Chapter 3
Source Coding
3.1 Mathematical Models for Information Sources
3.2 A Logarithmic Measure of Information
  3.2.1 Average Mutual Information and Entropy
  3.2.2 Information Measures for Continuous Random Variables
3.3 Coding for Discrete Sources
  3.3.1 Coding for Discrete Memoryless Sources
  3.3.2 Discrete Stationary Sources
  3.3.3 The Lempel-Ziv Algorithm
Table of Contents (cont.)

3.4 Coding for Analog Sources – Optimum Quantization
   3.4.1 Rate-Distortion Function
   3.4.2 Scalar Quantization
   3.4.3 Vector Quantization

3.5 Coding Techniques for Analog Sources
   3.5.1 Temporal Waveform Coding
   3.5.2 Spectral Waveform Coding
   3.5.3 Model-Based Source Coding
Introduction

- Two types of sources: analog source and digital source.
- Whether a source is analog or discrete, a digital communication system is designed to transmit information in digital form.
- The output of the source must be converted to a format that can be transmitted digitally.
- This conversion of the source output to a digital form is generally performed by the source encoder, whose output may be assumed to be a sequence of binary digits.
- In this chapter, we treat source encoding based on mathematical models of information sources and provide a quantitative measure of the information emitted by a source.
3.1 Mathematical Models for Information Sources

- The output of any information source is random.
- The source output is characterized in statistical terms.
- To construct a mathematical model for a discrete source, we assume that each letter in the alphabet \( \{x_1, x_2, \ldots, x_L\} \) has a given probability \( p_k \) of occurrence.

\[
p_k = P(X = x_k), \quad 1 \leq k \leq L
\]

\[
\sum_{k=1}^{L} p_k = 1
\]
3.1 Mathematical Models for Information Sources

- Two mathematical models of discrete sources:
  - If the output sequence from the source is statistically independent, i.e. the current output letter is statistically independent from all past and future outputs, then the source is said to be memoryless. Such a source is called a discrete memoryless source (DMS).
  - If the discrete source output is statistically dependent, we may construct a mathematical model based on statistical stationarity. By definition, a discrete source is said to be stationary if the joint probabilities of two sequences of length $n$, say $a_1, a_2, \ldots, a_n$ and $a_{1+m}, a_{2+m}, \ldots, a_{n+m}$, are identical for all $n \geq 1$ and for all shifts $m$. In other words, the joint probabilities for any arbitrary length sequence of source outputs are invariant under a shift in the time origin.
3.1 Mathematical Models for Information Sources

Analog source

- An analog source has an output waveform $x(t)$ that is a sample function of a stochastic process $X(t)$.
- Assume that $X(t)$ is a stationary stochastic process with autocorrelation function $\phi_{xx}(\tau)$ and power spectral density $\Phi_{xx}(f)$. When $X(t)$ is a band-limited stochastic process, i.e., $\Phi_{xx}(f)=0$ for $|f| \geq W$, the sampling theorem may be used to represent $X(t)$ as:

$$X(t) = \sum_{n=-\infty}^{\infty} X\left(\frac{n}{2W}\right) \sin \left[2\pi W \left(t - \frac{n}{2W}\right)\right]$$

- By applying the sampling theorem, we may convert the output of an analog source into an equivalent discrete-time source.
3.1 Mathematical Models for Information Sources

Analog source (cont.)

- The output samples \( \{X(n/2W)\} \) from the stationary sources are generally continuous, and they can’t be represented in digital form without some loss in precision.
- We may quantize each sample to a set of discrete values, but the quantization process results in loss of precision.
- Consequently, the original signal can’t be reconstructed exactly from the quantized sample values.
3.2 A Logarithmic Measure of Information

- Consider two discrete random variables with possible outcomes $x_i$, $i=1,2,\ldots,n$, and $y_i$, $i=1,2,\ldots,m$.
- When $X$ and $Y$ are statistically independent, the occurrence of $Y=y_j$ provides no information about the occurrence of $X=x_i$.
- When $X$ and $Y$ are fully dependent such that the occurrence of $Y=y_j$ determines the occurrence of $X=x_i$, the information content is simply that provided by the event $X=x_i$.
- **Mutual Information** between $x_i$ and $y_j$: the information content provided by the occurrence of the event $Y=y_j$ about the event $X=x_i$, is defined as:

$$I(x_i; y_j) = \log \frac{P(x_i \mid y_j)}{P(x_i)}$$
3.2 A Logarithmic Measure of Information

- The units of $I(x_i, y_j)$ are determined by the base of the logarithm, which is usually selected as either 2 or $e$.
- When the base of the logarithm is 2, the units of $I(x_i, y_j)$ are **bits**.
- When the base is $e$, the units of $I(x_i, y_j)$ are called **nats** (natural units).
- The information measured in nats is equal to $\ln 2$ times the information measured in bits since:

$$\ln a = \ln 2 \log_2 a = 0.69315 \log_2 a$$

- When $X$ and $Y$ are statistically independent, $p(x_i|y_j) = p(x_i)$, $I(x_i, y_j) = 0$.
- When $X$ and $Y$ are fully dependent, $P(x_i|y_j) = 1$, and hence $I(x_i, y_j) = -\log P(x_i)$. 

\[
\log_a b = \frac{\log_c b}{\log_c a}
\]
3.2 A Logarithmic Measure of Information

- **Self-information** of the event $X=x_i$ is defined as $I(x_i)=-\log P(x_i)$.
- Note that a high-probability event conveys less information than a low-probability event.
- If there is only a single event $x$ with probability $P(x)=1$, then $I(x)=0$.
- Example 3.2-1: A discrete information source that emits a binary digit with equal probability.
  - The information content of each output is:
    $$I(x_i) = -\log_2 P(x_i) = -\log_2 \frac{1}{2} = 1 \text{ bit, } x_i=0,1$$
  - For a block of $k$ binary digits, if the source is memoryless, there are $M=2^k$ possible $k$-bit blocks. The self-information is:
    $$I(x'_i) = -\log_2 2^{-k} = k \text{ bits}$$
The information provided by the occurrence of the event \( Y = y_j \) about the event \( X = x_i \) is identical to the information provided by the occurrence of the event \( X = x_i \) about the event \( Y = y_j \) since:

\[
I(x_i; y_j) = \frac{P(x_i | y_j)}{P(x_i)} = \frac{P(x_i | y_j)P(y_j)}{P(x_i)P(y_j)} = \frac{P(x_i, y_j)}{P(x_i)P(y_j)}
\]

\[
= \frac{P(y_j | x_i)P(x_i)}{P(x_i)P(y_j)} = \frac{P(y_j | x_i)}{P(y_j)} = I(y_j; x_i)
\]

Example 3.2-2: \( X \) and \( Y \) are binary-valued \( \{0,1\} \) random variables that represent the input and output of a binary channel.

- The input symbols are equally likely.
3.2 A Logarithmic Measure of Information

Example 3.2-2 (cont.):

The output symbols depend on the input according to the conditional probability:

\[ P(Y = 0 \mid X = 0) = 1 - p_0 \quad P(Y = 1 \mid X = 0) = p_0 \]

\[ P(Y = 1 \mid X = 1) = 1 - p_1 \quad P(Y = 0 \mid X = 1) = p_1 \]

Mutual information about \( X=0 \) and \( X=1 \), given that \( Y=0 \):

\[ P(Y = 0) = P(Y = 0 \mid X = 0) P(X = 0) + P(Y = 0 \mid X = 1) P(X = 1) = \frac{1}{2} (1 - p_0 + p_1) \]

\[ P(Y = 1) = P(Y = 1 \mid X = 0) P(X = 0) + P(Y = 1 \mid X = 1) P(X = 1) = \frac{1}{2} (1 - p_1 + p_0) \]
3.2 A Logarithmic Measure of Information

Example 3.2-2 (cont.)

The mutual information about $X=0$ given that $Y=0$ is:

$$I(x_1; y_1) = I(0; 0) = \log_2 \frac{P(Y = 0 | X = 0)}{P(Y = 0)} = \log_2 \frac{2(1 - p_0)}{1 - p_0 + p_1}$$

The mutual information about $X=1$ given that $Y=0$ is:

$$I(x_2; y_1) \equiv I(1; 0) = \log_2 \frac{2p_1}{1 - p_0 + p_1}$$

- If the channel is **noiseless**, $p_0 = p_1 = 0$:
  
  $$I(0; 0) = \log_2 2 = 1 \text{ bit}$$

- If the channel is **useless**, $p_0 = p_1 = 0.5$:
  
  $$I(0; 0) = \log_2 1 = 0 \text{ bit}$$
### 3.2 A Logarithmic Measure of Information

- **Conditional self-information** is defined as:

  \[ I(x_i \mid y_j) = \log \frac{1}{P(x_i \mid y_j)} = - \log P(x_i \mid y_j) \]

  \[ I(x_i ; y_j) = \log P(x_i \mid y_j) - \log P(x_i) = I(x_i) - I(x_i \mid y_j) \]

- We interpret \( I(x_i \mid y_j) \) as the self-information about the event \( X=x_i \) after having observed the event \( Y=y_j \).

- The mutual information between a pair of events can be either positive or negative, or zero since both \( I(x_i \mid y_j) \) and \( I(x_i) \) are greater than or equal to zero.
3.2.1 Average Mutual Information and Entropy

- **Average mutual information** between $X$ and $Y$:
  \[
  I(X;Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) I(x_i; y_j)
  \]

  \[
  = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i) P(y_j)}
  \]

  - $I(X;Y)=0$ when $X$ and $Y$ are statistically independent.
  - $I(X;Y) \geq 0$. (Problem 3-4)

- **Average self-information** $H(X)$:
  \[
  H(X) = \sum_{i=1}^{n} P(x_i) I(x_i)
  \]

  - When $X$ represents the alphabet of possible output letters from a source, $H(X)$ represents the average self-information per source letter, and it is called the *entropy*. 
In the special case, in which the letter from the source are equally probable, \( P(x_i) = 1/n \), we have:

\[
H(X) = -\sum_{i=1}^{n} \frac{1}{n} \log \frac{1}{n} = \log n
\]

In general, \( H(X) \leq \log n \) for any given set of source letter probabilities.

In other words, the entropy of a discrete source is a maximum when the output letters are equally probable.
Example 3.2-3: Consider a source that emits a sequence of statistically independent letters, where each output letter is either 0 with probability $q$ or 1 with probability $1-q$.

The entropy of this source is:

$$H(X) = H(q) = -q \log q - (1-q) \log (1-q)$$

Maximum value of the entropy function occurs at $q=0.5$ where $H(0.5)=1$. 

\[ H(q) \text{ (Binary Entropy Function)} \]
The **average conditional self-information** is called the **conditional entropy** and is defined as:

\[
H(X \mid Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \log \frac{1}{P(x_i \mid y_j)}
\]

**\(H(X \mid Y)\)** is the information or uncertainty in \(X\) after \(Y\) is observed.

\[
I(X; Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i) P(y_j)}
\]

\[
= \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \left[ \log \left\{ P(x_i \mid y_j) P(y_j) \right\} - \log P(x_i) - \log P(y_j) \right]
\]

\[
= \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \left[ \log P(x_i \mid y_j) - \log P(x_i) \right]
\]

\[
= - \sum_{i=1}^{n} P(x_i) \log P(x_i) + \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \left[ \log P(x_i \mid y_j) \right]
\]

\[
= H(X) - H(X \mid Y)
\]
Since $I(X;Y) \geq 0$, it follows that $H(X) \geq H(X|Y)$, with equality if and only if $X$ and $Y$ are statistically independent.

$H(X|Y)$ can be interpreted as the average amount of (conditional self-information) uncertainty in $X$ after we observe $Y$.

$H(X)$ can be interpreted as the average amount of uncertainty (self-information) prior to the observation.

$I(X;Y)$ is the average amount of (mutual information) uncertainty provided about the set $X$ by the observation of the set $Y$.

Since $H(X) \geq H(X|Y)$, it is clear that conditioning on the observation $Y$ does not increase the entropy.
Example 3.2-4: Consider Example 3.2-2 for the case of $p_0 = p_1 = p$. Let $P(X=0) = q$ and $P(X=1) = 1-q$.

The entropy is:

$$H(X) = H(q) = -q \log q - (1-q) \log (1-q)$$
As in the proceeding example, when the conditional entropy $H(X|Y)$ is viewed in terms of a channel whose input is $X$ and whose output is $Y$, $H(X|Y)$ is called the **equivocation** and is interpreted as the **amount of average uncertainty remaining in $X$ after observation of $Y$.**
3.2.1 Average Mutual Information and Entropy

Entropy for two or more random variables:

\[ H(X_1X_2...X_K) = -\sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} ... \sum_{j_k=1}^{n_k} P(x_{j_1}x_{j_2}...x_{j_k}) \log P(x_{j_1}x_{j_2}...x_{j_k}) \]

since \( P(x_1x_2...x_k) = P(x_1)P(x_2|x_1)P(x_3|x_1x_2)...P(x_k|x_1x_2...x_{k-1}) \)

\[ H(X_1X_2X_3...X_K) = H(X_1) + H(X_2|X_1) + H(X_3|X_1X_2) \]
\[ + ... + H(X_k|X_1X_2...X_{k-1}) \]
\[ = \sum_{i=1}^{k} H(X_i|X_1X_2...X_{i-1}) \]

Since \( H(X) \geq H(X|Y) \Rightarrow H(X_1X_2...X_k) \leq \sum_{m=1}^{k} H(X_m) \)

where \( X = X_m \) and \( Y = X_1X_2...X_{m-1} \)
3.2.2 Information Measures for Continuous Random Variables

- If $X$ and $Y$ are random variables with joint PDF $p(x,y)$ and marginal PDFs $p(x)$ and $p(y)$, the average mutual information between $X$ and $Y$ is defined as:

$$I(X;Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x)p(y|x) \log \frac{p(y|x)p(x)}{p(x)p(y)} \, dx \, dy$$

- Although the definition of the average mutual information carriers over to continuous random variables, the concept of self-information does not. The problem is that a continuous random variable requires an infinite number of binary digits to represent it exactly. Hence, its self-information is infinite and, therefore, its entropy is also infinite.
3.2.2 Information Measures for Continuous Random Variables

- **Differential entropy** of the continuous random variables $X$ is defined as:

$$H(X) = -\int_{-\infty}^{\infty} p(x) \log p(x) \, dx$$

Note that this quantity does not have the physical meaning of self-information, although it may appear to be a natural extension of the definition of entropy for a discrete random variable.

- **Average conditional entropy** of $X$ given $Y$ is defined as:

$$H(X \mid Y) = -\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x, y) \log p(x \mid y) \, dx \, dy$$

- **Average mutual information** may be expressed as:

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$
3.2.2 Information Measures for Continuous Random Variables

- Suppose $X$ is discrete and has possible outcomes $x_i$, $i=1,2,\ldots,n$, and $Y$ is continuous and is described by its marginal PDF $p(y)$.
  - When $X$ and $Y$ are statistically dependent, we may express $p(y)$ as:
    
    $$ p(y) = \sum_{i=1}^{n} p(y| x_i) P(x_i) $$
  
  - The *mutual information* provided about the event $X=x_i$ by the occurrence of the event $Y=y$ is:
    
    $$ I(x_i; y) = \frac{p(y| x_i) P(x_i)}{p(y) P(x_i)} = \log \frac{p(y| x_i)}{p(y)} $$
  
  - The *average mutual information* between $X$ and $Y$ is:
    
    $$ I(X; Y) = \sum_{i=1}^{n} \int_{-\infty}^{\infty} p(y| x_i) P(x_i) \log \frac{p(y| x_i)}{p(y)} \, dy $$

Example 3.2-5. Let $X$ be a discrete random variable with two equally probable outcomes $x_1=A$ and $x_2=-A$.

Let the conditional PDFs $p(y|x_i)$, $i=1,2$, be Gaussian with mean $x_i$ and variance $\sigma^2$.

$$p(y \mid A) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-A)^2}{2\sigma^2}} \quad p(y \mid -A) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y+A)^2}{2\sigma^2}}$$

The average mutual information is:

$$I(X;Y) = \frac{1}{2} \int_{-\infty}^{\infty} \left[ p(y \mid A) \log \frac{p(y \mid A)}{p(y)} + p(y \mid -A) \log \frac{p(y \mid -A)}{p(y)} \right] dy$$

where $p(y) = \frac{1}{2} \left[ p(y \mid A) + p(y \mid -A) \right]$
3.3 Coding for Discrete Sources

- Consider the process of encoding the output of a source, i.e., the process of representing the source output by a sequence of binary digits.
- A measure of the efficiency of a source-encoding method can be obtained by comparing the average number of binary digits per output letter from the source to the entropy $H(X)$.
- The discrete memoryless source (DMS) is by far the simplest model that can be devised for a physical model. Few physical sources closely fit this idealized mathematical model.
- It is always more efficient to encode blocks of symbols instead of encoding each symbol separately.
- 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.
3.3.1 Coding for Discrete Memoryless Sources

- Suppose that a DMS produces an output letter or symbol every $\tau_s$ seconds.
- Each symbol is selected from a finite alphabet of symbols $x_i$, $i=1,2,\ldots,L$, occurring with probabilities $P(x_i)$, $i=1,2,\ldots,L$.
- The entropy of the DMS in bits per source symbol is:
  \[
  H(X) = - \sum_{i=1}^{L} P(x_i) \log_2 P(x_i) \leq \log_2 L
  \]
  The equality holds when the symbols are equally probable.
- The average number of bits per source symbol is $H(X)$.
- The source rate in bits/s is defined as $H(X)/\tau_s$. 

Problem 3-5.
3.3.1 Coding for Discrete Memoryless Sources

- Fixed-length code words
  - 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 is:
    \[
    R = \log_2 L \quad \text{when } L \text{ is a power of 2.}
    \]
    \[
    R = \left\lfloor \log_2 L \right\rfloor + 1 \quad \text{when } L \text{ is not a power of 2.}
    \]
    \[
    \left\lfloor x \right\rfloor \text{ denotes the largest integer less than } x.
    \]
  - The code rate $R$ in bits per symbol is $R$.
  - Since $H(X) \leq \log_2 L$, it follows that $R \geq H(X)$. 

3.3.1 Coding for Discrete Memoryless Sources

- **Fixed-length code words**
  - The *efficiency* of the encoding for the DMS is defined as the ratio $H(X)/R$.
  - When $L$ is a power of 2 and the source letters are equally probable, $R = H(X)$.
  - If $L$ is not a power of 2, but the source symbols are equally probable, $R$ differs from $H(X)$ by at most 1 bit per symbol.
  - When $\log_2 L \gg 1$, the efficiency of this encoding scheme is high.
  - 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 achieve this, we need $L^J$ unique code words.
3.3.1 Coding for Discrete Memoryless Sources

Fixed-length code words

- By using sequences of $N$ binary digits, we have $2^N$ possible code words. $N \geq J \log_2 L$. The minimum integer value of $N$ required is $N = \left\lceil J \log_2 L \right\rceil + 1$

- The average number of bits per source symbol is $N/J = R$ and the inefficiency has been reduced by approximately a factor of $1/J$ relative to the symbol-by-symbol encoding.

- By making $J$ sufficiently large, the efficiency of the encoding procedure, measured by the ratio $H(X)/R = JH(X)/N$, can be made as close to unity as desired.

- The above mentioned methods introduce no distortion since the encoding of source symbols or block of symbols into code words is unique. This is called noiseless.
3.3.1 Coding for Discrete Memoryless Sources

- **Block coding failure** (or distortion), with probability of $P_e$, occurs when the encoding process is not unique.
- Source coding theorem I: (by Shannon)
  - 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 $J$ is sufficiently large and
    \[
    R \equiv \frac{N}{J} \geq H(X) + \varepsilon
    \]
  - Conversely, if $R \leq H(X) - \varepsilon$, $P_e$ becomes arbitrarily close to 1 as $J$ is sufficiently large.
3.3.1 Coding for Discrete Memoryless Sources

- **Variable-length code words**
  - When the source symbols are not equally probable, a more efficient encoding method is to use variable-length code words.
  - In the Morse code, the letters that occur more frequently are assigned short code words and those that occur infrequently are assigned long code words.
  - *Entropy coding* devises a method for selecting and assigning the code words to source letters.
3.3.1 Coding for Discrete Memoryless Sources

Variable-length code words

<table>
<thead>
<tr>
<th>Letter</th>
<th>$P(a_k)$</th>
<th>Code I</th>
<th>Code II</th>
<th>Code III</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>1/2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a_2$</td>
<td>1/4</td>
<td>00</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>$a_3$</td>
<td>1/8</td>
<td>01</td>
<td>110</td>
<td>011</td>
</tr>
<tr>
<td>$a_4$</td>
<td>1/8</td>
<td>10</td>
<td>111</td>
<td>111</td>
</tr>
</tbody>
</table>

- Code I is not uniquely decodable. (Example: 1001)
- Code II is \textit{uniquely decodable and instantaneously decodable}.
  - Digit 0 indicates the end of a code word and no code word is longer than three binary digits.
  - \textit{Prefix condition}: no code word of length $l<k$ that is identical to the first $l$ binary digits of another code word of length $k>l$. 
3.3.1 Coding for Discrete Memoryless Sources

- Variable-length code words
  - Code III has a tree structure:
  
  ![Tree Diagram](image)

- The code is uniquely decodable.
- The code is not instantaneously decodable.
- This code does not satisfy the prefix condition.

**Objective:** devise a systematic procedure for constructing uniquely decodable variable-length codes that minimizes:

\[
\bar{R} = \sum_{k=1}^{L} n_k P\left(a_k\right)
\]
3.3.1 Coding for Discrete Memoryless Sources

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

- **Proof of sufficient condition:**
  
  Consider a code tree that is embedded in the full tree of $2^n \ (n=n_L)$ nodes.

  ![Code Tree Diagram]
3.3.1 Coding for Discrete Memoryless Sources

- Kraft inequality
  - Proof of sufficient condition (cont.)
    - Let’s select any node of order \( n_1 \) as the first code word \( C_1 \). This choice eliminates \( 2^{n-n_1} \) terminal nodes (or the fraction \( 2^{-n_1} \) of the \( 2^n \) terminal nodes).
    - From the remaining available nodes of order \( n_2 \), we select one node for the second code word \( C_2 \). This choice eliminates \( 2^{n-n_2} \) terminal nodes.
    - This process continues until the last code word is assigned at terminal node \( L \).
    - At the node \( j<L \), the fraction of the number of terminal nodes eliminated is:
      \[
      \sum_{k=1}^{j} 2^{-n_k} < \sum_{k=1}^{L} 2^{-n_k} \leq 1
      \]
      - At node \( j<L \), there is always a node \( k>j \) available to be assigned to the next code word. Thus, we have constructed a code tree that is embedded in the full tree. Q.E.D.
3.3.1 Coding for Discrete Memoryless Sources

- Kraft inequality
  - Proof of necessary condition
    - In code tree of order $n=n_L$, the number of terminal nodes eliminated from the total number of $2^n$ terminal nodes is:
    \[
    \sum_{k=1}^{L} 2^{n-n_k} \leq 2^n \quad \Rightarrow \quad \sum_{k=1}^{L} 2^{-n_k} \leq 1
    \]

- 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 occurrence $p_k$, $1 \leq k \leq L$. It is possible to construct a code that satisfies the prefix condition and has an average length $\bar{R}$ that satisfies the inequalities:
    \[
    H(X) \leq \bar{R} < H(X) + 1
    \]
3.3.1 Coding for Discrete Memoryless Sources

❖ Source coding theorem II (cont.)

❖ Proof of lower bound:

\[ H(X) - \bar{R} = \sum_{k=1}^{L} p_k \log_2 \frac{1}{p_k} - \sum_{k=1}^{L} p_k n_k = \sum_{k=1}^{L} p_k \log_2 \frac{2^{-n_k}}{p_k} \]

\[ = \sum_{k=1}^{L} p_k \left( \ln \frac{2^{-n_k}}{p_k} / \ln 2 \right) = \sum_{k=1}^{L} p_k \left( \ln \frac{2^{-n_k}}{p_k} \cdot \log_2 e \right) \]

since \( \ln x \leq x - 1 \), we have:

\[ H(X) - \bar{R} \leq (\log_2 e) \sum_{k=1}^{L} p_k \left( \frac{2^{-n_k}}{p_k} - 1 \right) \leq (\log_2 e) \left( \sum_{k=1}^{L} 2^{-n_k} - 1 \right) \leq 0 \]

Equality holds if and only if \( p_k = 2^{-n_k} \) for \( 1 \leq k \leq L \).

Note that \( \log_a b = \log b / \log a \).

Kraft inequality.
3.3.1 Coding for Discrete Memoryless Sources

- Source coding theorem II (cont.)
  - Proof of upper bound:
    - The upper bound may be established under the constraint that \( n_k, 1 \leq k \leq L \), are integers, by selecting the \( \{n_k\} \) such that \( 2^{-n_k} \leq p_k < 2^{-n_k+1} \).
    - If the terms \( p_k \geq 2^{-n_k} \) are summed over \( 1 \leq k \leq L \), we obtain the Kraft inequality, for which we have demonstrated that there exists a code that satisfies the prefix condition.
    - On the other hand, if we take the logarithm of \( p_k < 2^{-n_k+1} \), we obtain \( \log_2 p_k < -n_k + 1 \) or \( n_k < 1 - \log_2 p_k \).
    - If we multiply both sides by \( p_k \) and sum over \( 1 \leq k \leq L \), we obtain the desired upper bound.
3.3.1 Coding for Discrete Memoryless Sources

Huffman coding algorithm

1. The source symbols are listed in order of decreasing probability. The two source symbols of lowest probability are assigned a 0 and a 1.

2. These two source symbols are regarded as being combined into a new source symbol with probability equal to the sum of the two original probabilities. The probability of the new symbol is placed in the list in accordance with its value.

3. The procedure is repeated until we are left with a final list of source statistics of only two for which a 0 and a 1 are assigned.

4. The code for each (original) source symbol is found by working backward and tracing the sequence of 0s and 1s assigned to that symbol as well as its successors.
3.3.1 Coding for Discrete Memoryless Sources

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
<th>Code Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0.4</td>
<td>00</td>
</tr>
<tr>
<td>S1</td>
<td>0.2</td>
<td>10</td>
</tr>
<tr>
<td>S2</td>
<td>0.2</td>
<td>11</td>
</tr>
<tr>
<td>S3</td>
<td>0.1</td>
<td>010</td>
</tr>
<tr>
<td>S4</td>
<td>0.1</td>
<td>011</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Stage 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>S1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>S2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>
### 3.3.1 Coding for Discrete Memoryless Sources

**Huffman coding algorithm**

**Example 3.3-1**

<table>
<thead>
<tr>
<th>Letter</th>
<th>Probability</th>
<th>Self-information</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.35</td>
<td>1.5146</td>
<td>00</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.30</td>
<td>1.7370</td>
<td>01</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0.20</td>
<td>2.3219</td>
<td>10</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0.10</td>
<td>3.3219</td>
<td>110</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0.04</td>
<td>4.6439</td>
<td>1110</td>
</tr>
<tr>
<td>$x_6$</td>
<td>0.005</td>
<td>7.6439</td>
<td>11110</td>
</tr>
<tr>
<td>$x_7$</td>
<td>0.005</td>
<td>7.6439</td>
<td>11111</td>
</tr>
</tbody>
</table>

$H(X) = 2.11 \quad \hat{R} = 2.21$
3.3.1 Coding for Discrete Memoryless Sources

- Huffman coding algorithm
- Example 3.3-1 (cont.)

![Huffman Coding Diagram](image)

<table>
<thead>
<tr>
<th>Letter</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>0</td>
</tr>
<tr>
<td>X₂</td>
<td>10</td>
</tr>
<tr>
<td>X₃</td>
<td>110</td>
</tr>
<tr>
<td>X₄</td>
<td>1110</td>
</tr>
<tr>
<td>X₅</td>
<td>11100</td>
</tr>
<tr>
<td>X₆</td>
<td>111100</td>
</tr>
<tr>
<td>X₇</td>
<td>111111</td>
</tr>
</tbody>
</table>

\[ \bar{R} = 2.21 \]
3.3.1 Coding for Discrete Memoryless Sources

- Huffman coding algorithm
- Example 3.3-2

![Huffman tree and code table]

<table>
<thead>
<tr>
<th>Letter</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1)</td>
<td>00</td>
</tr>
<tr>
<td>(x_2)</td>
<td>010</td>
</tr>
<tr>
<td>(x_3)</td>
<td>011</td>
</tr>
<tr>
<td>(x_4)</td>
<td>100</td>
</tr>
<tr>
<td>(x_5)</td>
<td>101</td>
</tr>
<tr>
<td>(x_6)</td>
<td>110</td>
</tr>
<tr>
<td>(x_7)</td>
<td>1110</td>
</tr>
<tr>
<td>(x_8)</td>
<td>1111</td>
</tr>
</tbody>
</table>

\[H(X) = 2.63 \quad \bar{R} = 2.70\]
Huffman coding algorithm

This algorithm 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, which allows the received sequence to be uniquely and instantaneously decodable.

Huffman encoding process is not unique.

Code words for different Huffman encoding process can have different lengths. However, the average code-word length is the same.

When a combined symbol is moved as high as possible, the resulting Huffman code has a significantly smaller variance than when it is moved as low as possible.
Huffman coding algorithm

The variable-length encoding (Huffman) algorithm described in the above mentioned examples generates a prefix code having an $\bar{R}$ that satisfies:

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

A more efficient procedure is to encode blocks of $J$ symbols at a time. In such a case, the bounds of source coding theorem II become:

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

$\bar{R}$ can be made as close to $H(X)$ as desired by selecting $J$ sufficiently large.

To design a Huffman code for a DMS, we need to know the probabilities of occurrence of all the source letters.
3.3.1 Coding for Discrete Memoryless Sources

- Huffman coding algorithm
- Example 3.3-3

<table>
<thead>
<tr>
<th>Letter</th>
<th>Probability</th>
<th>Self-information</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.45</td>
<td>1.156</td>
<td>1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.35</td>
<td>1.520</td>
<td>00</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0.20</td>
<td>2.330</td>
<td>01</td>
</tr>
</tbody>
</table>

$H(X) = 1.518 \text{ bits/letter}$

$\bar{R_1} = 1.55 \text{ bits/letter}$

Efficiency = 97.9\%
3.3.1 Coding for Discrete Memoryless Sources

- Huffman coding algorithm
- Example 3.3-3 (cont.)

<table>
<thead>
<tr>
<th>Letter pair</th>
<th>Probability</th>
<th>Self-information</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 x_1$</td>
<td>0.2025</td>
<td>2.312</td>
<td>10</td>
</tr>
<tr>
<td>$x_1 x_2$</td>
<td>0.1575</td>
<td>2.676</td>
<td>001</td>
</tr>
<tr>
<td>$x_2 x_1$</td>
<td>0.1575</td>
<td>2.676</td>
<td>010</td>
</tr>
<tr>
<td>$x_2 x_2$</td>
<td>0.1225</td>
<td>3.039</td>
<td>011</td>
</tr>
<tr>
<td>$x_1 x_3$</td>
<td>0.09</td>
<td>3.486</td>
<td>111</td>
</tr>
<tr>
<td>$x_3 x_1$</td>
<td>0.09</td>
<td>3.486</td>
<td>0000</td>
</tr>
<tr>
<td>$x_2 x_3$</td>
<td>0.07</td>
<td>3.850</td>
<td>0001</td>
</tr>
<tr>
<td>$x_3 x_2$</td>
<td>0.07</td>
<td>3.850</td>
<td>1100</td>
</tr>
<tr>
<td>$x_3 x_3$</td>
<td>0.04</td>
<td>4.660</td>
<td>1101</td>
</tr>
</tbody>
</table>

$2H(X) = 3.036 \text{ bits/letter pair}$

$\bar{R}_2 = 3.0675 \text{ bits/letter pair}$

$\frac{1}{2} \bar{R}_2 = 1.534 \text{ bits/letter}$

Efficiency = 99.0%
3.3.2 Discrete Stationary Sources

- We consider discrete sources for which the sequence of output letters is statistically stationary (letters are statistically dependent).
- The *entropy* of a block of random variables \(X_1X_2\ldots X_k\) is:
  \[
  H(X_1X_2\ldots X_K) = \sum_{i=1}^{k} H(X_i \mid X_1X_2\ldots X_{i-1})
  \]
  \(H(X_i\mid X_1X_2\ldots X_{i-1})\) is the *conditional entropy* of the \(i\)th symbol from the source given the previous \(i-1\) symbols.
- The *entropy per letter* for the \(k\)-symbol block is defined as
  \[
  H_k(X) = \frac{1}{k} H(X_1X_2\ldots X_k)
  \]
- *Information content* of a stationary source is defined as the entropy per letter in the limit as \(k \to \infty\).
  \[
  H_{\infty}(X) \equiv \lim_{k \to \infty} H_k(X) = \lim_{k \to \infty} \frac{1}{k} H(X_1X_2\ldots X_k)
  \]

From 3.2-13 and 3.2-15.
### 3.3.2 Discrete Stationary Sources

- **The entropy per letter** from the source can be defined in terms of the conditional entropy $H(X_k | X_1X_2…X_{k-1})$ in the limit as $k$ approaches infinity.

$$H_\infty (X) = \lim_{k \to \infty} H \left( X_k \mid X_1X_2…X_{k-1} \right)$$

- Existence of the above equation can be found in page 100-101.

- For a discrete stationary source that emits $J$ letters with $H_J(X)$ as the entropy per letter.

$$H \left( X_1…X_J \right) \leq \bar{R}_J < H \left( X_1…X_J \right) + 1$$

$$H_J \left( X \right) \leq \bar{R} = \frac{\bar{R}_J}{J} < H_J \left( X \right) + \frac{1}{J}$$

- In the limit as $J \to \infty$, we have:

$$H_\infty (X) \leq \bar{R} < H_\infty (X) + \varepsilon$$
3.3.3 The Lempel-Ziv Algorithm

- For Huffman Coding, except for the estimation of the marginal probabilities \( \{p_k\} \), corresponding to the frequency of occurrence of the individual source output letters, the computational complexity involved in estimating joint probabilities is extremely high.

- The application of the Huffman coding method to source coding for many real sources with memory is generally impractical.

- The *Lempel-Ziv source coding algorithm* is designed to be independent of the source statistics.

- It belongs to the class of *universal source coding algorithms*.

- It is a variable-to-fixed-length algorithm.
3.3.3 The Lempel-Ziv Algorithm

Operation of Lempel-Ziv algorithm

1. The sequence at the output of the discrete source is parsed into variable-length blocks, which are called *phrases*.

2. A new phrase is introduced every time a block of letters from the source differs from some previous phrase in the last letter.

3. The phrases are listed in a dictionary, which stores the location of the existing phrases.

4. In encoding a new phrase, we simply specify the location of the existing phrase in the dictionary and append the new letter.

- 1010110100111110101000011100011011
- 1,0,10,11,01,00,100,111,010,1000,011,001,110,101,10001, 1011
### 3.3.3 The Lempel-Ziv Algorithm

**Operation of Lempel-Ziv algorithm (cont.)**

To encode the phrases, we construct a dictionary:

<table>
<thead>
<tr>
<th>Dictionary location</th>
<th>Dictionary contents</th>
<th>Code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>01</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>00</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>111</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>010</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>1000</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>011</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>001</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>110</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>101</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>10001</td>
</tr>
<tr>
<td>16</td>
<td>1011</td>
<td>1111</td>
</tr>
</tbody>
</table>
3.3.3 The Lempel-Ziv Algorithm

Operation of Lempel-Ziv algorithm (cont.)

5. The code words are determined by listing the dictionary location (in binary form) of the previous phrase that matches the new phrase in all but the last location.

6. The new output letter is appended to the dictionary location of the previous phrase.

7. The location 0000 is used to encode a phrase that has not appeared previously.

8. The source decoder for the code constructs an identical copy of the dictionary at the receiving end of the communication system and decodes the received sequence in step with the transmitted data sequence.
3.3.3 The Lempel-Ziv Algorithm

- Operation of Lempel-Ziv algorithm (cont.)
  - As the sequence is increased in length, the encoding procedure becomes more efficient and results in a compressed sequence at the output of the source.
  - No matter how large the table is, it will eventually overflow.
  - To solve the overflow problem, the source encoder and decoder must use an identical procedure to remove phrases from the dictionaries that are not useful and substitute new phrases in their place.
  - Lempel-Ziv algorithm is widely used in the compression of computer files.
    - E.g. “compress” and “uncompress” utilities under the UNIX© OS.
3.4 Coding for Analog Sources - Optimum Quantization

- 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 take at the Nyquist rate.
- The samples are then quantized in amplitude and encoded.
- Quantization of the amplitudes of the sampled signal results in data compression, but it also introduces some distortion of the waveform or a loss of signal fidelity.
3.4.1 Rate-Distortion Function

- Distortion is introduced when the samples from the information source are quantized to a fixed number of bits.
- Squared-error distortion:
  \[ d(x_k, \tilde{x}_k) = (x_k - \tilde{x}_k)^2 \]
- Distortion of the general form:
  \[ d(x_k, \tilde{x}_k) = |x - \tilde{x}_k|^p \]
- The distortion between a sequence of \( n \) samples \( X_n \) and the corresponding \( n \) quantized values \( \widetilde{X}_n \) is the average over the \( n \) source output samples, i.e.,
  \[ d(X_n, \widetilde{X}_n) = \frac{1}{n} \sum_{k=1}^{n} d(x_k, \tilde{x}_k) \]
3.4.1 Rate-Distortion Function

- $d\left(X_n, \tilde{X}_n\right)$ is a random variable and its expected value is defined as the distortion $D$, i.e.,

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

- The minimum rate in bits per source output that is required to represent the output $X$ 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:

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

where $I\left(X; \tilde{X}\right)$ is the average mutual information between $X$ and $\tilde{X}$. Note that $R(D)$ decreases as $D$ increases.
3.4.1 Rate-Distortion Function

- Theorem: Rate-distortion function for a memoryless Gaussian source (Shannon 1959)
- 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:

\[
R_g (D) = \begin{cases} 
\frac{1}{2} \log_2 \left( \frac{\sigma_x^2}{D} \right) & (0 \leq D \leq \sigma_x^2) \\
0 & (D > \sigma_x^2)
\end{cases}
\]

where \( \sigma_x^2 \) is the variance of the Gaussian source output.
3.4.1 Rate-Distortion Function

- Rate-distortion function for a memoryless Gaussian source
  - No information need be transmitted when the distortion $D \geq \sigma_x^2$.
  - $D = \sigma_x^2$ 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.
Theorem: Source coding with a distortion measure (Shannon 1959)

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 sufficient to reconstruct the source output with an average distortion that is arbitrarily close to $D$.

It is clear 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.

**Distortion-rate function** for the discrete-time, memoryless Gaussian is (from Equation 3.4-6):

$$D_g(R) = 2^{-2^R} \sigma_x^2 \quad \text{or} \quad 10 \log_{10} D_g(R) = -6R + 10 \log_{10} \sigma_x^2 \quad \text{[dB]}$$
3.4.1 Rate-Distortion Function

Theorem: Upper Bound on $R(D)$

The rate-distortion function of a memoryless, continuous-amplitude source with zero mean and finite variance $\sigma_x^2$ with respect to the mean-square-error distortion measure is upper-bounded as:

$$R(D) \leq \frac{1}{2} \log_2 \frac{\sigma_x^2}{D}, \quad 0 \leq D \leq \sigma_x^2$$

The theorem implies that the Gaussian source requires the maximum rate among all other sources for a specified level of mean-square-error distortion.

The rate distortion $R(D)$ of any continuous-amplitude, memoryless source with zero mean and finite variance $\sigma_x^2$ satisfies the condition $R(D) \leq R_g(D)$. 
The distortion-rate function of the same source satisfies the condition:
\[ D(R) \leq D_g(R) = 2^{-2R} \sigma_x^2 \]

The lower bound on the rate-distortion function exists and is called the *Shannon lower bound* for a mean-square-error distortion measure:
\[ R^*(D) = H(X) - \frac{1}{2} \log_2 2\pi e D \]
where \( H(X) \) is the differential entropy of the continuous-amplitude, memoryless source. The distortion-rate function is:
\[ D^*(R) = \frac{1}{2\pi e} 2^{-2[R - H(X)]} \]

For any continuous-amplitude, memoryless source:
\[ R^*(D) \leq R(D) \leq R_g(D) \quad \text{and} \quad D^*(R) \leq D(R) \leq D_g(R) \]
### 3.4.1 Rate-Distortion Function

- The differential entropy of the memoryless Gaussian source is:
  \[ H_g(X) = \frac{1}{2} \log_2 2\pi e \sigma_x^2 \Rightarrow R^*(D) = R_g(D) \]

- By setting \( \sigma_x^2 = 1 \), we obtain from Equation 3.4-12:
  \[ 10 \log_{10} D^*(R) = -6R - 6\left[ H_g(X) - H(X) \right] \]

  \[ 10 \log_{10} \frac{D_g(R)}{D^*(R)} = 6\left[ H_g(X) - H(X) \right] dB \]

    \[ = 6\left[ R_g(D) - R^*(D) \right] dB \]

- The differential entropy \( H(X) \) is upper-bounded by \( H_g(X) \) (shown by Shannon 1948).
### 3.4.1 Rate-Distortion Function

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

<table>
<thead>
<tr>
<th>PDF</th>
<th>( p(x) )</th>
<th>( H(x) )</th>
<th>( R_g(D) - R^*(D) ) (bits/sample)</th>
<th>( D_g(R) - D^*(R) ) (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>( \frac{1}{\sqrt{2\pi\sigma_x}} e^{-x^2/2\sigma_x^2} )</td>
<td>( \frac{1}{2} \log_2 \left( 2\pi e\sigma_x^2 \right) )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Uniform</td>
<td>( \frac{1}{2\sqrt{3}\sigma_x} ), (</td>
<td>x</td>
<td>\leq \sqrt{3}\sigma_x )</td>
<td>( \frac{1}{2} \log_2 \left( 12\sigma_x^2 \right) )</td>
</tr>
<tr>
<td>Laplacian</td>
<td>( \frac{1}{\sqrt{2\sigma_x}} e^{-\sqrt{2}</td>
<td>x</td>
<td>/\sigma_x} )</td>
<td>( \frac{1}{2} \log_2 \left( 2e^2\sigma_x^2 \right) )</td>
</tr>
<tr>
<td>Gamma</td>
<td>( \frac{4\sqrt{3}}{\sqrt{8\pi\sigma_x}} \frac{1}{</td>
<td>x</td>
<td>} e^{-\sqrt{3}</td>
<td>x</td>
</tr>
</tbody>
</table>
3.4.1 Rate-Distortion Function

Consider a band-limited Gaussian source with spectral density:

\[
\Phi(f) = \begin{cases} 
\frac{\sigma_x^2}{2W} & (|f| \leq W) \\
0 & (|f| > W)
\end{cases}
\]

When the output of this source is sampled at the Nyquist rate, the samples are uncorrelated and, since the source is Gaussian, they are also statistically independent.

The equivalent discrete-time Gaussian source is memoryless.

The rate-distortion function in bits/s is:

\[
R_g(D) = W \log_2 \frac{\sigma_x^2}{D} \quad 0 \leq D \leq \sigma_x^2
\]

The distortion-rate function is:

\[
D_g(R) = 2^{-\frac{R}{W}} \sigma_x^2
\]
3.4.2 Scalar Quantization

- In source encoding, the quantizer can be optimized if we know the PDF of the signal amplitude at the input to the quantizer.
- Suppose that the sequence \( \{x_n\} \) at the input to the quantizer has a PDF \( p(x) \) and let \( L=2^R \) be the desired number of levels. Design the optimum scalar quantizer that minimizes some function of the quantization error \( q = \tilde{x} - x \), where \( \tilde{x} \) is the quantized value of \( x \).
- \( f(\tilde{x} - x) \) denotes the desired function of the error. Then, the distortion resulting from quantization of the signal amplitude is:

\[
D = \int_{-\infty}^{\infty} f(\tilde{x} - x)p(x) \, dx
\]

- **Lloyd-Max quantizer**: an optimum quantizer that minimizes \( D \) by optimally selecting the output levels and the corresponding input range of each output level.
For a uniform quantizer, the output levels are specified as $x_k = (2k-1)\Delta/2$, corresponding to an input signal amplitude in the range $(k-1)\Delta \leq x < k\Delta$, where $\Delta$ is the step size.

When the uniform quantizer is symmetric with an even number of levels, the average distortion may be expressed as:

$$D = \int_{-\infty}^{\infty} f(\tilde{x} - x)p(x)\,dx$$

$$= 2\sum_{k=1}^{L-1} \int_{(k-1)\Delta}^{k\Delta} f\left(\frac{1}{2}(2k-1)\Delta - x\right)p(x)\,dx$$

$$+ 2\int_{(L-1)\Delta}^{\infty} f\left(\frac{1}{2}(L-1)\Delta - x\right)p(x)\,dx$$

The minimization of $D$ is carried out with respect to the step-size.
3.4.2 Scalar Quantization

- By differentiating D with respect to $\Delta$, we obtain:

$$\sum_{k=1}^{L-1} (2k-1) \int_{(k-1)\Delta}^{k\Delta} f\left(\frac{1}{2} (2k-1) \Delta - x\right) p(x) dx$$

$$+ (L-1) \int_{-(L-1)\Delta}^{\infty} f'\left(\frac{1}{2} (L-1) \Delta - x\right) p(x) dx = 0$$

- For mean-square-error criterion, for which $f(x)=x^2$: 
3.4.2 Scalar Quantization

- The distortion can be reduced further by using non-uniform quantizer.

- Let the output level be \( \tilde{x} = \tilde{x}_k \) when the input signal amplitude is in the range \( x_{k-1} \leq x < x_k \) with end points are \( x_0 = -\infty \) and \( x_L = \infty \). By optimally selecting the \( \{\tilde{x}_k\} \) and \( \{x_k\} \), we can minimize the distortion:

\[
D = \sum_{k=1}^{L} \int_{x_{k-1}}^{x_k} f\left(\tilde{x}_k - x\right)p\left(x\right)dx
\]

- Differentiating \( D \) with respect to the \( \{x_k\} \) and \( \{\tilde{x}_k\} \), we obtain:

\[
f\left(\tilde{x}_k - x_k\right) = f\left(\tilde{x}_{k+!} - x_k\right) \quad k = 1, 2, \ldots, L - 1
\]

\[
\int_{x_{k-1}}^{x_k} f'\left(\tilde{x}_k - x\right)p\left(x\right)dx \quad k = 1, 2, \ldots, L
\]
3.4.2 Scalar Quantization

For \( f(x) = x^2 \):
\[
x_k = \frac{1}{2} \left( \tilde{x}_k + \tilde{x}_{k+1} \right) \quad k = 1, 2, \ldots, L - 1
\]
\[
\int_{x_{k-1}}^{x_k} \left( \tilde{x}_k - x \right) p(x) \, dx \quad k = 1, 2, \ldots, L
\]

The difference in the performance of the uniform and non-uniform quantizers is relatively small for small values of \( R \), but it increases as \( R \) increases.

<table>
<thead>
<tr>
<th>Number of output levels</th>
<th>Optimum step size ( \Delta_{opt} )</th>
<th>Minimum MSE ( D_{min} )</th>
<th>10 log ( D_{min} ) (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.596</td>
<td>0.3634</td>
<td>-4.4</td>
</tr>
<tr>
<td>4</td>
<td>0.9957</td>
<td>0.1188</td>
<td>-9.25</td>
</tr>
<tr>
<td>8</td>
<td>0.5860</td>
<td>0.03744</td>
<td>-14.27</td>
</tr>
<tr>
<td>16</td>
<td>0.3352</td>
<td>0.01154</td>
<td>-19.38</td>
</tr>
<tr>
<td>32</td>
<td>0.1881</td>
<td>0.00349</td>
<td>-24.57</td>
</tr>
</tbody>
</table>
3.4.2 Scalar Quantization

Optimum eight-level quantizer for a Gaussian random variable.

<table>
<thead>
<tr>
<th>Level $k$</th>
<th>$x_k$</th>
<th>$\tilde{x}_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.9816</td>
<td>-1.510</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
<td>-0.4528</td>
</tr>
<tr>
<td>3</td>
<td>0.9816</td>
<td>0.4528</td>
</tr>
<tr>
<td>4</td>
<td>$\infty$</td>
<td>1.510</td>
</tr>
</tbody>
</table>

$D_{\text{min}} = 0.1175$

$10 \log D_{\text{min}} = -9.3 \text{ dB}$
### 3.4.2 Scalar Quantization

Optimum eight-level quantizer for a Gaussian random variable.

<table>
<thead>
<tr>
<th>Level $k$</th>
<th>$x_k$</th>
<th>$\tilde{x}_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1.748</td>
<td>-2.152</td>
</tr>
<tr>
<td>2</td>
<td>-1.050</td>
<td>-1.344</td>
</tr>
<tr>
<td>3</td>
<td>-0.5006</td>
<td>-0.7560</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-0.2451</td>
</tr>
<tr>
<td>5</td>
<td>0.5006</td>
<td>0.2451</td>
</tr>
<tr>
<td>6</td>
<td>1.050</td>
<td>0.7560</td>
</tr>
<tr>
<td>7</td>
<td>1.748</td>
<td>1.344</td>
</tr>
<tr>
<td>8</td>
<td>$\infty$</td>
<td>2.152</td>
</tr>
</tbody>
</table>

$D_{\text{min}} = 0.03454$

$10 \log D_{\text{min}} = -14.62 \text{ dB}$
3.4.2 Scalar Quantization

Comparison of optimum uniform and nonuniform quantizers for a Gaussian random variable.

<table>
<thead>
<tr>
<th>$R$ (bits/sample)</th>
<th>$10 \log_{10} D_{\text{min}}$ Uniform (dB)</th>
<th>$10 \log_{10} D_{\text{min}}$ Nonuniform (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>−4.4</td>
<td>−4.4</td>
</tr>
<tr>
<td>2</td>
<td>−9.25</td>
<td>−9.30</td>
</tr>
<tr>
<td>3</td>
<td>−14.27</td>
<td>−14.62</td>
</tr>
<tr>
<td>4</td>
<td>−19.38</td>
<td>−20.22</td>
</tr>
<tr>
<td>5</td>
<td>−24.57</td>
<td>−26.02</td>
</tr>
<tr>
<td>6</td>
<td>−29.83</td>
<td>−31.89</td>
</tr>
<tr>
<td>7</td>
<td>−35.13</td>
<td>−37.81</td>
</tr>
</tbody>
</table>
3.4.2 Scalar Quantization

- Distortion versus rate curves for discrete-time memoryless Gaussian source

![Graph showing distortion-rate curves for scalar quantization.](image)
Since any quantizer reduces a continuous amplitude source into a discrete amplitude source, we may treat the discrete amplitude as letters, say $\tilde{X} = \{\tilde{x}_k, 1 \leq k \leq L\}$ with associated probabilities $\{p_k\}$. If the signal amplitudes are statistically independent, the discrete source is memoryless and its entropy is:

$$H(\tilde{X}) = -\sum_{k=1}^{L} p_k \log_2 p_k$$

<table>
<thead>
<tr>
<th>$\tilde{R}$ (bits/sample)</th>
<th>Entropy (bits/letter)</th>
<th>Distortion $10 \log_{10} D_{\text{min}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>-4.4</td>
</tr>
<tr>
<td>2</td>
<td>1.911</td>
<td>-9.30</td>
</tr>
<tr>
<td>3</td>
<td>2.825</td>
<td>-14.62</td>
</tr>
<tr>
<td>4</td>
<td>3.765</td>
<td>-20.22</td>
</tr>
<tr>
<td>5</td>
<td>4.730</td>
<td>-26.02</td>
</tr>
</tbody>
</table>
Conclusions:

- The quantizer can be optimized when the PDF of the continuous source output is known.
- The optimum quantizer of $L=2^R$ levels results in a minimum distortion of $D(R)$, where $R=\log_2 L$ bits per sample.
- This distortion can be achieved by simply representing each quantized sample by $R$ bits, though more efficient encoding is possible since the discrete source output that results from quantization is characterized by a set of probabilities that can be used to design efficient variable-length codes for the source output (entropy coding).
The joint quantization of a block of signal samples or a block of signal parameters is called *block or vector quantization*. Vector quantization is widely used in speech coding for digital cellular systems. Even if the continuous-amplitude source is memoryless, quantizing vectors performs better than quantizing scalar. If the signal samples or signal parameters are statistically dependent, we can achieve an even greater efficiency.

We have an $n$-dimensional vector $X = [x_1, x_2, \ldots, x_n]$ with real-valued, continuous-amplitude components $\{x_k, 1 \leq k \leq n\}$ and joint PDF $p(x_1, x_2, \ldots, x_n)$. The vector $X$ is quantized into another $n$-dimensional vector $\tilde{X}$ with components $\{\tilde{x}_k, 1 \leq k \leq n\}: \tilde{X} = Q(X)$. 
Vector quantization of blocks of data may be viewed as a pattern recognition problem involving the classification of blocks of data into a discrete number of categories or cells in a way that optimizes some fidelity criterion, such as mean-square-error distortion.

Example of quantization in two-dimensional space.
The average distortion over the set of input vectors $X$ is:

$$D = \sum_{k=1}^{L} P(X \in C_k) E \left[ d \left( X, \tilde{X}_k \right) \middle| X \in C_k \right]$$

$$= \sum_{k=1}^{L} P(X \in C_k) \int_{X \in C_k} d \left( X, \tilde{X}_k \right) p(X) dX$$

We can minimize $D$ by selecting the cells $\{C_k, 1 \leq k \leq L\}$ for a given PDF $p(X)$.

A commonly used distortion measure for vector quantization is the mean square error ($l_2$ norm) defined as:

$$d_2 \left( X, \tilde{X} \right) = \frac{1}{n} \left( X - \tilde{X} \right)' \left( X - \tilde{X} \right) = \frac{1}{n} \sum_{k=1}^{n} \left( x_k - \tilde{x}_k \right)^2$$

Weighted mean square error:

$$d_{2w} \left( X, \tilde{X} \right) = \left( X - \tilde{X} \right)' W \left( X - \tilde{X} \right)$$
3.4.3 Vector Quantization

- $W$ is a positive-definite weighting matrix and is usually selected to be the inverse of the covariance matrix of the input data vector $X$.
- $l_p$ norm is defined as: 
  \[ d_p(X, \tilde{X}) = \frac{1}{n} \sum_{k=1}^{n} \left| x_k - \tilde{x}_k \right|^p \]
- Two conditions for the partitioning of the $n$-dimensional space into $L$ cells so that the average distortion is minimized.
  - Nearest-neighbor selection rule:
    \[ d \left( X, \tilde{X}_k \right) \leq d \left( X, \tilde{X}_j \right), \quad k \neq j, \quad 1 \leq j \leq L \]
  - Each output vector be chosen to minimize the average distortion in cell $C_k$:
    \[ D_k = E \left[ d \left( X, \tilde{X} \right) \mid X \in C_k \right] = \int_{X \in C_k} d \left( X, \tilde{X} \right) p(X) dX \]

The $\tilde{X}_k$ that minimizes $D_k$ is called the centroid of the cell.
In general, we expect the code vectors to be closer together in regions where the joint PDF is large and farther apart in regions where \( p(X) \) is small.

The distortion per dimension is defined as:

\[
d(\bar{X}, \tilde{X}) = \frac{1}{n} \sum_{k=1}^{n} (x_k - \tilde{x}_k)^2
\]

The vectors can be transmitted at an average bit rate of:

\[
R = \frac{H(\tilde{X})}{n} \text{ bits per sample}
\]

\( H(\tilde{X}) \) is the entropy of the quantized source output defined as

\[
H(\tilde{X}) = -\sum_{i=1}^{L} p(\tilde{X}_i) \log_2 p(\tilde{X}_i)
\]
3.4.3 Vector Quantization

- For a given average rate $R$, the minimum achievable distortion $D_n(R)$ is:

$$D_n(R) = \min_{Q(X)} \mathbb{E} \left[ d \left( X, \tilde{X} \right) \right]$$

$$D(R) = \lim_{n \to \infty} D_n(R)$$

- The distortion-rate function can be approached arbitrarily closely by increasing the size $n$ of the vectors.

- The development above is predicated on the assumption that the joint PDF $p(X)$ of the data vector is known. However, the joint PDF $p(X)$ of the data may not be known.

- It is possible to select the quantized output vectors adaptively from a set of training vectors $X(m)$.
3.4.3 Vector Quantization

**K-Means Algorithm**

**Step 1** Initialize by setting the iteration number \( i = 0 \). Choose a set of output vectors \( \tilde{X}_k (0), 1 \leq k \leq L \).

**Step 2** Classify the training vectors \( \{X (m), 1 \leq m \leq M\} \) into the clusters \( \{C_k\} \) by applying the nearest-neighbor rule:

\[
X \in C_k (i) \text{ iff } d \left( X, \tilde{X}_k (i) \right) \leq d \left( X, \tilde{X}_j (i) \right) \text{ for all } k \neq j
\]

**Step 3** Recompute (set \( i \) to \( i + 1 \)) the output vectors of every cluster by computing the centroid:

\[
\tilde{X}_k (i) = \frac{1}{M_k} \sum_{x \in C_k} X (m), \quad 1 \leq k \leq L
\]

of the training vectors that fall in each cluster. Also, compute the resulting average distortion \( D(i) \) at the \( i \)th iteration.

**Step 4** Terminate the test if the change \( D(i - 1) - D(i) \) in the average distortion is relatively small. Otherwise, go to step 2.
3.4.3 Vector Quantization

- The $K$-means algorithm converges to a local minimum.
- By beginning the algorithm with different sets of initial output vectors and each time performing the optimization described in the $K$-mean algorithm, it is possible to find a global optimum.
- Once we have selected the output vectors, we have established what is called a code book.
- If the computation involves evaluating the distance between $X(m)$ and each of the $L$ possible output vectors, the procedure constitutes a full search.
- If we assume that each computation requires $n$ multiplications and additions, the computational requirement for a full search is $C=nL$ multiplication and additions per input vector.
If we select $L$ to be a power of 2, then $\log_2 L$ is the number of bits required to represent each vector. If $R$ denotes the bit rate per sample, we have $nR = \log_2 L$ and the computational cost is $C = n2^{nR}$.

The computational cost associated with a full search can be reduced by constructing the code book based on a binary tree search.

Uniform tree for binary search vector quantization.
3.4.3 Vector Quantization

- The computational cost of the binary tree search is $C=2n\log_2 L=2n^2R$.
- Although the computational cost has been significantly reduced, the memory required to store the (centroid) vectors has increased from $nL$ to approximately $2nL$, due to the fact that we now have to store the vectors at the intermediate nodes in addition to the vectors at the terminal nodes.
- The binary tree search algorithm generates a uniform tree and the resulting code will be sub-optimum in the sense that the code words result in more distortion compared to the code words generated by the method corresponding to a full search.
- A code book resulting in lower distortion is obtained by subdividing the cluster of test vectors having the largest total distortion at each step.
3.4.3 Vector Quantization

- Non-uniform tree for binary search vector quantization
Example 3.4-1: Let $x_1$ and $x_2$ be two random variables with a uniform joint PDF:

$$p(x_1, x_2) = p(X) = \begin{cases} \frac{1}{ab} & (X \in C) \\ 0 & \text{(otherwise)} \end{cases}$$
Example 3.4-1 (cont.)

If we quantize $x_1$ and $x_2$ separately by using uniform intervals of length $\Delta$, the number of levels needed is:

$$L_1 = L_2 = \frac{a + b}{\sqrt{2\Delta}}$$

The number of bits needed for coding the vector $X=[x_1 x_2]$ is:

$$R_x = R_1 + R_2 = \log_2 L_1 + \log_2 L_2$$

$$R_x = \log_2 \left( \frac{(a + b)^2}{2\Delta^2} \right)$$

Scalar quantization of each component is equivalent to vector quantization with the total number of levels:

$$L_x = L_1 L_2 = \frac{(a + b)^2}{2\Delta^2}$$
Example 3.4-1 (cont.)

- This approach is equivalent to covering the large square that encloses the rectangle by square cells, where each cell represents one of the $L_x$ quantized regions.
- This encoding is wasteful and results in an increase of the bit rate.
- If we were to cover only the region for which $p(X) > 0$ with squares having area $\Delta^2$, the total number of levels that will result is the area of the rectangle divided by $\Delta^2$, i.e.,

$$L_x' = \frac{ab}{\Delta^2}$$

- The difference in bit rate between the scalar and vector quantization method is:

$$R_x - R_x' = \log_2 \left( \frac{(a + b)^2}{2ab} \right)$$
Example 3.4-1 (cont.)

- A linear transformation (rotation by 45 degrees) will decorrelate $x_1$ and $x_2$ and render the two random variables statistically independent. Then scalar quantization and vector quantization achieve the same efficiency.
- Vector quantization will always equal or exceed the performance of scalar quantization.
- Vector quantization has been applied to several types of speech encoding methods including both waveform and model-based methods.
Three types of analog source encoding methods

- **Temporal waveform coding**: 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**: based on a mathematical model of the source.
3.5.1 Temporal Waveform Coding

- Pulse-code modulation (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.
  - The quantization process may be modeled mathematically as:
    \[
    \tilde{x}_n = x_n + q_n
    \]
    where \( q_n \) represents the quantization error, which we treat as an additive noise.
3.5.1 Temporal Waveform Coding

- Pulse-code modulation (PCM)
- For a uniform quantizer, the quantization noise is well characterized statistically by the uniform PDF:
  \[ p(q) = \frac{1}{\Delta}, \quad -\frac{1}{2}\Delta \leq q \leq \frac{1}{2}\Delta, \quad \Delta = 2^{-R} \]
- Mean square value of the quantization error is:
  \[ E(q^2) = \frac{1}{12}\Delta^2 = \frac{1}{12} \times 2^{-2R} \]
3.5.1 Temporal Waveform Coding

- Pulse-code modulation (PCM)
  - Many source signals such as speech waveforms have the characteristic that small signal amplitudes occur more frequently than large ones.
  - A better approach is to employ a non-uniform quantizer.
  - A non-uniform quantizer characteristic is usually obtained by passing the signal through a non-linear device that compresses the signal amplitude, followed by a uniform quantizer.
  - In the reconstruction of the signal from the quantized values, the inverse logarithmic relation is used to expand the signal amplitude.
  - The combined compressor-expandor pair is termed as a *comandor*. 
3.5.1 Temporal Waveform Coding

- Pulse-code modulation (PCM)
  - In North America: \( \mu \)-law compression:
    \[
    y = y_{\text{max}} \frac{\log_e[1 + \mu(|x|/x_{\text{max}})]}{\log_e(1 + \mu)} \cdot \text{sgn } x
    \]
    \[
    \text{sgn } x = \begin{cases} 
    +1 & \text{for } x \geq 0 \\
    -1 & \text{for } x < 0 
    \end{cases}
    \]
  - In Europe: A-Law compression:
    \[
    y = \begin{cases} 
    y_{\text{max}} \frac{A(|x|/x_{\text{max}})}{1 + \log_e A} \cdot \text{sgn } x & 0 < \frac{|x|}{x_{\text{max}}} \leq \frac{1}{A} \\
    y_{\text{max}} \frac{\log_e[1 + \log_e[ A(|x|/x_{\text{max}})]]}{1 + \log_e A} \cdot \text{sgn } x & \frac{1}{A} < \frac{|x|}{x_{\text{max}}} \leq 1 
    \end{cases}
    \]
3.5.1 Temporal Waveform Coding

- Pulse-code modulation (PCM)
  - Standard values of $\mu$ is 255 and $A$ is 87.6.

\[ \begin{align*}
\text{(a)} & \quad \mu \text{-law characteristic} \\
\text{(b)} & \quad A \text{-law characteristic}
\end{align*} \]
3.5.1 Temporal Waveform Coding

- **Differential pulse-code modulation (DPCM)**
  - Most source signals sampled at the Nyquist rate or faster exhibit significant correlation between successive samples.
  - An encoding scheme that exploits the redundancy in the samples will result in a lower bit rate for the source output.
  - A relatively simple solution is to encode the differences between successive samples rather than the samples themselves.
  - Since differences between samples are expected to be 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.
3.5.1 Temporal Waveform Coding

- **Differential pulse-code modulation (DPCM)**
  - Let $x_n$ denote the current samples from the source and the predicted value of $x_n$ is defined as:
    $$\hat{x}_n = \sum_{i=1}^{p} a_i x_{n-i}$$
  - The $\{a_i\}$ are selected to minimize some function of the error.
  - A mathematically and practically convenient error function is the mean square error (MSE):
    $$\varepsilon_p = E\left(e_n^2\right) = E\left[\left(x_n - \sum_{i=1}^{p} a_i x_{n-i}\right)^2\right]$$
    $$= E\left(x_n^2\right) - 2 \sum_{i=1}^{p} a_i E\left(x_n x_{n-i}\right) + \sum_{i=1}^{p} \sum_{j=1}^{p} a_i a_j E\left(x_{n-i} x_{n-j}\right)$$
Differential pulse-code modulation (DPCM)

If the source output is wide-sense stationary, we have:

\[ \varepsilon_p = \phi(0) - 2 \sum_{i=1}^{p} a_i \phi(i) + \sum_{i=1}^{p} \sum_{j=1}^{p} a_i a_j \phi(i - j) \]

To minimize \( \varepsilon_p \), we need to solve the linear equations, called the normal equations or the Yule-Walker equations:

\[ \sum_{i=1}^{p} a_i \phi(i - j) = \phi(j), \quad j = 1, 2, \ldots, p \]

When the autocorrelation function \( \phi(n) \) is not known a priori, it may be estimated from the samples \( \{x_n\} \) using the relation:

\[ \hat{\phi}(n) = \frac{1}{N} \sum_{i=1}^{N-n} x_i x_{i+n}, \quad n = 0, 1, 2, \ldots, p \]
3.5.1 Temporal Waveform Coding

Differential pulse-code modulation (DPCM)

Block diagram of a DPCM encoder (a) and decoder (b).

\[ e_n = x_n - \hat{x}_n - \sum_{k=1}^{P} a_k \hat{x}_{n-k} \]
\[ \hat{e}_n = \bar{e}_n - (x_n - \hat{x}_n) = e_n + \hat{x}_n - x_n \]
\[ = \tilde{x}_n - x_n = q_n = \text{quantization error} \]

(a) Encoder

(b) Decoder
3.5.1 Temporal Waveform Coding

Differential pulse-code modulation (DPCM)

- For the DPCM encoder, the output of the predictor is:

\[ \hat{x}_n = \sum_{i=1}^{p} a_i \hat{x}_{n-i} \]

- Input to the quantizer is the difference:

\[ e_n = x_n - \hat{x}_n \]

- Each value of the quantized prediction error is encoded into a sequence of binary digits and transmitted over the channel to the destination.

- At the destination, the same predictor that was used at the transmitting end is synthesized.
3.5.1 Temporal Waveform Coding

- Differential pulse-code modulation (DPCM)
  - The use of feedback around the quantizer ensures that the error in $x_n$ is simply the quantization error $q_n = e_n - e_n$ and that there is no accumulation of previous quantization errors in the implementation of the decoder. That is:

$$q_n = \tilde{e}_n - e_n = \tilde{e}_n - \left(x_n - \hat{x}_n\right) = \hat{x}_n - x_n$$

$$\Rightarrow \quad \tilde{x}_n = x_n + q_n$$

- This means that the quantized sample $\tilde{x}_n$ differs from the input $x_n$ by the quantization error $q_n$ independent of the predictor used.
- Therefore, the quantization errors do not accumulate.
3.5.1 Temporal Waveform Coding

- Differential pulse-code modulation (DPCM)
  - An improvement in the quality of the estimate is obtained by including linearly filtered past values of the quantized error.
  - The estimate may be expressed as:
    \[
    \hat{x}_n = \sum_{i=1}^{p} a_i \tilde{x}_{n-i} + \sum_{i=1}^{m} b_i \tilde{e}_{n-i}
    \]
    where \( \{b_i\} \) are the coefficients of the filter for the quantized error sequence.
  - The two sets of coefficients \( \{a_i\} \) and \( \{b_i\} \) are selected to minimize some function of the error \( e_n \), such as the mean square error.
3.5.1 Temporal Waveform Coding

- DPCM modified by the addition of linearly filtered error sequence.

![Diagram](image)
3.5.1 Temporal Waveform Coding

Adaptive PCM and DPCM

- Many real sources are quasi-stationary in nature.
- One aspect of the quasi-stationary characteristic is that the variance and the autocorrelation function of the source output vary slowly with time.
- PCM and DPCM encoders are designed on the basis that the source output 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.
- One improvement of PCM and DPCM that reduces the dynamic range of the quantization noise is the use of an adaptive quantizer.
Adaptive PCM and DPCM

- A relatively simple method is to use a uniform quantizer that varies its step size in accordance with the variance of the past signal samples.

- For example, a short-term running estimate of the variance of $x_n$ can be computed from the input sequence $\{x_n\}$ and the step size can be adjusted on the basis of such an estimate. Such an algorithm has been successfully used by Jayant in the encoding of speech signals.

- The step size is adjusted recursively according to:
  \[ \Delta_{n+1} = \Delta_n M(n) \]
  where $M(n)$ is a factor, whose value depends on the quantizer level for the sample and $\Delta_n$ is the step size.
A 3-bit quantizer with an adaptive step size.
### 3.5.1 Temporal Waveform Coding

- **Multiplication factors for adaptive step size adjustment**

<table>
<thead>
<tr>
<th></th>
<th>PCM</th>
<th></th>
<th>DPCM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$M(1)$</td>
<td>0.60</td>
<td>0.85</td>
<td>0.80</td>
<td>0.80</td>
</tr>
<tr>
<td>$M(2)$</td>
<td>2.20</td>
<td>1.00</td>
<td>0.80</td>
<td>1.60</td>
</tr>
<tr>
<td>$M(3)$</td>
<td>1.00</td>
<td>0.80</td>
<td></td>
<td>1.25</td>
</tr>
<tr>
<td>$M(4)$</td>
<td>1.50</td>
<td>0.80</td>
<td></td>
<td>1.70</td>
</tr>
<tr>
<td>$M(5)$</td>
<td></td>
<td>1.20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M(6)$</td>
<td></td>
<td>1.60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M(7)$</td>
<td></td>
<td>2.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M(8)$</td>
<td></td>
<td>2.40</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.5.1 Temporal Waveform Coding

- Delta modulation (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.
3.5.1 Temporal Waveform Coding

- **Delta modulation (DM)**

\[
\hat{x}_n = x_{n-1} = x_{n-1} + e_{n-1}
\]

\[
q_n = \hat{e}_n - e_n = \hat{e}_n - \left( x_n - \hat{x}_n \right)
\]

\[
\hat{x}_n = \hat{x}_{n-1} + \hat{e}_{n-1} = \left( q_{n-1} - \hat{e}_{n-1} + x_{n-1} \right) + \hat{e}_{n-1} = x_{n-1} + q_{n-1}
\]

- The estimated (predicted) value of \(x_n\) is really the previous sample \(x_{n-1}\) modified by the quantization noise \(q_{n-1}\).

- An equivalent realization of the one-step predictor is an accumulator with an input equal to the quantized error signal \(e_n\).
3.5.1 Temporal Waveform Coding

**Delta modulation (DM)**

- In general, the quantized error signal is scaled by some value, say $\Delta_1$, which is called the step size.
- The encoder shown in the previous figure approximates a waveform $x(t)$ by a linear staircase function.
- In order for the approximation to be relatively good, the waveform must change slowly relative to the sampling rate.
- The sampling rate must be several (a factor of at least 5) times the Nyquist rate.
- Two types of distortion:
  - Slope-overload distortion: $\Delta_1$ too small.
  - Granular noise: $\Delta_1$ too large.
3.5.1 Temporal Waveform Coding

- Delta modulation (DM)
  - Two types of distortion

```
\[ x(t) \]
```

```
Granular noise
```

```
Slope-overload distortion
```

WITS Lab, NSYSU.
3.5.1 Temporal Waveform Coding

- **Delta modulation (DM)**
  - Even when $\Delta_1$ is optimized to minimize the total mean square value of the slope-overload distortion and the granular noise, the performance of the DM encoder may still be less than satisfactory.
  - An alternative solution is to employ a variable step size that adapts itself to the short-term characteristics of the source signal.
3.5.1 Temporal Waveform Coding

- Delta modulation (DM)
  - Jayant devised a simple rule in 1970 to adaptively vary the step size according to:
    \[ \Delta_n = \Delta_{n-1} K^{\tilde{e}_n \tilde{e}_{n-1}}, \quad n = 1, 2, \ldots \]
  - where \( k \geq 1 \) is a constant that is selected to minimize the total distortion.
3.5.1 Temporal Waveform Coding

- Delta Modulation (DM)
- An example of a delta modulation with adaptive step size.
3.5.1 Temporal Waveform Coding

- Delta Modulation (DM)
  - Continuously variable slope delta modulation (CVSD)
    \[
    \Delta_n = \alpha \Delta_{n-1} + k_1 \quad \text{if } \tilde{e}_n, \tilde{e}_{n-1}, \text{ and } \tilde{e}_{n-2} \text{ have the same sign.}
    \]
    \[
    \Delta_n = \alpha \Delta_{n-1} = k_2 \quad \text{otherwise}
    \]

    where \(0 < \alpha < 1\) and \(k_1 \gg k_2 > 0\).
3.5.2 Spectral Waveform Coding

Subband coding (SBC)

- In SBC of speech and image signals, the signal is divided into a small number of subbands and the time waveform in each subband is encoded separately.

Example: Speech Coding:
- In speech coding, the lower-frequency bands contain most of the spectral energy in voiced speech.
- Quantization noise is more noticeable to the ear in the lower-frequency bands.
- 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 in SBC.
3.5.2 Spectral Waveform Coding

- **Adaptive Transform Coding (ATC)**
  - In ATC, the source signal is sampled and subdivided into frames of $N_f$ samples, and 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 (D/A) converter.
  - To achieve coding efficiency, we assign more bits to the more important spectral coefficients and fewer bits to the less important spectral coefficients.
3.5.3 Model-Based Source Coding

- Source is modeled as a linear system (filter) that, when excited by an appropriate input signal, results in the observed source output.

- Instead of transmitting the samples of the source waveform to the receiver, the parameters of the linear system are transmitted along with an appropriate excitation signal.

- If the number of parameters is sufficiently small, the model-based methods provide a large compression of the data.
3.5.3 Model-Based Source Coding

**Linear Predictive Coding (LPC)**

- The sampled sequence, denoted by $x_n, n=0,1,...,N-1$, is assumed to have been generated by an all-pole (discrete-time) filter having the transfer function:

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}}$$

- Suppose that the input sequence is denoted by $v_n, n=0,1,2,...$. The output sequence of the all-pole model satisfies the difference equation:

$$x_n = \sum_{k=1}^{p} a_k x_{n-k} + Gv_n, \quad n = 0,1,2,...$$

This equation is not true in general.
Linear predictive coding (LPC)

If the input is a white-noise sequence or an impulse, we may form an estimate (or prediction) of \( x_n \) by the weighted linear combination:

\[
\hat{x}_n = \sum_{k=1}^{p} a_k x_{n-k}, \quad n > 0
\]

The error between the observed value \( x_n \) and the estimated (predicted) value \( \hat{x}_n \):

\[
e_n = x_n - \hat{x}_n = x_n - \sum_{k=1}^{p} a_k x_{n-k}
\]

The filter coefficients \( \{a_k\} \) can be selected to minimize the mean square value of this error.
### 3.5.3 Model-Based Source Coding

**Linear predictive coding (LPC)**

- Suppose that the input \( \{v_n\} \) is a white-noise sequence. Then, the filter output \( x_n \) is a random sequence and so is the difference \( e_n \). The ensemble average of the squared error is:

\[
\epsilon_p = E\left( e_n^2 \right) = E\left[ \left( x_n - \sum_{k=1}^{p} a_k x_{n-k} \right)^2 \right]
\]

\[
= \phi(0) - 2 \sum_{k=1}^{p} a_k \phi(k) + \sum_{k=1}^{p} \sum_{m=1}^{p} \phi(k-m)
\]

- To specify the filter \( H(z) \), we must determine the filter gain \( G \).

\[
E\left[ (Gv_n)^2 \right] = G^2 E\left( v_n^2 \right) = G^2 = E\left[ \left( x_n - \sum_{k=1}^{p} a_k x_{n-k} \right)^2 \right] = \epsilon_p
\]
### 3.5.3 Model-Based Source Coding

- **Linear predictive coding (LPC)**
  - Hence, \( G^2 \) simplifies to:
    \[
    \varepsilon_p = G^2 = \phi(0) - \sum_{k=1}^{p} a_k \phi(k)
    \]
  - In practice, we do not usually know a priori the true autocorrelation function of the source output. Hence, in place of \( \phi(n) \), we substitute an estimate \( \hat{\phi}(n) \).
Linear predictive coding (LPC)

The Levinson-Durbin algorithm derived in Appendix A may be used to solve for the predictor coefficients \( \{a_k\} \) recursively.

\[
\hat{\phi}(i) - \sum_{k=1}^{i-1} a_{i-k} \hat{\phi}(i-k) = a_{ii} \frac{\sum_{k=1}^{i-1} a_{i-k} \hat{\phi}(i-k)}{\hat{\varepsilon}_{i-1}}, \quad i = 2, 3, \ldots, p
\]

\[
a_{ik} = a_{i-k} - a_{ii} a_{i-1-i-k}, \quad 1 \leq k \leq i - 1
\]

\[
\hat{\varepsilon}_i = (1 - a_{ii}) \hat{\varepsilon}_{i-1} \quad a_{11} = \frac{\hat{\phi}(1)}{\hat{\phi}(0)} \quad \hat{\varepsilon}_0 = \hat{\phi}(0)
\]

\( a_{ik}, k=1,2,\ldots,i, \) are the coefficients of the \( i \)th-order predictor.
3.5.3 Model-Based Source Coding

- Linear predictive coding (LPC)
  - The desired coefficients for the predictor of order \( p \) are:
    \[ a_k \equiv a_{pk}, \quad k = 1, 2, \ldots, p \]
  - The residual MSE is:
    \[ \hat{\varepsilon} = G^2 = \hat{\phi}(0) - \sum_{k=1}^{p} a_k \phi(k) = \hat{\phi}(0) \prod_{i=1}^{p} (1 - a_{ii}^2) \]
  - The residual MSE \( \hat{\varepsilon}_i \), \( i = 1, 2, \ldots, p \), forms a monotone decreasing sequence, i.e., \( \hat{\varepsilon}_p \leq \hat{\varepsilon}_{p-1} \leq \ldots \hat{\varepsilon}_1 \leq \hat{\varepsilon}_0 \), and the prediction coefficients \( a_{ii} \) satisfy the condition:
    \[ |a_{ii}| < 1, \quad i = 1, 2, \ldots, p \]

The condition is necessary and sufficient for all the poles of \( H(z) \) to be inside the unit circle.
3.5.3 Model-Based Source Coding

- **Linear predictive coding (LPC)**
  - LPC has been successfully used in the modeling of a speech source. In this case, the coefficients $a_{ii}, i=1,2,...,p$, are called *reflection coefficients* as a consequence of their correspondence to the reflection coefficients in the acoustic tube model of the vocal tract.
  - Once the predictor coefficients and the gain $G$ have been estimated from the source output $\{x_n\}$, each parameter is coded into a sequence of binary digits and transmitted to the receiver.
3.5.3 Model-Based Source Coding

- Linear predictive coding (LPC)
- Block diagram of a waveform synthesizer (source decoder) for an LPC system
Linear predictive coding (LPC)

- The signal generator is used to produce the excitation function \( \{v_n\} \), which is scaled by \( G \) to produce the desired input to the all-pole filter model \( H(z) \) synthesized from the received prediction coefficient.

- The analog signal may be reconstructed by passing the output sequence from \( H(z) \) through an analog filter that basically performs the function of interpolating the signal between sample points.

- In this realization of the waveform synthesizer, the excitation function and the gain parameter must be transmitted along with the prediction coefficients to the receiver.
3.5.3 Model-Based Source Coding

- Linear predictive coding (LPC)
  - When the source output is stationary, the filter parameters need to be determined only once.
  - Since most sources encountered in practice are at best quasi-stationary, it is necessary to periodically obtain new estimates of the filter coefficients, the gain $G$, and the type of excitation function, and to transmit these estimates to the receiver.
3.5.3 Model-Based Source Coding

- Linear predictive coding (LPC)
  - Example 3.5-1: Block diagram model of the generation of a speech signal.
3.5.3 Model-Based Source Coding

- Linear predictive coding (LPC)
- Example 3.5-1 (cont.)
  - There are two mutually exclusive excitation functions to model voiced and unvoiced speech sounds.
  - On a short-time basis, voiced speech is periodic with a fundamental frequency $f_0$ or a pitch period $1/f_0$ that depends on the speaker.
  - Voiced speech is generated by exciting an all-pole filter model of the vocal tract by a periodic impulse train with a period equal to the desired pitch period.
  - Unvoiced speech sounds are generated by exciting the all-pole filter model by the output of a random-noise generator.
Linear predictive coding (LPC)

Example 3.5-1 (cont.)

- The speech encoder at the transmitter must determine the proper excitation function, the pitch period of voiced speech, the gain parameter $G$, and the prediction coefficients.
- Typically, the voiced and unvoiced information requires 1 bit, the pitch period is adequately represented by 6 bits, and the gain parameter may be represented by 5 bits after its dynamic range is compressed logarithmically.
- The prediction coefficients require 8-10 bits per coefficient for adequate representation.
- The bit rate from the source encoder is 4800-2400 bits/s.
Linear predictive coding (LPC)

- When the reflection coefficients are transmitted to the decoder, it is not necessary to re-compute the prediction coefficients in order to realize the speech synthesizer.
- The synthesis is performed by realizing a lattice filter, which utilizes the reflection coefficients directly and which is equivalent to the linear prediction filter.
3.5.3 Model-Based Source Coding

- Linear predictive coding (LPC)
  - A more general source model is a linear filter that contains both poles and zeros. The source output $x_n$ satisfies the difference equation:

$$x_n = \sum_{i=1}^{p} a_k x_{n-k} + \sum_{k=0}^{q} b_k v_{n-k}$$

where $v_n$ is the input excitation sequence.

- The MSE criterion applied to the minimization of the error $e_n = x_n - \hat{x}_n$, where $\hat{x}_n$ is an estimate of $x_n$, results in a set of non-linear equations for the parameters $\{a_k\}$ and $\{b_k\}$.

- For speech coding, the model-based methods are generally called *vocoders*.
### Model-Based Source Coding

#### Encoding techniques applied to speech signals

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