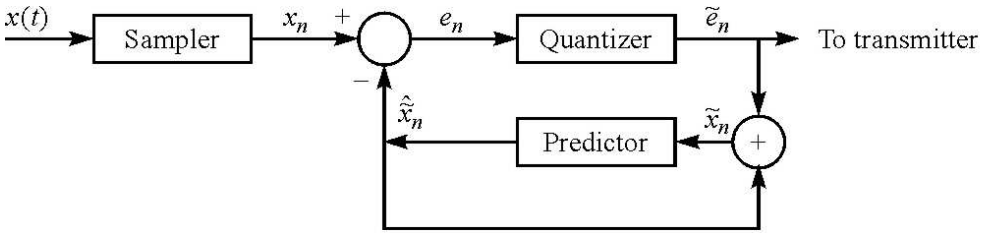
  where $\phi(m)$ is the autocorrelation function of the sampled signal sequence x_n.
- When the autocorrelation function is unknown, we can use an estimate of it instead of the true value. A frequently used estimator based on N samples of x is the sample autocorrelation function

$$\hat{\phi}(n) = \frac{1}{N} \sum_{i=1}^{N-n} x_i x_{i+n}, \qquad n = 0, 1, 2, \ldots, p.$$

- Obviously, the sample autocorrelation function is a biased estimate for $n \neq 0$. Other estimators are available and studied in the field of spectral estimation (e.g. using parametric or non-parametric approaches) since the power spectrum is the Fourier transform of the autocorrelation function. A simple yet accurate estimate of the power spectrum is provided by averaged or smoothed periodograms.
- Upon calculation of $\hat{\phi}(n)$, we use it instead of $\phi(n)$ in the Yule-Walker equations.

DPCM

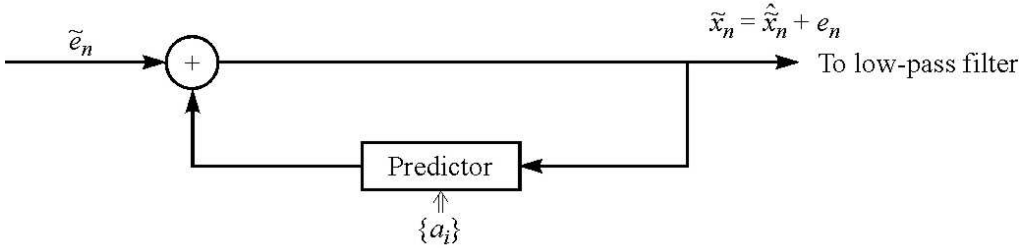
- The Yule-Walker equations can be solved efficiently using the so-called Levinson-Durbin algorithm which provides the predictor coefficients a_i for $i = 1, 2, \ldots, p$.
- Consider the block diagram of a practical DPCM system where the predictor is implemented with a feedback loop around the quantizer.



$$e_n = x_n - \hat{\tilde{x}}_n = x_n - \sum_{k=1}^{p} a_k \tilde{x}_{n-k}$$

$$\begin{aligned} \tilde{e}_n - e_n = \tilde{e}_n - (x_n - \hat{\tilde{x}}_n) &= e_n + \hat{\tilde{x}}_n - x_n \\ &= \tilde{x}_n - x_n \\ &= q_n = \text{quantization error} \end{aligned}$$

(a) Encoder



(b) Decoder

(a) Block diagram of a DPCM encoder. (b) DPCM decoder at the receiver

DPCM

- The input to the predictor is denoted by $\tilde{x}_n$ which, as before, represents the signal sample x_n modified by the quantization process.
- The output of the predictor is

$$\hat{\tilde{x}}_n = \sum_{i=1}^{p} a_i \tilde{x}_{n-i}.$$

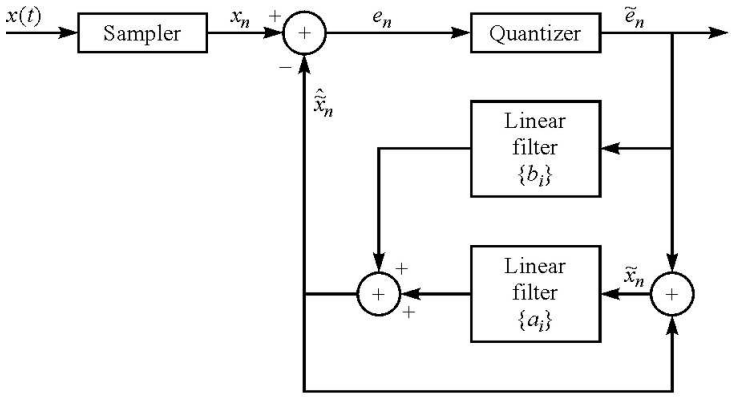
- The difference $e_n = x_n - \hat{\tilde{x}}_n$ is the input to the quantizer and $\tilde{e}_n$ denotes the output.
- Each value of the quantized prediction error $\tilde{e}_n$ is encoded into a sequence of binary digits and transmitted over the channel to the destination receiver. Furthermore, the quantized error $\tilde{e}_n$ is also added to the predicted value $\hat{\tilde{x}}_n$ to yield $\tilde{x}_n$.
- At the destination, the same predictor that was used at the transmitter is synthesized and its output $\hat{\tilde{x}}_n$ is added to $\tilde{e}_n$ to yield $\tilde{x}_n$. The signal $\tilde{x}_n$ is the desired excitation for the predictor and also the desired output sequence from which the reconstructed signal $\tilde{x}(t)$ is obtained by a suitable low-pass filtering.
- **Problem** Show that the use of the feedback around the quantizer guarantees that the quantized sample $\tilde{x}_n$ differs from the input x_n by the quantization error q_n independently of the predictor used.
- As a result, quantization errors do not accumulate.

DPCM

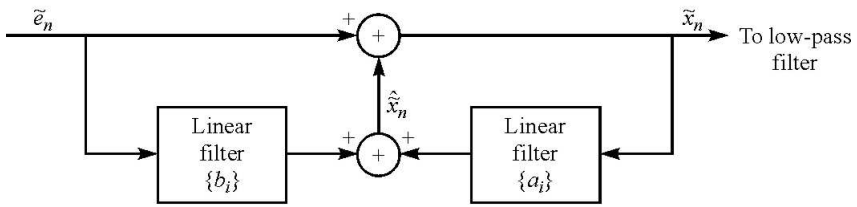
- An improvement in the quality of the estimate of $\hat{\tilde{x}}_n$ can be achieved if we include a linearly filtered version of the error signals $\tilde{e}_n$ according to

$$\hat{\tilde{x}}_n = \sum_{i=1}^{p} a_i \tilde{x}_{n-i} + \sum_{i=1}^{m} b_i \tilde{e}_{n-i},$$

where $\{b_i\}$ is the set of filter coefficients for the quantized error sequence $\tilde{e}_n$



(a) Encoder



(b) Decoder

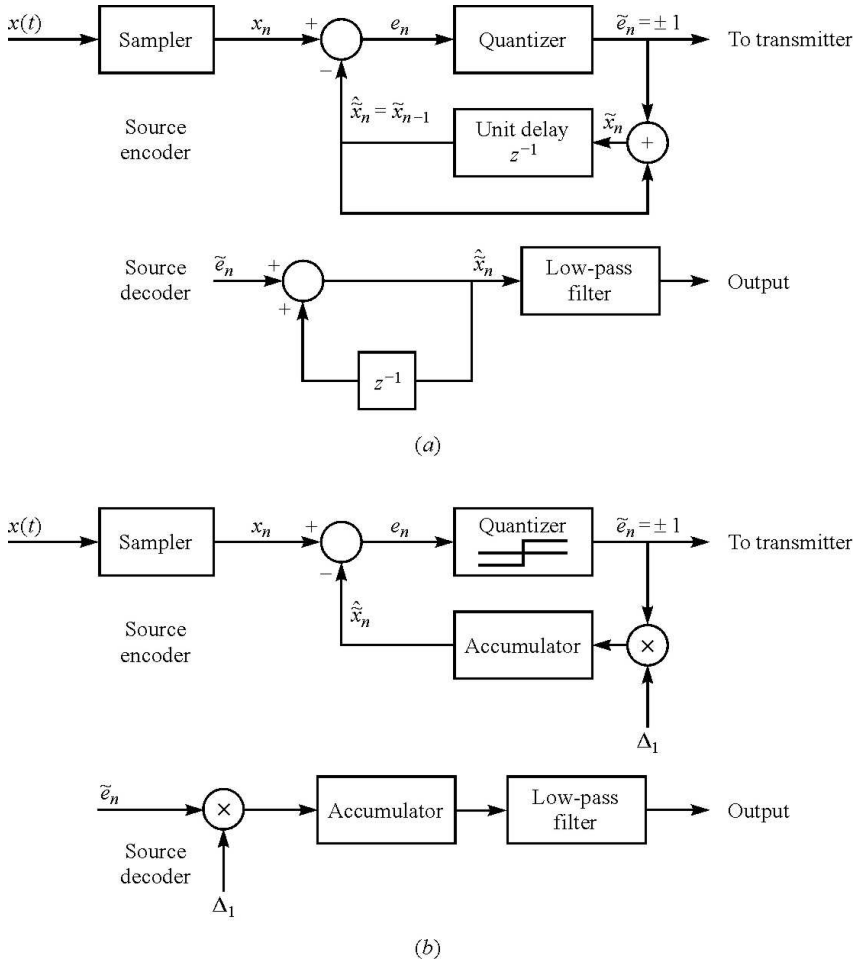
DPCM modified by the addition of linearly filtered error sequence.

APCM

- Many real sources are quasi-stationary in nature, i.e., the variance and the autocorrelation function of the source output vary slowly with time.
- PCM and DPCM encoders, however, are designed on the bases that the source outputs is stationary.
- The efficiency and performance of these encoders can be improved by having them adapt to the slowly time variant statistics of the source.
- In both PCM and DPCM, the quantization error q_n resulting from a uniform quantizer operating on a quasi-stationary input signal will have a time-variant variance (quantization noise power).
- One way of reducing the dynamic range of the quantization noise is the use of an adaptive quantizer.
- There are different ways to implement this adaptation with correspondingly different consequences for the decoding process and the required bit rate over the channel.

DM

- Delta modulation may be viewed as a simplified form of DPCM in which a two-level (1-bit) quantizer is used in conjunction with a fixed first-order predictor.

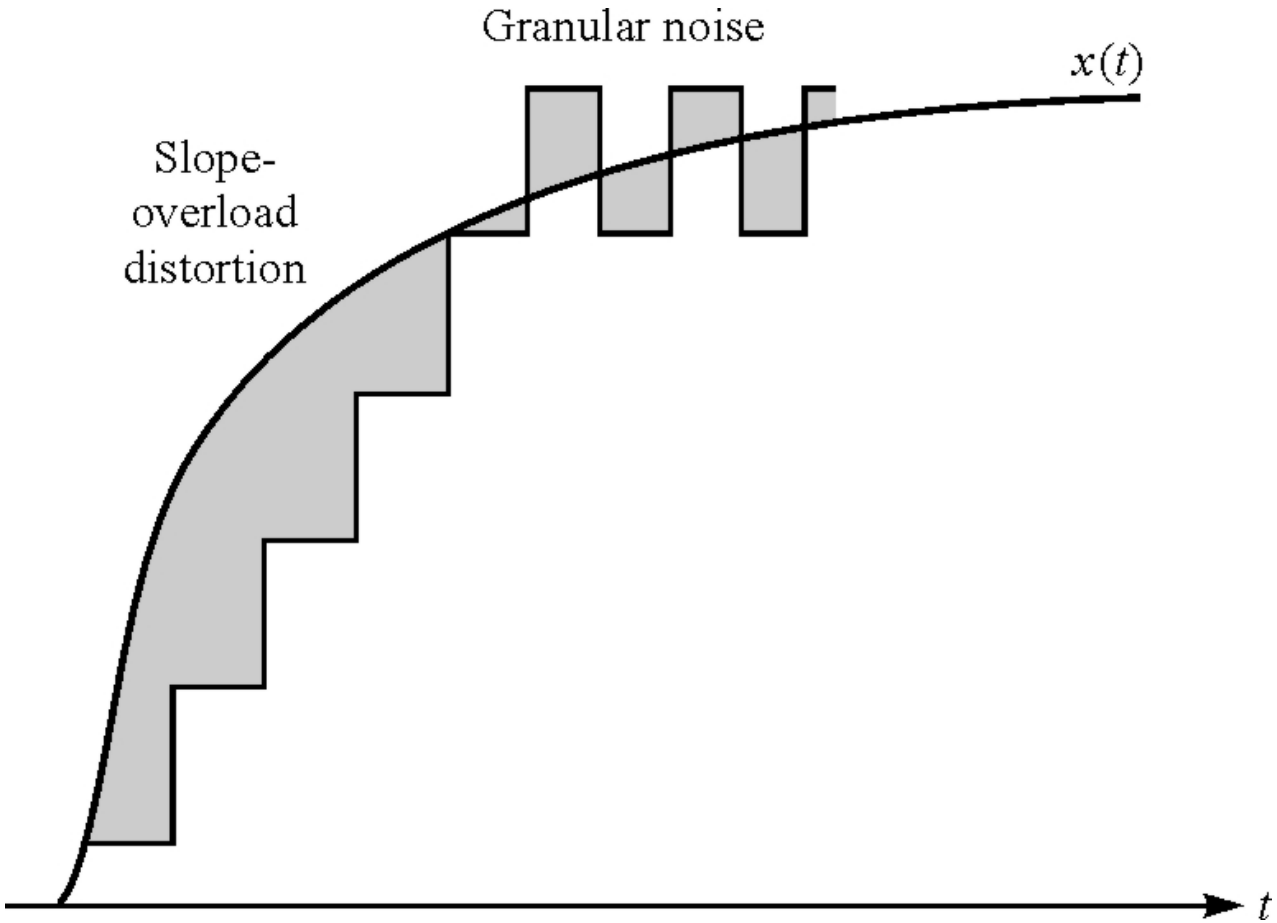


(a) Block diagram of a delta modulation system. (b) An equivalent realization of a delta modulation system.

DM

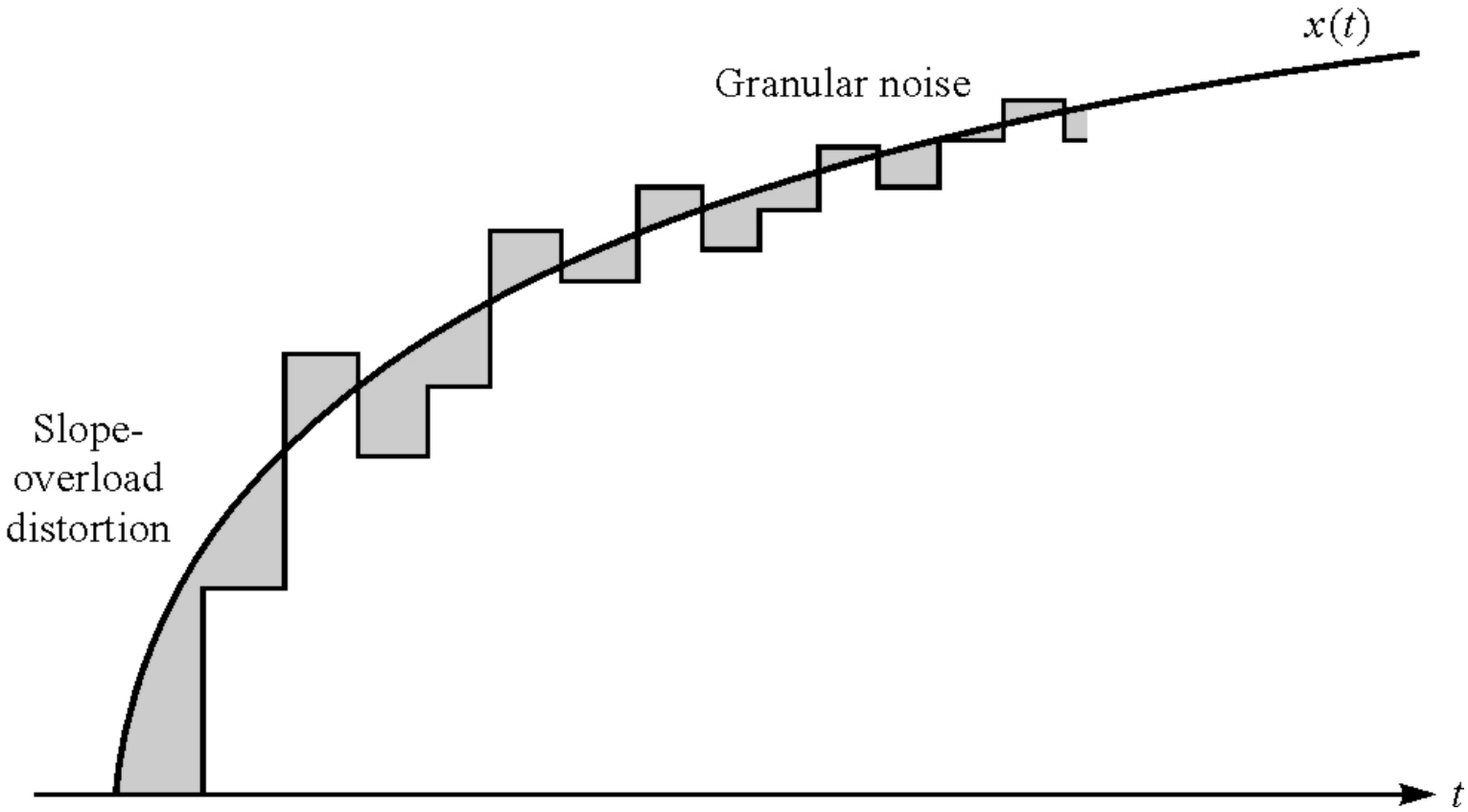
- We have
$$\hat{\tilde{x}}_n = \tilde{x}_{n-1} = \hat{\tilde{x}}_{n-1} + \tilde{e}_{n-1}.$$
- Since
$$q_{n-1} = \tilde{e}_{n-1} - e_{n-1} = \tilde{e}_{n-1} - \left(x_{n-1} - \hat{\tilde{x}}_{n-1}\right) = \hat{\tilde{x}}_{n-1} + \tilde{e}_{n-1} - x_{n-1} = \hat{\tilde{x}}_n - x_{n-1},$$
we have
$$\hat{\tilde{x}}_n = x_{n-1} + q_{n-1}.$$
- Thus, the estimated (predicted) value of x_n is really the previous sample x_{n-1} modified by the quantization noise q_{n-1}.
- Obviously, $\hat{\tilde{x}}_n = \hat{\tilde{x}}_{n-1} + \tilde{e}_{n-1}$ is an integrator (accumulator) with input $\tilde{e}_n$.
- The encoder approximates the waveform $x(t)$ by a linear staircase function.
- In order for the approximation to be relatively accurate, the waveform $x(t)$ must change slowly relative to the sampling rate requiring a certain oversampling (a factor of at least 5) as compared to the Nyquist rate.
- At any given sampling rate, the encoder is limited by slope-overload distortion (step size too small to follow changes with steep slope) or granular noise (step size too large for changes with small slope).

DM



Distortion in a delta modulation system (slope-overload and granular noise).

DM



Distortion in a delta modulation system with variable step-size adapted to the signal slope.

Spectral waveform coding

Subband coding:

- Subband coding is usually applied to speech and image signals where the signal is divided into a small number of subbands and the time waveform in each subband is encoded separately.
- In speech coding, for example, the lower-frequency bands contain most of the spectral energy.
- In addition, quantization noise is more noticeable to the ear in the lower -frequency bands.
- Consequently, more bits are used for the lower-band signals and fewer are used for the higher-frequency bands.
- Filter design is particularly important in achieving good performance. Usually, quadrature-mirror filters (QMFs) having the so-called perfect reconstruction property are used.

Spectral waveform coding

Subband coding:

- Using QMFs, the lower-frequency band is repeatedly subdivided by factors of 2, in order to reduce the sampling rate. For example, suppose a bandwidth of the speech signal of 3200 Hz.
- The first pair of QMFs divides the spectrum into the low (0-1600 Hz) and the high (1600-3200 Hz) bands. Then, the low band is again split into two band of bandwidth 800 Hz each.
- A third subdivison provides two further subbands. Thus, with three pairs of QMFs, we have obtained signals in the frequency bands 0-400, 400-800, 800-1600 and 1600-3200 Hz.
- The time-domain signal in each subband may now be encoded with different precision.
- Adaptive PCM is used for waveform encoding of the signal in each subband.

Spectral waveform coding

Adaptive transform coding:

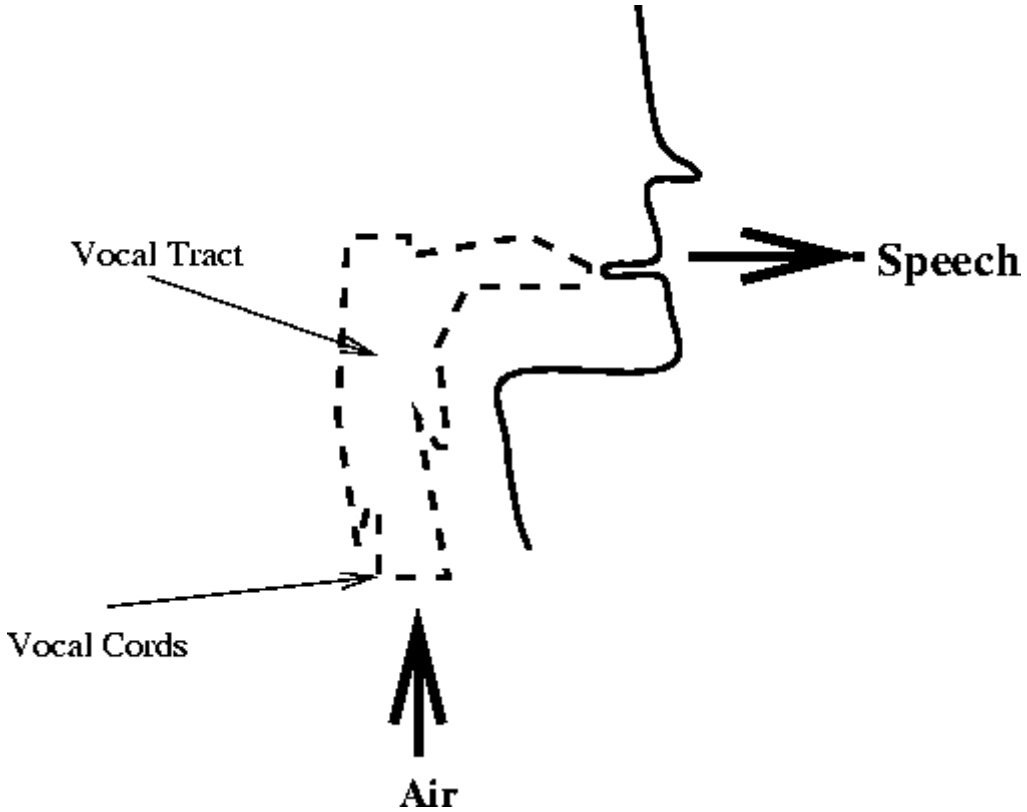
- In adaptive transform coding (ATC), the source signal is sampled and subdivided into frames of N_f samples. The data in each frame is transformed into the spectral domain for coding and transmission.
- At the source decoder, each frame of spectral samples is transformed back into the time domain and the signal is synthesized from the time-domain samples and passed through a digital-to-analog converter.
- To achieve coding efficiency, we assign more bits to the more important spectral coefficients and fewer bits to the less important spectral coefficients.
- In addition, by designing an adaptive allocation in the assignment of the total number of bits to the spectral coefficients, we can adapt to possibly changing statistics of the source signal.
- The transformation to the frequency domain should result in uncorrelated spectral samples. To this end, a discrete cosine transform is employed.
- In speech coding using ATC, about 9600 bit/s are needed to achieve communication-quality speech.

Model-based source coding

- The compression of speech signals has many practical applications. One example is in digital cellular technology where many users share the same frequency bandwidth.
- Compression allows more users to share the system than otherwise possible. Another example is in digital voice storage (e.g. answering machines). For a given memory size, compression allows longer messages to be stored than otherwise.
- Historically, digital speech signals are sampled at a rate of 8000 samples/s. Typically, each sample is represented by 8 bits (using a logarithmic compressor). This corresponds to an uncompressed rate of 64 kbit/s.
- With current compression techniques (all of which are lossy), it is possible to reduce the rate to 8 kbit/s with almost no perceptible loss in quality.
- Further compression is possible at a cost of lower quality.
- All of the current low-rate speech coders are based on the principle of linear predictive coding (LPC).

Model-based source coding

LPC modeling:



Physical model of speech generation.

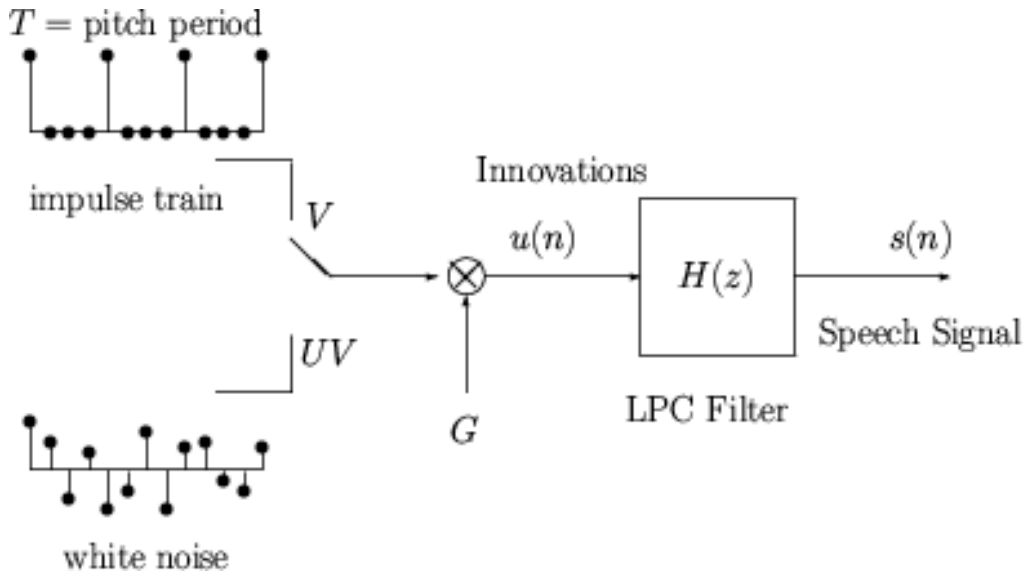
Model-based source coding

When you speak:

- Air is pushed from your lung through your vocal tract and out of your mouth comes speech.
- For certain voiced sound, your vocal cords vibrate (open and close). The rate at which the vocal cords vibrate determines the pitch of your voice. Women and young children tend to have high pitch (fast vibration) while adult males tend to have low pitch (slow vibration).
- For certain fricatives and plosive (or unvoiced) sound, your vocal cords do not vibrate but remain constantly opened.
- The shape of your vocal tract determines the sound that you make.
- As you speak, your vocal tract changes its shape producing different sound.
- The shape of the vocal tract changes relatively slowly (on the scale of 10 msec to 100 msec).
- The amount of air coming from your lung determines the loudness of your voice.

Model-based source coding

Mathematical model:



Mathematical model of speech generation.

- The above model is often called the LPC Model.
- The model says that the digital speech signal is the output of a digital filter (called the LPC filter) whose input is either a train of impulses or a white noise sequence.

Model-based source coding

The relationship between the physical and the mathematical models:

Vocal Tract	$\Leftrightarrow$	$H(z)$	(LPC Filter)
Air	$\Leftrightarrow$	$u(n)$	(Innovations)
Vocal Cord Vibration	$\Leftrightarrow$	V	(voiced)
Vocal Cord Vibration Period	$\Leftrightarrow$	T	(pitch period)
Fricatives and Plosives	$\Leftrightarrow$	UV	(unvoiced)
Air Volume	$\Leftrightarrow$	G	(gain)

- A standard LPC filter is given by

$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{10} z^{-10}},$$

which is equivalent to saying that the input-output relation of the filter is given by the linear difference equation

$$s(n) + \sum_{i=1}^{10} a_i s(n-i) = u(n).$$

Model-based source coding

- The LPC model can be represented in vector form as

$$\mathbf{A} = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, G, V/UV, T).$$

- The vector $\mathbf{A}$ changes changes every 20 ms or so. At a sampling rate of 8000 samples/s, 20 ms is equivalent to 160 samples.
- The digital speech signal is divided into frames of size 20 ms. There are 50 frames/s.
- The model says that

$$\mathbf{A} = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, G, V/UV, T).$$

  is equivalent to

$$\mathbf{S} = (s(0), s(1), \ldots, s(159)).$$

  Thus the 160 values of $\mathbf{S}$ are compactly represented by the 13 values of $\mathbf{A}$.
- There is almost no perceptual difference in $\mathbf{S}$ if
  - for Voiced Sounds (V), the impulse train is shifted (insensitive to phase change)
  - for Unvoiced Sounds (UV), a different white noise sequence is used.

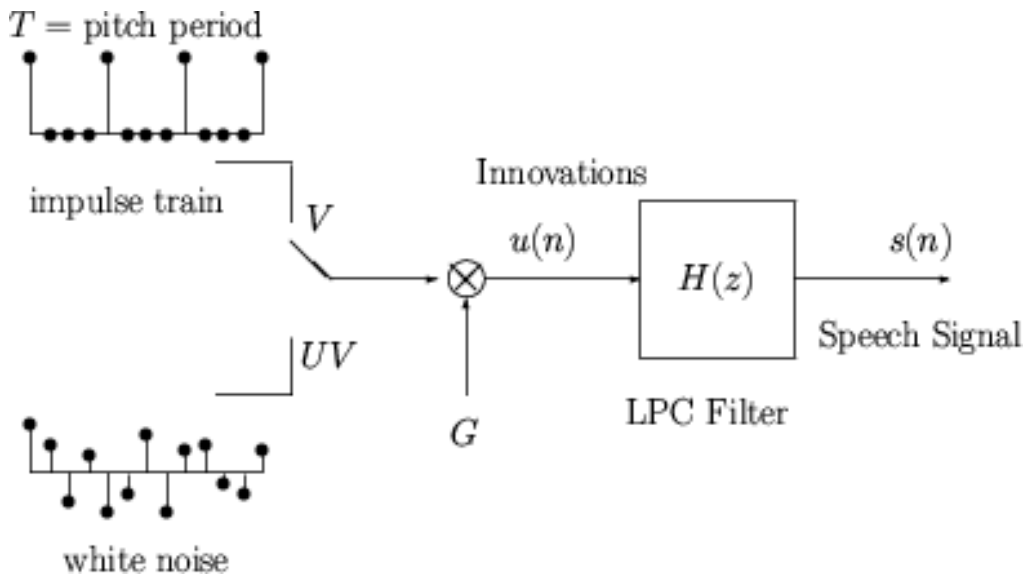
Model-based source coding

LPC analysis at the transmitter:

- For a given $\mathbf{S}$, find the $\mathbf{A}$ minimizing the expected squared error (i.e., solve the Yule-Walker equations).

LPC synthesis at the receiver:

- Upon reception of the vector $\mathbf{A}$, estimate the signal $\mathbf{S}$.
- This is done using standard filtering techniques based on the mathematical model.



Mathematical model of speech generation.

Comparison of encoding techniques for speech signals

Encoding method	Quantizer	Coder	Transmission rate (bits/s)
PCM	Linear	12 bits	96,000
Log PCM	Logarithmic	7–8 bits	56,000–64,000
DPCM	Logarithmic	4–6 bits	32,000–48,000
ADPCM	Adaptive	3–4 bits	24,000-32,000
DM	Binary	1 bit	32,000–64,000
ADM	Adaptive binary	1 bit	16,000–32,000
LPC/CELP			2400–9600

Encoding techniques applied to speech signals.