Digital Communications and Signal Processing – with Matlab Examples

Prof. Jianfeng Feng

Department of Computer science and Centre for Scientific Computing
University of Warwick CV4 7AL, UK

Contents

1 Introduction

Digital communications and signal processing refers to the field of study concerned with the transmission and processing of digital data. This is in contrast with analog communications. While analog communications use a continuously varying signal, a digital transmission can be broken down into discrete messages. Transmitting data in discrete messages allows for greater signal processing capability. The ability to process a communications signal means that errors caused by random processes can be detected and corrected. Digital signals can also be sampled instead of continuously monitored and multiple signals can be multiplexed together to form one signal.

Because of all these advantages, and because recent advances in wideband communication channels and solid-state electronics have allowed scientists to fully realize these advantages, digital communications has grown quickly. Digital communications is quickly edging out analog communication because of the vast demand to transmit computer data and the ability of digital communications to do so.

Here is a summary on what we will cover in this course.

1. Data transmission: Channel characteristics, signalling methods, interference and noise, and synchronisation;

2. Information Sources and Coding: Information theory, coding of information for efficiency and error protection encryption;

3. Signal Representation: Representation of discrete time signals in time and frequency; z transform and Fourier representations; discrete approximation of continuous signals; sampling and quantisation; and data compression;

4. Filtering: Analysis and synthesis of discrete time filters; finite impulse response and infinite impulse response filters; frequency response of digital filters; poles and zeros; filters for correlation and detection; matched filters and stochastic signals and noise processes;

5. Digital Signal Processing applications: Processing of images using digital techniques.

The application of DCSP in industry and our daily life is enormous, although in this introductory module we are only able to touch several simple examples.

Part of the current lecture notes on DSP is taken from lecture notes of Prof. R. Wilson. Many materials are adopted from public domain materials. Many thanks to Dr. Enrico Rossoni who has spent a considerable time on going through the manuscript several times to correct typos. The sections I include only for your reference and I will not go through them during lectures are marked with a $*$.

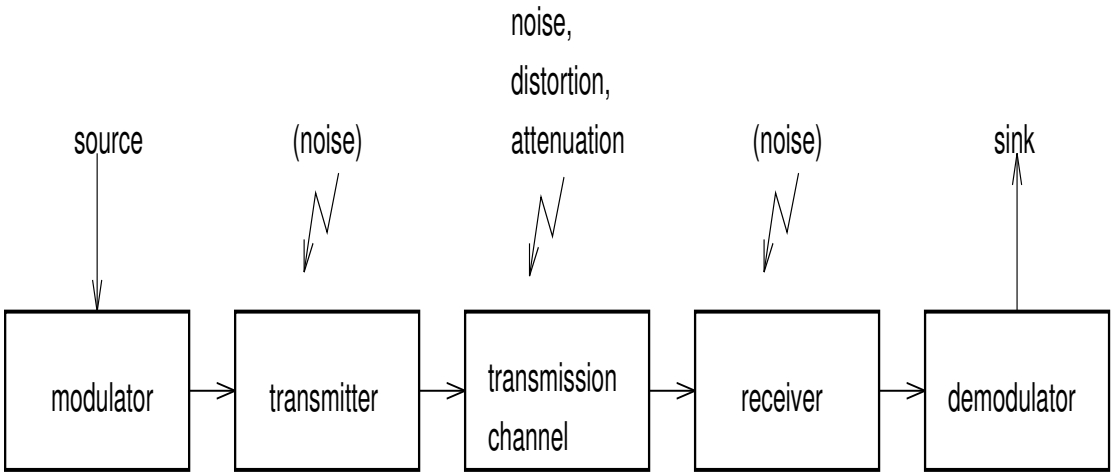

Figure 1: A communications system

2 Data Transmission

2.1 The transmission of information

2.1.1 General Form

A communications system is responsible for the transmission of information from the sender to the recipient. At its simplest, the system contains (see Fig. 1)

1. A modulator that takes the source signal and transforms it so that it is physically suitable for the transmission channel

2. A transmission channel that is the physical link between the communicating parties

3. A transmitter that actually introduces the modulated signal into the channel, usually amplifying the signal as it does so

4. A receiver that detects the transmitted signal on the channel and usually amplifies it (as it will have been attenuated by its journey through the channel)

5. A demodulator that receives the original source signal from the received signal and passes it to the sink

At each stage, signal processing techniques are required to detect signals, filter out noise and extract features, as we will discuss in the second part of our course.

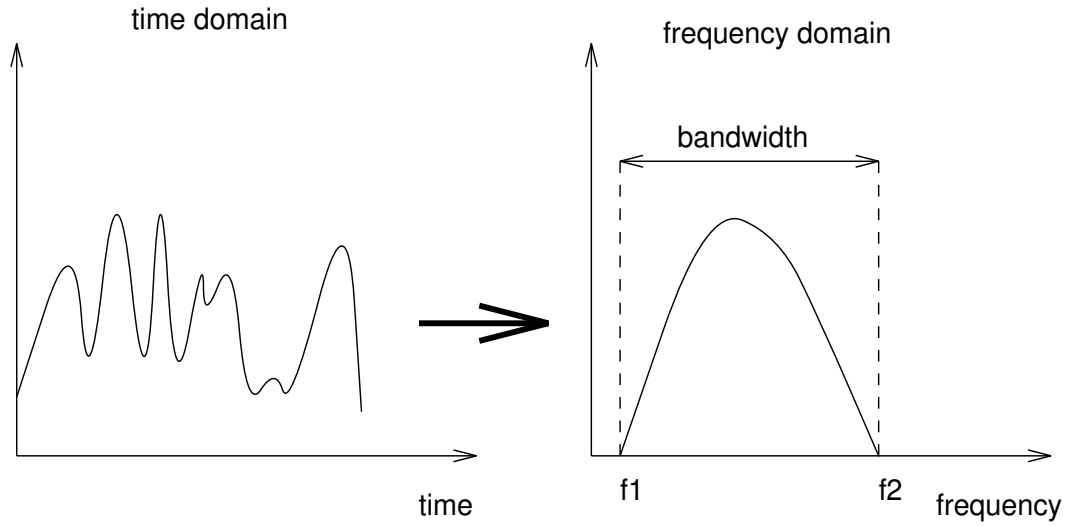

Figure 2: Time domain and frequency domain representation of a signal.

Digital data is universally represented by strings of 1s or 0s. Each one or zero is referred to as a *bit*. Often, but not always, these bit strings are interpreted as numbers in a binary number system. Thus $101001_2 = 41_{10}$. The information content of a digital signal is equal to the number of bits required to represent it. Thus a signal that may vary between 0 and 7 has an information content of 3 bits. Written as an equation this relationship is

$$I = \log_2(n) \text{ bits} \tag{2.1}$$

where n is the number of levels a signal may take. It is important to appreciate that information is a measure of the number of different outcomes a value may take.

The information rate is a measure of the speed with which information is transferred. It is measured in bits/second or b/s.

2.1.2 Examples

Telecommunications traffic is characterised by great diversity. A non-exclusive list is the following:

1. Audio signals. An audio signal is an example of an analogue signal. It occupies a frequency range from about 200 Hz to about 15KHz. Speech signals occupy a smaller range of frequencies, and telephone speech typically occupies the range 300 Hz to 3300 Hz. The range of frequencies occupied by the signal is called its bandwidth (see Fig. 2).

2. Television. A television signal is an analogue signal created by linearly scanning a two dimensional image. Typically the signal occupies a bandwidth of about 6 MHz.

3. Teletext is written (or drawn) communications that are interpreted visually. Telex describes a message limited to a predetermined set of alphanumeric characters.
4. Reproducing cells, in which the daughter cells's DNA contains information from the parent cells;
5. A disk drive
6. Our brain

The use of digital signals and modulation has great advantages over analogue systems. These are:

1. High fidelity. The discrete nature of digital signals makes their distinction in the presence of noise easy. Very high fidelity transmission and representation are possible.
2. Time independence. A digitised signal is a stream of numbers. Once digitised a signal may be transmitted at a rate unconnected with its recording rate.
3. Source independence. The digital signals may be transmitted using the same format irrespective of the source of the communication. Voice, video and text may be transmitted using the same channel.
4. Signals may be coded. The same transmitted message has an infinite number of meanings according to the rule used to interpret it.

One disadvantage of digital communication is the increased expense of transmitters and receivers. This is particularly true of real-time communication of analogue signals.

2.1.3 The conversion of analogue and digital signals

In order to send analogue signals over a digital communication system, or process them on a digital computer, we need to convert analogue signals to digital ones. This process is performed by an analogue-to-digital converter (ADC). The analogue signal is sampled (i.e. measured at regularly spaced instant) (Fig 3) and then quantised (Fig. 3, bottom panel) i.e. converted to discrete numeric values. The converse operation to the ADC is performed by a digital-to-analogue converter (DAC).

The ADC process is governed by an important law. The Nyquist-Shannon Theorem (which will be discussed in Chapter 3) states that an analogue signal of bandwidth *B* can be completely recreated from its sampled form provided it is sampled at a rate equal to at least twice its bandwidth. That is

$$S \geq 2B \tag{2.2}$$

The rate at which an ADC generates bits depends on how many bits are used in the converter. For example, a speech signal has an approximate bandwidth of 4kHz. If this is sampled by an 8-bit ADC at the Nyquist sampling rate, the bit rate *R* to transform the signal without loss of information is

$$R = 8\text{bits} \times 2B = 64000b/s \tag{2.3}$$

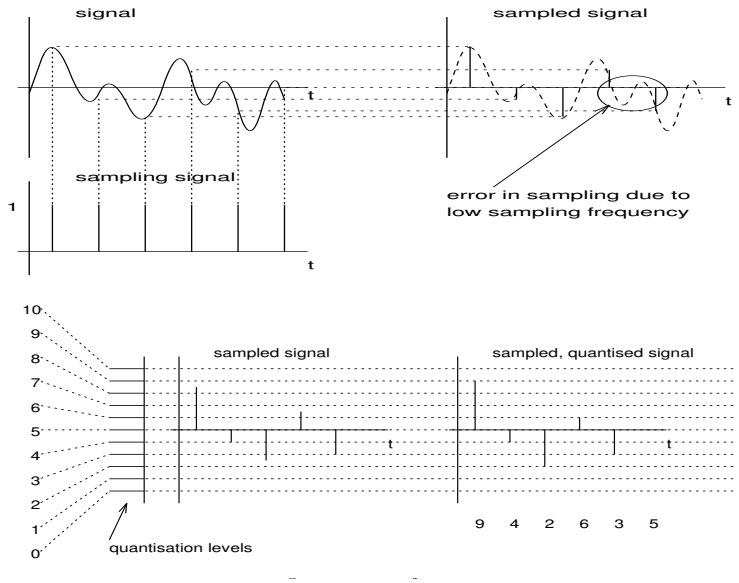

Figure 3: Upper panel: Periodic sampling of an analogue signal. Bottom panel: Quantisation of a sampled signal.

2.1.4 The relationship between information, bandwidth and noise

The most important question associated with a communication channel is the maximum rate at which it can transfer information. Analogue signals passing through physical channels may not achieve arbitrarily fast changes. The rate at which a signal may change is determined by the bandwidth. Namely, a signal of bandwidth *B* may change at a maximum rate of 2*B*, so the maximum information rate is 2*B*. If changes of differing magnitude are each associated with a separate bit, the information rate may be increased. Thus, if each time the signal changes it can take one of n levels, the information rate is increased to

$$R = 2B \log_2(n) \text{ b/s} \tag{2.4}$$

This formula states that as n tends to infinity, so does the information rate.

Is there a limit on the number of levels? The limit is set by the presence of noise. If we continue to subdivide the magnitude of the changes into ever decreasing intervals, we reach a point where we cannot distinguish the individual levels because of the presence of noise. Noise therefore places a limit on the maximum rate at which we can transfer information. Obviously, what really matters is the signal to noise ratio (SNR). This is defined by the ratio signal power S to noise power N, and is often expressed in deciBels (dB):

$$SNR = 10 \log_{10}(S/N) \;\; dB \tag{2.5}$$

The source of noise signals vary widely.

1. Input noise is common in low frequency circuits and arises from electric fields generated by electrical switching. It appears as bursts at the receiver, and when present can have a catastrophic effect due to its large power. Other people's signals can generate noise: cross-talk is the term given to the pick-up of radiated signals from adjacent cabling. When radio links are used, interference from other transmitters can be problematic.

2. Thermal noise is always present. This is due to the random motion of electric charges present in all media. It can be generated externally, or internally at the receiver.

There is a theoretical maximum to the rate at which information passes error free over a channel. This maximum is called the *channel capacity*, C. The famous Hartley-Shannon Law states that the channel capacity, C (which we will discuss in details later) is given by

$$C = B\log_2(1 + (S/N)) \qquad b/s \tag{2.6}$$

For example, a 10kHz channel operating at a SNR of 15dB has a theoretical maximum information rate of $10000\log_2(31.623) = 49828b/s$.

The theorem makes no statement as to how the channel capacity is achieved. In fact, in practice channels only approach this limit. The task of providing high channel efficiency is the goal of coding techniques.

2.2 Communication Techniques

2.2.1 Time, frequency and bandwidth

Most signal carried by communication channels are modulated forms of sine waves. A sine wave is described mathematically by the expression

$$s(t) = A\cos(\omega t + \phi) \tag{2.7}$$

The quantities A, ω, ϕ are termed the amplitude, frequency and phase of the sine wave. We can describe this signal in two ways. One way is to describe its evolution in time domain, as in the equation above. The other way is to describe its frequency content, in frequency domain. The cosine wave, $s(t)$, has a single frequency, $\omega = 2\pi f$.

This representation is quite general. In fact we have the following theorem due to Fourier.

Theorem 1 *Any signal $x(t)$ of period T can be represented as the sum of a set of cosinusoidal and sinusoidal waves of different frequencies and phases.*

Mathematically

$$x(t) = A_0 + \sum_{n=1}^{\infty} A_n\cos(\omega n t) + \sum_{n=1}^{\infty} B_n\sin(\omega n t) \tag{2.8}$$

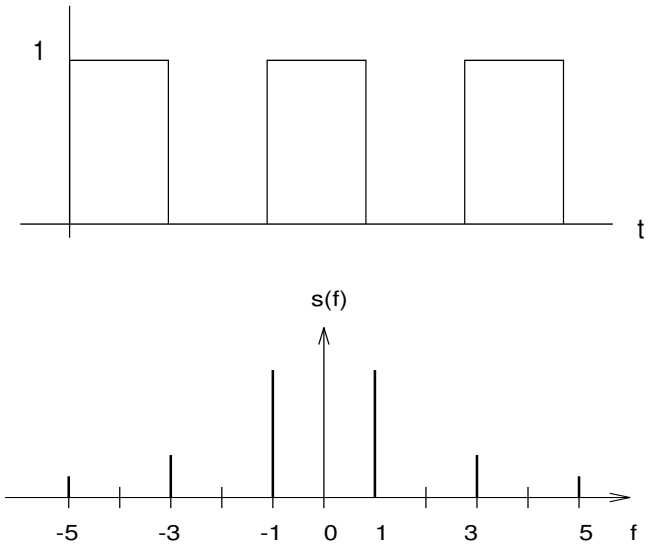

Figure 4: Upper panel: a square wave. Bottom panel: The frequency spectrum for the square wave.

where

$$\begin{cases} A_0 &= \dfrac{1}{T}\displaystyle\int_{-T/2}^{T/2} x(t)dt \\ A_n &= \dfrac{2}{T}\displaystyle\int_{-T/2}^{T/2} x(t)\cos(\omega nt)dt \\ B_n &= \dfrac{2}{T}\displaystyle\int_{-T/2}^{T/2} x(t)\sin(\omega nt)dt \\ \omega &= \dfrac{2\pi}{T} \end{cases} \tag{2.9}$$

where A_0 is the d.c. term, and T is the period of the signal. The description of a signal in terms of its constituent frequencies is called its frequency spectrum.

Example 1 *As an example, consider the square wave (Fig. 4)*

$$s(t) = 1, 0 < t < \pi, 2\pi < t < 3\pi, ... \tag{2.10}$$

and zero otherwise. This has the Fourier series:

$$s(t) = \frac{1}{2} + \frac{2}{\pi}[\sin(t) + \frac{1}{3}\sin(3t) + \frac{1}{5}\sin(5t) + \cdots] \tag{2.11}$$

A graph of the spectrum has a line at the odd harmonic frequencies, 1,3,5,9, ..., whose respective amplitudes decay as $2/\pi, 2/3\pi, \cdots$. The spectrum of a signal is usually shown as a two-sided

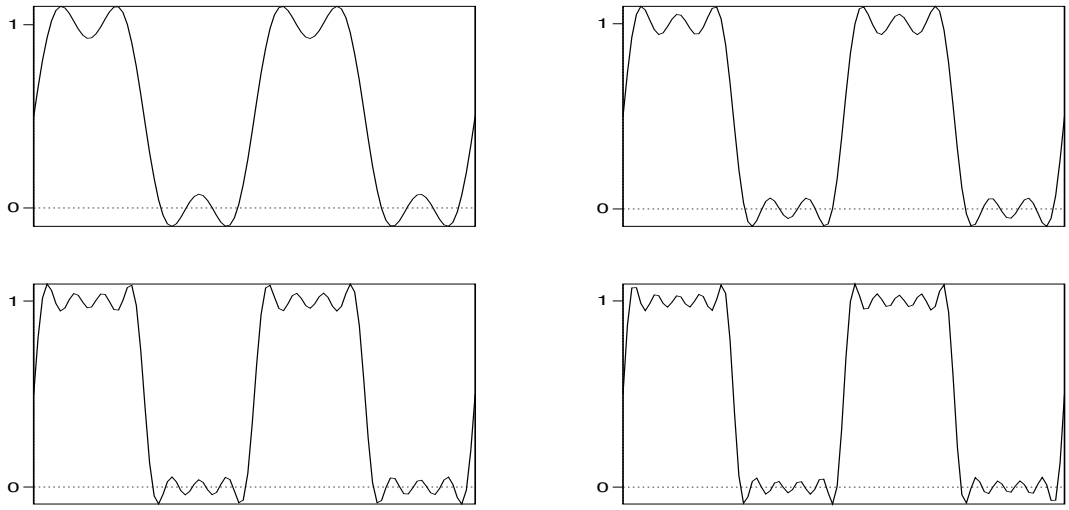

Figure 5: A square wave with 3,4,5 and 6 of its Fourier terms.

spectrum with positive and negative frequency components. The coefficients are obtained according to

$$\begin{aligned} A_0 &= \frac{1}{2\pi}\int_0^{\pi} 1dt \\ &= \frac{1}{2} \\ A_n &= \frac{2}{2\pi}\int_0^{\pi} 1\cos(nt)dt \\ &= \frac{1}{n\pi}\sin(n\pi) \\ &= 0 \\ B_n &= \frac{2}{2\pi}\int_0^{\pi} 1\sin(nt)dt \\ &= \frac{1}{n\pi}(1-\cos(n\pi)) \end{aligned} \tag{2.12}$$

which gives $B_1 = 2/\pi, B_2 = 0, B_3 = 2/3\pi, ...$

A periodic signal is uniquely decided by its coefficients A_n, B_n. For example, in the Example 1, we have

$$\begin{array}{c} x(t) \\ || \\ \{\cdots, B_{-n}, \cdots, B_{-2}, B_{-1}, \quad B_0 \quad , B_1, B_2, \cdots, B_n, \cdots, \} \end{array} \tag{2.13}$$

If we truncate the series into finite terms, the signal can be approximated by a finite series as shown in Fig. 5.

In general, a signal can be represented as follows (see Appendix 6.7)

$$\begin{array}{rcl} x(t) & = & A_0 + \sum_{n=1}^{\infty} A_n \cos(\omega nt) + \sum_{n=1}^{\infty} B_n \sin(\omega nt) \\ & = & A_0 + \sum_{n=1}^{\infty} A_n [\exp(j\omega nt) + \exp(-j\omega nt)]/2 \\ & & + \sum_{n=1}^{\infty} B_n [\exp(j\omega nt) - \exp(-j\omega nt)]/2j \\ & = & \sum_{n=-\infty}^{\infty} c_n \exp(j\omega nt) \end{array} \tag{2.14}$$

which is the exponential form of the Fourier series. In this expression, the values c_n are complex and so $|c_n|$ and $arg(c_n)$ are the magnitude and the phase of the spectral component respectively,

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} x(t) \exp(-j\omega nt) dt \tag{2.15}$$

where $\omega = 2\pi/T$.

Signals whose spectra consist of isolated lines are periodic, i.e. they represent themselves indefinitely. The lines in this spectrum are infinitely thin, i.e. they have zero bandwidth. The Hartley-Shannon law tells us that the maximum information rate of a zero bandwidth channel is zero. Thus zero bandwidth signals carry no information. To permit the signal to carry information we must introduce the capacity for aperiodic change. The consequence of an aperiodic change is to introduce a spread of frequencies into the signal.

If the square wave signal discussed in the previous example is replaced with an aperiodic sequence, the spectrum changes substantially.

$$\left\{ \begin{array}{rcl} X(F) & = & FT\{x(t)\} = \int_{-\infty}^{\infty} x(t) \exp(-j2\pi Ft) dt \\ x(t) & = & IFT\{X(t)\} = \int_{-\infty}^{\infty} X(t) \exp(j2\pi Ft) dt \end{array} \right. \tag{2.16}$$

Example 2 *Consider the case of a rectangular pulse. In particular define the signal*

$$x(t) = \left\{ \begin{array}{ll} 1, & \textit{if } -0.5 \leq t < 0.5 \\ 0, & \textit{otherwise} \end{array} \right.$$

This is shown in Fig. 6 and its Fourier transform can be readily computed from the definition, as

$$X(F) = \int_{-0.5}^{0.5} x(t) \exp(-j2\pi Ft) dt = \frac{\sin(\pi F)}{\pi F}$$

and is plotted in Fig. 6.

There are a number of features to note:

1. The bandwidth of the signal is only approximately finite. Most of the energy is contained in a limited region called the main-lobe. However, some energy is found at all frequencies.

2. The spectrum has positive and negative frequencies. These are symmetric about the origin. This may seem non-intuitive but can be seen from equations above.

The bandwidth of a communication channel is limited by the physical construction of the channel.

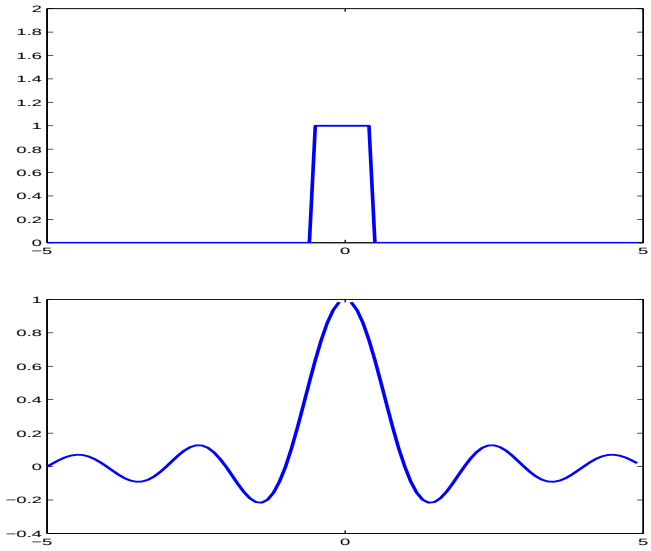

Figure 6: A square wave and its Fourier transform.

2.2.2 Digital modulation: ASK, FSK and PSK

There are three ways in which the bandwidth of the channel carrier may be altered simply. These are the altering of the amplitude, frequency and phase of the carrier wave. These techniques give rise to amplitude-shift-keying (ASK), frequency-shift-keying (FSK) and phase-shift-keying (PSK), respectively.

ASK describes the technique by which a carrier wave is multiplied by the digital signal $f(t)$. Mathematically the modulated carrier signal $s(t)$ is (Fig. 7)

$$s(t) = f(t)\cos(\omega_c t + \phi) \tag{2.17}$$

ASK is a special case of amplitude modulation (AM). Amplitude modulation has the property of translating the spectrum of the modulation $f(t)$ to the carrier frequency. The bandwidth of the signal remains unchanged. This can be seen if we examine a simple case when $f(t) = cos(\omega t)$ and we use the identities:

$$\begin{aligned} \cos(A+B) &= \cos(A)\cos(B) - \sin(A)\sin(B) \\ \cos(A-B) &= \cos(A)\cos(B) + \sin(A)\sin(B) \end{aligned} \tag{2.18}$$

then

$$s(t) = \cos(\omega t)\cos(\omega_c t) = 1/2[\cos((\omega+\omega_c)t) + \cos((\omega-\omega_c)t)$$

See fig. 8.

FSK describes the modulation of a carrier (or two carriers) by using a different frequency for a 1 or 0. The resultant modulated signal may be regarded as the sum of two amplitude modulated

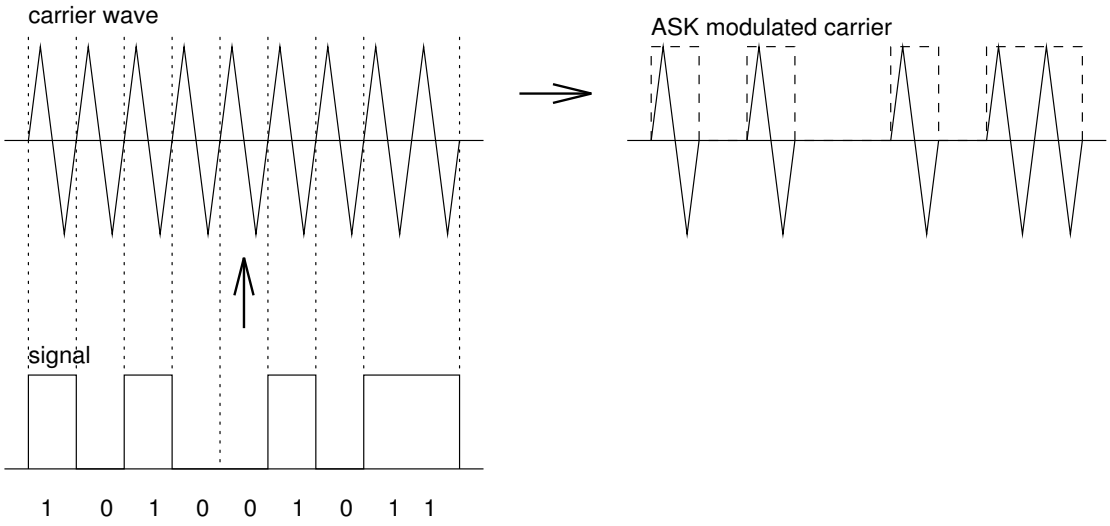

Figure 7: Amplitude shift keying

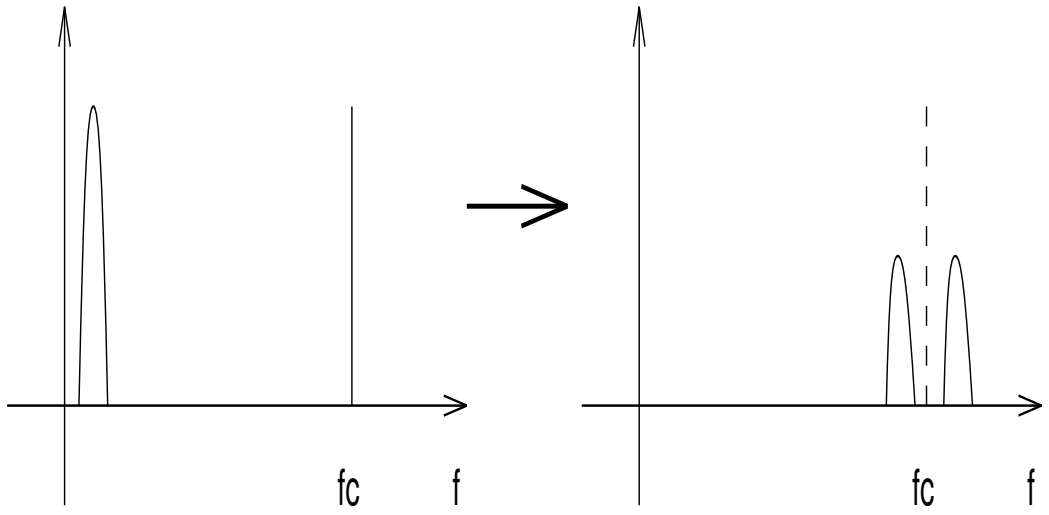

Figure 8: Amplitude shift keying: frequency domain

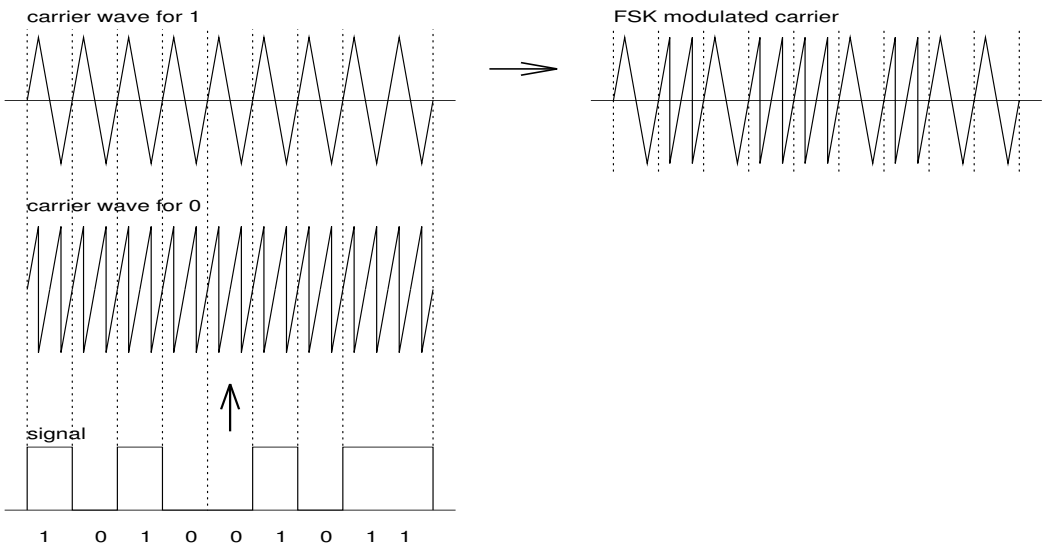

Figure 9: Frequency shift keying

signals of different carrier frequency

$$s(t) = f_0(t)\cos(\omega_0 t + \phi) + f_1(t)\cos(\omega_1 t + \phi)$$

FSK is classified as wide-band if the separation between the two carrier frequencies is larger than the bandwidth of the spectrums of f_0 and f_1. In this case the spectrum of the modulated signal appears as two separate ASK signals.

PSK describes the modulation technique that alters the phase of the carrier. Mathematically

$$s(t) = \cos(\omega_c t + \phi(t))$$

Binary phase-shift-keying (BPSK) has only two phases, 0 and π. It is therefore a type of ASK with $f(t)$ taking he values -1 and 1, and its bandwidth is the same as that of ASK (Fig. 11). Phase-shift keying offers a simple way of increasing the number of levels in the transmission without increasing the bandwidth by introducing smaller phase shifts. Quadrature phase-shift-keying (QPSK) has four phases, 0,$\pi/2, \pi, 3\pi/2$. M-ary PSK has M phases.

2.2.3 Spread spectrum techniques

Spread-spectrum techniques are methods in which energy generated at a single frequency is deliberately spread over a wide band of frequencies. This is done for a variety of reasons, including increasing resistance to natural interference or jamming and to prevent hostile detection.

We shall not delve deeply into mechanisms, but shall look at one particular technique that is used called *frequency hopping*, as shown in Fig. 12. In frequency hopping, the bandwidth is

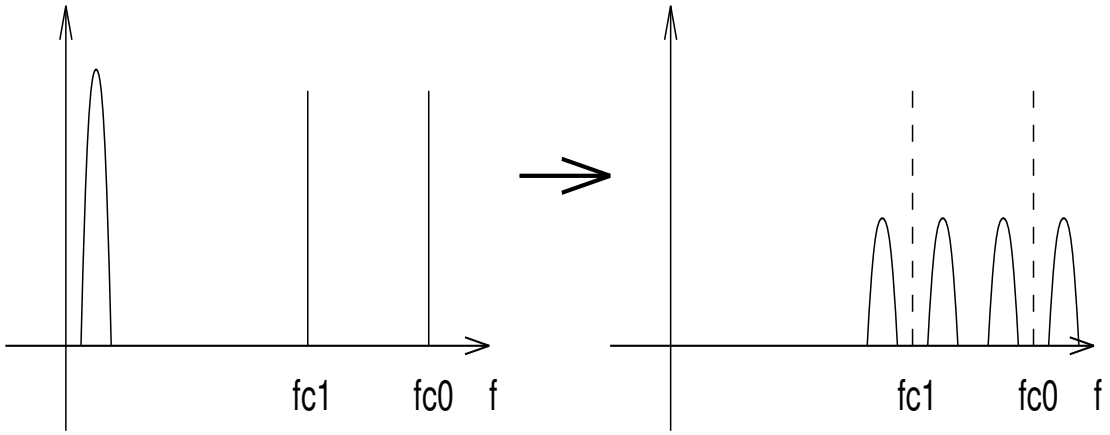

Figure 10: Frequency shift keying: frequency domain

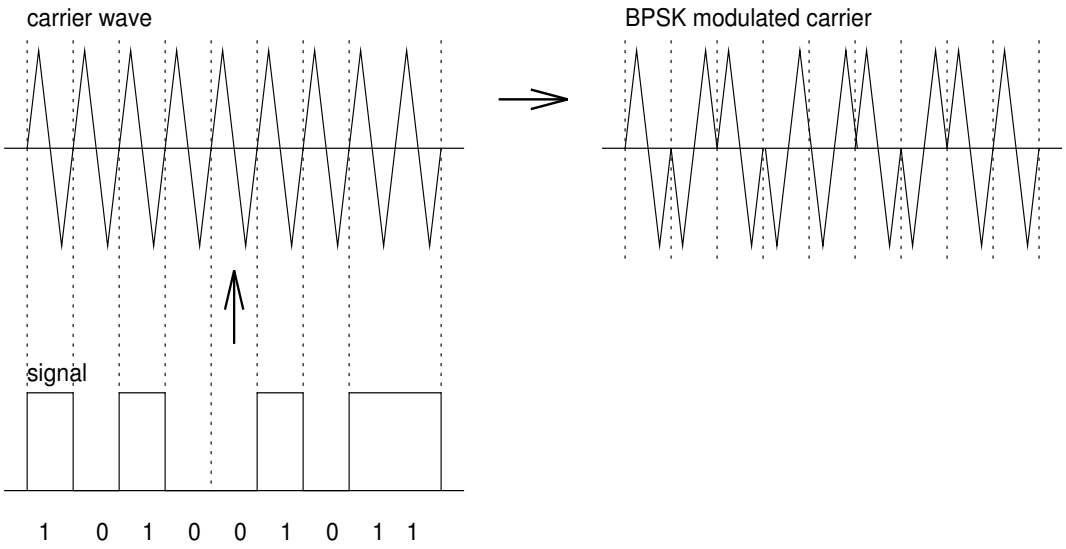

Figure 11: Phase shift keying

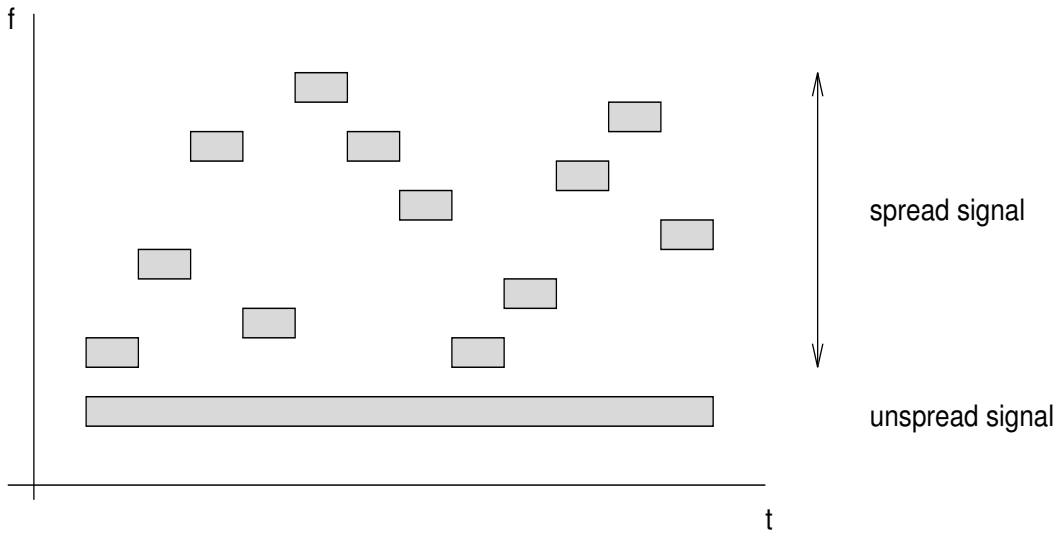

Figure 12: Frequency hopping spread spectrum technique

effectively split into frequency channels. The signal is then spread across the channels. The hop set (channel hopping sequence) is not arbitrary, but determined by the use of a pseudo random sequence. The receiver can reproduce the identical hop set and so decode the signal. The hop rate (the rate at which the signal switches channels) can be thousands of times a second, so the dwell time (time spent on one channel) is very short. If the hop set is generated by a pseudo random number generator then the seed to that generator is effectively a key decoding the transmitted message, and so this technique has obvious security applications, for instance military use or in mobile phone systems.

2.2.4 Digital demodulation

From the discussion above it might appear that QPSK offers advantages over both ASK, FSK and PSK. However, the demodulation of these signals requires various degrees of difficulty and hence expense. The method of demodulation is an important factor in determining the selection of a modulation scheme. There are two types of demodulation which are distinguished by the need to provide knowledge of the phase of the carrier. Demodulation schemes requiring the carrier phase are termed coherent. Those that do not need knowledge of the carrier phase are termed incoherent. Incoherent demodulation can be applied to ASK and wide-band FSK. It describes demodulation schemes that are sensitive only to the power in the signal. With ASK, the power is either present, or it is not. With wide-band FSK, the power is either present at one frequency, or the other. Incoherent modulation is inexpensive but has poorer performance. Coherent demodulation requires more complex circuity, but has better performance.

In ASK incoherent demodulation, the signal is passed to an envelope detector. This is a device that produces as output the outline of the signal. A decision is made as to whether the signal is

present or not. Envelope detection is the simplest and cheapest method of demodulation. In optical communications, phase modulation is technically very difficult, and ASK is the only option. In the electrical and microwave context, however, it is considered crude. In addition, systems where the signal amplitude may vary unpredictably, such as microwave links, are not suitable for ASK modulation.

Incoherent demodulation can also be used for wide-band FSK. Here the signals are passed to two circuits, each sensitive to one of the two carrier frequencies. Circuits whose output depends on the frequency of the input are called discriminators or filters. The outputs of the two discriminators are interrogated to determine the signal. Incoherent FSK demodulation is simple and cheap, but very wasteful of bandwidth. The signal must be wide-band FSK to ensure the two signals $f_0(t)$ and $f_1(t)$ are distinguished. It is used in circumstances where bandwidth is not the primary constraint.

With coherent demodulation systems, the incoming signal is compared with a replica of the carrier wave. This is obviously necessary with PSK signals, because here the power in the signal is constant. The difficulty with coherent detection is the need to keep the phase of the replica signal, termed local oscillator, 'locked' to the carrier. This is not easy to do. Oscillators are sensitive to (among other things) temperature, and a 'free-running' oscillator will gradually drift in frequency and phase.

Another way to demodulate the signal is performed by multiplying the incoming signal with a replica of the carrier. If the output of this process is $h(t)$, we have that

$$h(t) = f(t)\cos(\omega_c t)\cos(\omega_c t) = \frac{f(t)}{2}[1 + \cos(2\omega_c t)] = \frac{f(t)}{2} + \frac{f(t)}{2}\cos(2\omega_c t)$$

i.e. the original signal plus a term at twice the carrier frequency. By removing, or filtering out, the harmonic term, the output of the demodulation is the modulation $f(t)$. Suppose there is some phase error ϕ present in the local oscillator signal. After filtering, the output of a demodulator will be

$$h(t) = f(t)\cos(\omega_c t)\cos(\omega_c t + \phi) = \frac{f(t)}{2}\cos(\phi) + \frac{f(t)}{2}\cos(2\omega_c t + \phi)$$

Clearly the consequence for the correct interpretation of the demodulated signal is catastrophic.

Therefore, some more sophisticated methods such as differential phase-shift-keying (DSPK) have to be introduced to resolve the issue.

2.2.5 Noise in communication systems: probability and random signals

Noise plays a crucial role in communication systems. In theory, it determines the theoretical capacity of the channel. In practise it determines the number of errors occurring in a digital communication. We shall consider how the noise determines the error rates in the next subsection. In this subsection we shall provide a description of noise.

Noise is a random signal. By this we mean that we cannot predict its value. We can only make statements about the probability of it taking a particular value, or range of values. The Probability density function (pdf) $p(x)$ of a random signal, or random variable x is defined to be the probability that the random variable x takes a value between x_0 and $x_0 + \delta x$. We write this as follows:

$$p(x_0)\delta x = P(x_0 < x < x_0 + \delta x)$$

The probability that the random variable will take a value lying between x_1 and x_2 is then the integral of the pdf over the interval $[x_1 x_2]$:

$$P(x_1 < x < x_2) = \int_{x_1}^{x_2} p(x)dx$$

The probability $P(-\infty < x < \infty)$ is unity. Thus

$$\int_{-\infty}^{\infty} p(x)dx = 1$$

a density satisfying the equation above is termed normalized. The cumulative distribution function (CDF) $P(x)$ is defined to be the probability that a random variable, x is less than x_0

$$P(x_0) = P(x < x_0) = \int_{-\infty}^{x_0} p(x)dx$$

From the rules of integration:

$$P(x_1 < x < x_2) = P(x_2) - P(x_1)$$

Some commonly used distributions are:

1. Continuous distributions. An example of a continuous distribution is the Normal, or Gaussian distribution:
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{-(x-m)^2}{2\sigma^2}) \tag{2.19}$$
where m is the mean value of $p(x)$. The constant term ensures that the distribution is normalized. This expression is important as many actually occurring noise source can be described by it, i.e. white noise. Further, we can simplify the expression by considering the source to be a zero mean random variable, i.e. $m = 0$. σ is the standard deviation of the distribution (see Fig. 13).

   How would this be used? If we want to know the probability of, say, the noise signal, $n(t)$, having the value $[-v_1, v_1]$, we would evaluate:

$$P(v_1) - P(-v_1)$$

   In general to evaluate $P(-x_1 < x < x_1)$ if we use

$$u = x/(\sqrt{2}\sigma), dx = du\sigma\sqrt{2}$$

   then we have ($m = 0$)

$$P(-x_1 < x < x_1) = \frac{1}{\sqrt{\pi}} \int_{-u_1}^{u_1} \exp(-u^2)du = \frac{2}{\sqrt{\pi}} \int_{0}^{u_1} \exp(-u^2)du$$

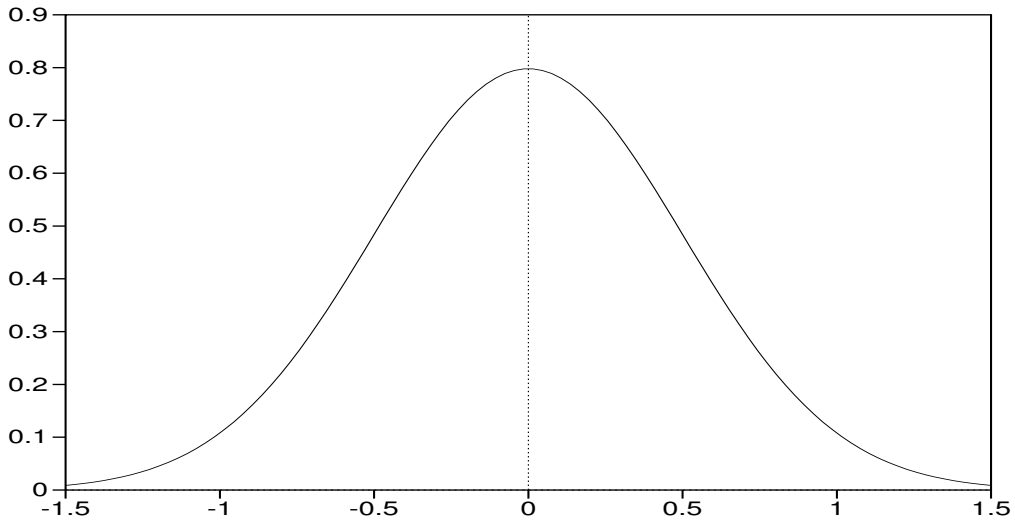

Figure 13: Gaussian distribution pdf.

where $u_1 = x_1/(\sqrt{2}\sigma)$. The distribution function $P(x)$ is usually written in terms of a function of the error function erf(x). The complementary error function $erfc$ is defined by

$$\text{erfc}(x) = 1 - \text{erf}(x)$$

2. Discrete distributions. Probability density functions need not be continuous. If a random variable can only take discrete value, its PDF takes the forms of lines (see Fig. 14). An example of a discrete distribution is the Poisson distribution

$$p(n) = P(x = n) = \frac{\alpha^n}{n!}\exp(-\alpha)$$

where $n = 0, 1, 2, \cdots$,.

We cannot predict the value a random variable may take on a particular occasion but we can introduce measures that summarise what we expect to happen on average. The two most important measures are the *mean* (or expectation) and the *standard deviation.*

The mean η of a random variable x is defined to be

$$\eta = \int xp(x)dx$$

or, for a discrete distribution:

$$\eta = \sum np(n)$$

In the examples above we have assumed that the mean of the Gaussian distribution to be 0, while the mean of the Poisson distribution is found to be α. The mean of a distribution is, in common sense, the average value taken by the corresponding random variable.

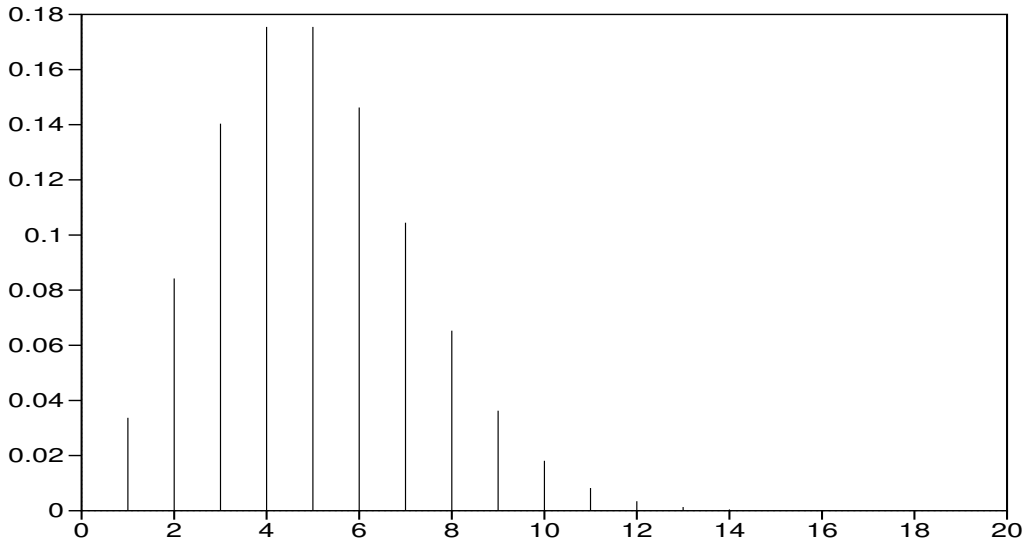

Figure 14: Discrete distribution pdf (Poisson).

The *variance* σ^2 is defined to be

$$\sigma^2 = \int (x-\eta)^2 p(x) dx$$

or, for a discrete distribution,

$$\sigma^2 = \sum (n-\eta)^2 p(n)$$

The square root of the variance is called standard deviation. The standard deviation is a measure of the spread of the probability distribution around the mean. A small standard deviation means the distribution is concentrated about the mean. A large value indicates a wide range of possible outcomes. The Gaussian distribution contains the standard deviation within its definition. The Poisson distribution has a standard deviation of α^2.

In many cases the noise present in communication signals can be modelled as a zero-mean, Gaussian random variable. This means that its amplitude at a particular time has a PDF given by Eq. (2.19) above. The statement that noise is zero mean says that, on average, the noise signal takes the values zero. We have already seen that the signal to noise ratio is an important quantity in determining the performance of a communication channel. The noise power referred to in the definition is the mean noise power. It can therefore be rewritten as

$$SNR = 10 \log_{10}(S/\sigma^2)$$

If only thermal noise is considered, we have $\sigma^2 = kT_mB$ where k is the Boltzman's constant ($k = 1.38 \times 10^{-23} J/K$), T_m is the temperature and B is the receiver bandwidth.

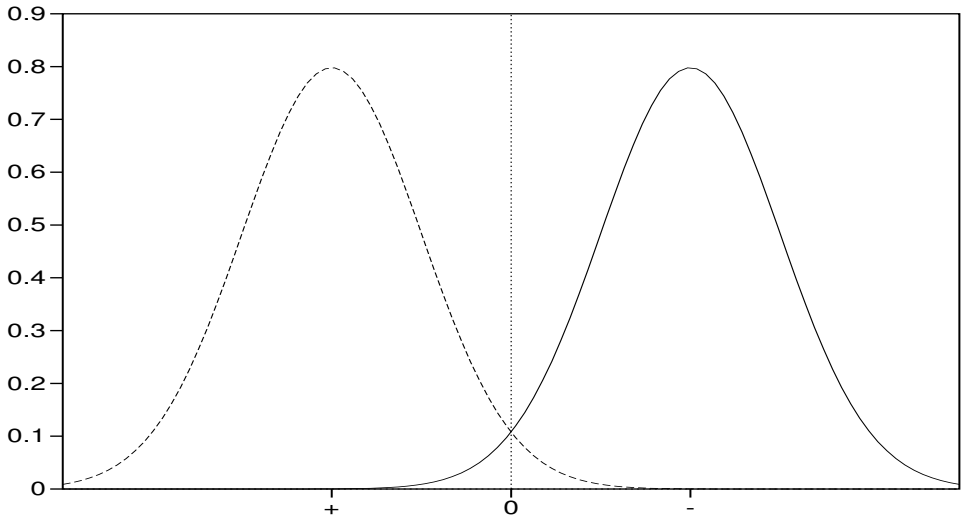

Figure 15: Schematic of noise on a two level line

2.2.6 Errors in digital communication

We noted earlier that one of the most important advantages of digital communications is that it permits very high fidelity. In this subsection we shall investigate this more closely. We shall consider in detail only BPSK systems, and comment on the alternative modulations.

In the absence of noise, the signal V, from a BPSK system can take one of two values $\pm v_b$. In the ideal case, if the signal is greater than 0, the value that is read is assigned to 1. If the signal is less than 0, the value that is read is assigned to 0. When noise is present, this distinction between $\pm v_b$ (with the threshold at 0) becomes blurred. There is a finite probability of the signal dropping below 0, and thus being assigned 0, even though a 1 was transmitted. When this happens, we say that a bit-error has occurred. The probability that a bit-error will occur in a given time is referred to as the bit-error rate (BER) (see Fig. 15).

We suppose that the signal V, which has the signal levels $\pm v_b$, is combined with noise N of variance σ^2. The probability that an error will occur in the transmission of a 1 is

$$P(N + v_b < 0) = P(N < -v_b) = \frac{2}{\sqrt{\pi}} \int_{-\infty}^{-v_b} \exp(-u^2) du = \frac{1}{2}\text{erfc}(v_b/2\sigma)$$

Similarly the probability that an error will occur in the transmission of a 0 is

$$P(N - v_b > 0) = P(N > v_b) = \frac{2}{\sqrt{\pi}} \int_{v_b}^{\infty} \exp(-u^2) du = \frac{1}{2}\text{erfc}(v_b/2\sigma)$$

It is usual to write these expressions in terms of the ratio of E_b (energy per bit) to E_n (noise power per unit Hz). The power S in the signal is, on average v_b^2, and the total energy in the

Modulation scheme	$P\{\text{error}\}$
Incoherent ASK	$\approx \frac{1}{2}\mathrm{e}^{-\frac{E_b}{4E_n}}$
Incoherent FSK	$\frac{1}{2}\mathrm{e}^{-\frac{E_b}{2E_n}}$
BPSK	$\frac{1}{2}\mathrm{erfc}(\sqrt{\frac{E_b}{E_n}})$
DPSK	$\frac{1}{2}\mathrm{e}^{-\frac{E_b}{E_n}}$

Figure 16: Expressions for error rates in some modulation schemes

signalling period T is $v_b^2 T$. Using the expressions above, we have

$$\frac{1}{2}\mathrm{erfc}(v_b/2\sigma) = \frac{1}{2}\mathrm{erfc}(\sqrt{\frac{E_b}{2TE_nB}})$$

where we have used the fact that $\sigma^2 = kT_mB = E_nB$ for temperature T_m.

For BPSK, the signaling period T is half the reciprocal of the bandwidth B, i.e. $T = 1/2B$; thus

$$P(\text{error}) = \frac{1}{2}\mathrm{erfc}(\sqrt{E_b/E_n}) \tag{2.20}$$

All coherent detection schemes give rise to error rates of the form in Eq. (2.20) above. For example, QPSK has twice the error probability of BPSK, reflecting the fact that with a quadrature scheme, there are more ways an error can occur. Narrow-band FSK has an error probability rather worse than QPSK, although its numerical value depends on the exact scheme used. Fig. 17 shows graphs of $P(\text{error})$ for incoherent ASK, incoherent FSK, BPSK, and DPSK; the expressions are given in Table 16.

Incoherent demodulation schemes have a higher probability of error than coherent schemes. Incoherent schemes are forms of power detection, i.e. produce an output proportional to the square of the input. Power detection always decreases the SNR. It is quite easy to see why this is so. Suppose the input, X, is of the form $X = v + N$, as before. The input SNR is

$$SNR_{in} = \frac{v^2}{N^2}$$

If we square the input, the output is

$$X^2 = (v + N)^2$$

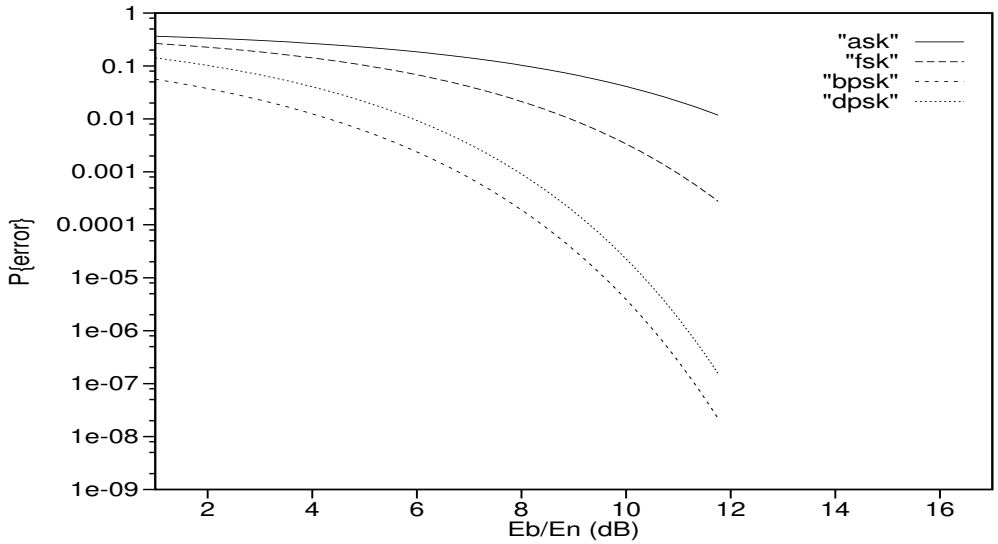

Figure 17: Comparison of error rates in some modulation schemes

Assuming the SNR is high, $vN >> N^2$, and the SNR of the output is

$$SNR_{out} \sim \frac{(v^2)^2}{(2vN)^2} = \frac{SNR_{in}}{4}$$

This decrease in the signal-to-noise ratio causes an increase in the error probability. The detailed analysis is beyond our scope. Although poorer, however, their performance is good nonetheless. This explains the widespread use of incoherent ASK and FSK.

Error rates are usually quoted as bit error rates (BER). The conversion from error probability to BER is numerically simple: BER $= P(\text{error})$. However, this conversion assumes that the probability of errors from bit-to-bit are independent. This may or may not be a reasonable assumption. In particular, loss of timing can cause multiple bit failures that can dramatically increase the BER.

When signals travel along the channel, they are being attenuated. As the signal is losing power, the BER increases with the length of the channel. Regenerators, placed at regular intervals, can dramatically reduce the error rate over long channels. To determine the BER of the channel with N regenerators, it is simple to calculate first the probability of no error. This probability is the probability of no error over one regenerator, raised to the Nth power:

$$P(\text{ No error over N regenerators}) = (1 - P(\text{error}))^N$$

assuming the regenerators are regularly spaced and the probabilities are independent. The BER is then determined simply by:

$$P(\text{ error over N regenerators}) = 1 - P(\text{ no error over N regenerators})$$

This avoids having to enumerate all the ways in which the multiple system can fail.

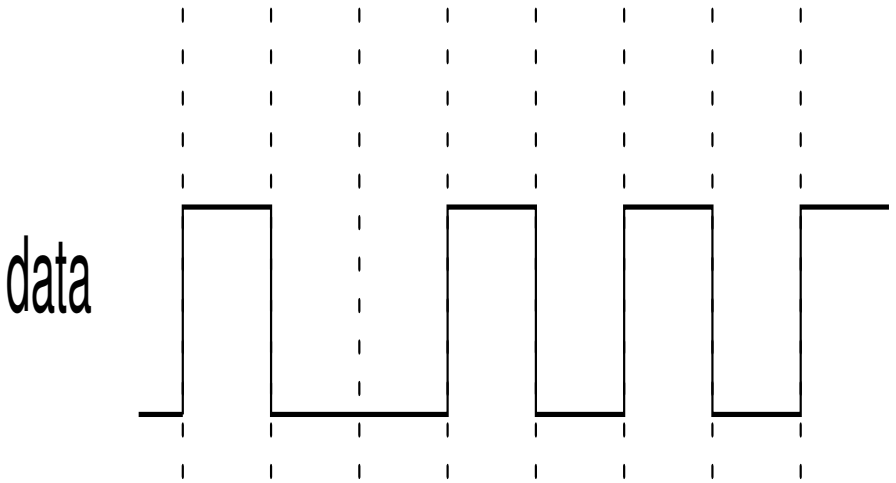

Figure 18: The received signal could be 1001010 or 11000011001100.

2.2.7 Timing control in digital communication

In addition to providing the analogue modulation and demodulation functions, digital communication also requires timing control. Timing control is required to identify the rate at which bits are transmitted and to identify the start and end of each bit. This permits the receiver to correctly identify each bit in the transmitted message. Bits are never sent individually. They are grouped together in segments, called blocks. A block is the minimum segment of data that can be sent with each transmission. Usually, a message will contain many such blocks. Each block is framed by binary characters identifying the start and end of the block.

The type of method used depends on the source of the timing information. If the timing in the receiver is generated by the receiver, separately from the transmitter, the transmission is termed asynchronous. If the timing is generated, directly or indirectly, from the transmitter clock the transmission is termed synchronous.

Asynchronous transmission is used for low data-rate transmission and stand-alone equipment. We will not discuss it in detail here. Synchronous transmission is used for high data rate transmission. The timing is generated by sending a separate clock signal, or embedding the timing information into the transmission. This information is used to synchronize the receiver circuitry to the transmitter clock. The necessity to introduce a clock in signal transmission is obvious if we look at Fig. 18. Without a clock, would we be able to tell whether it is a 1001010 or a 11000011001100?

Synchronous receivers require a timing signal from the transmitter. An additional channel may be used in the system to transmit the clock signal. This is wasteful of bandwidth, and it is more customary to embed the timing signal within the transmitted data stream by use of suitable encoding (self-clocking encoding).

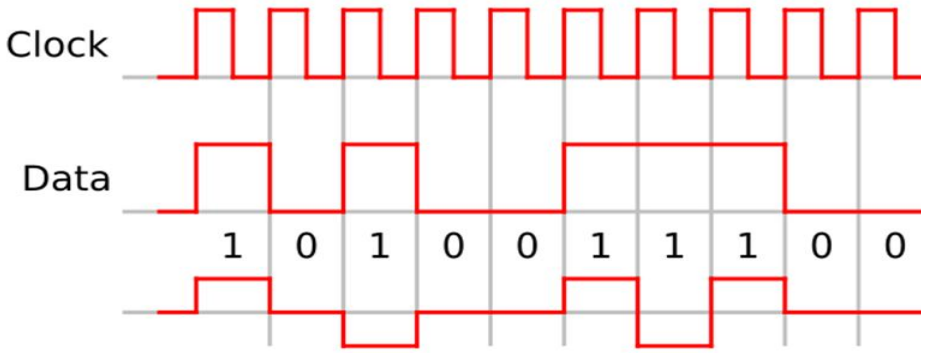

Figure 19: Bipolar coding.

In bipolar coding, a binary 0 is encoded as zero volts. A binary 1 is encoded alternately as a positive voltage and a negative voltage (see Fig. 19). Other systems must synchronize using some form of out-of-band communication, or add frame synchronization sequences that don't carry data to the signal. These alternative approaches require either an additional transmission medium for the clock signal or a loss of performance due to overhead, respectively. A bipolar encoding is an often good compromise: runs of ones will not cause a lack of transitions, however long sequences of zeroes are still an issue. Since the signal doesn't change for as long as the data to send is a zero, they will result in no transitions and a loss of synchronization. Where frequent transitions are a requirement, a self-clocking encoding such as Manchester code discussed below may be more appropriate.

Manchester code (also known as Phase Encoding, or PE) is a form of data communications in which each bit of data is signified by at least one voltage level transition. Manchester encoding is therefore considered to be self-clocking, which means that accurate synchronisation of a data stream is possible. Manchester coding has been adopted into many efficient and widely used telecommunications standards, such as Ethernet.

Here is a summary for Manchester code:

- Data and clock signals are combined to form a single self-synchronizing data stream
- each encoded bit contains a transition at the midpoint of a bit period
- the direction of transition determines whether the bit is a ”0” or a ”1,” and
- the first half is the true bit value and the second half is the complement of the true bit value (see Fig. 20).

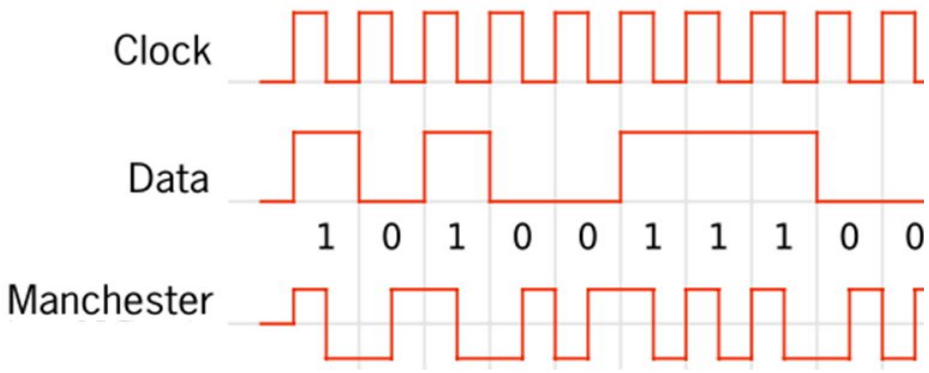

Figure 20: Manchester coding.

Manchester codes always have a transition at the middle of each bit period. The direction of the mid-bit transition is what carries the data, with a low-to-high transition indicating one binary value, and a high-to-low transition indicating the other. Transitions that don't occur mid-bit don't carry useful information, and exist only to place the signal in a state where the necessary mid-bit transition can take place. Though this allows the signal to be self-clocking, it essentially doubles the bandwidth.

However, there are today many more sophisticated codes (8B/10B encoding) which accomplish the same aims with less bandwidth overhead, and less synchronisation ambiguity in pathological cases.

3 Information and coding theory

Information theory is concerned with the description of information sources, the representation of the information from a source, and the transmission of this information over channel. This might be the best example to demonstrate how a deep mathematical theory could be successfully applied to solving engineering problems.

Information theory is a discipline in applied mathematics involving the quantification of data with the goal of enabling as much data as possible to be reliably stored on a medium and/or communicated over a channel. The measure of data, known as information entropy, is usually expressed by the average number of bits needed for storage or communication.

Applications of fundamental topics of information theory include ZIP files (lossless data compression), MP3s (lossy data compression), and DSL (channel coding). The field is at the crossroads of mathematics, statistics, computer science, physics, neurobiology, and electrical engineering. Its impact has been crucial to success of the Voyager missions to deep space, the invention of the CD, the feasibility of mobile phones, the development of the Internet, the study of linguistics and of human perception, the understanding of black holes, and numerous other fields.

Information theory is generally considered to have been founded in 1948 by Claude Shannon in his seminal work, **A Mathematical Theory of Communication**. The central paradigm of classic information theory is the engineering problem of the transmission of information over a noisy channel. The most fundamental results of this theory are Shannon's **source coding theorem**, which establishes that, on average, the number of bits needed to represent the result of an uncertain event is given by its entropy; and Shannon's noisy-channel coding theorem, which states that reliable communication is possible over noisy channels provided that the rate of communication is below a certain threshold called the **channel capacity**. The channel capacity can be approached by using appropriate encoding and decoding systems.

Information theory is closely associated with a collection of pure and applied disciplines that have been investigated and reduced to engineering practice under a variety of rubrics throughout the world over the past half century or more: adaptive systems, anticipatory systems, artificial intelligence, complex systems, complexity science, cybernetics, informatics, machine learning, along with systems sciences of many descriptions. Information theory is a broad and deep mathematical theory, with equally broad and deep applications, amongst which is the vital field of coding theory which is the main focus of our course.

Coding theory is concerned with finding explicit methods, called codes, of increasing the efficiency and reducing the net error rate of data communication over a noisy channel to near the limit that Shannon proved is the maximum possible for that channel. These codes can be roughly subdivided into data compression (source coding) and error-correction (channel coding) techniques. In the latter case, it took many years to find the methods Shannon's work proved were possible. A third class of information theory codes are cryptographic algorithms (both codes and ciphers). Concepts, methods and results from coding theory and information theory are widely used in cryptography and cryptanalysis.

3.1 Information sources and entropy

We start our examination of information theory by way of an example.

Consider predicting the activity of Prime Minister tomorrow. This prediction is an information source. Assume such information source has two outcomes:

- The Prime Minister will be in his office,
- The Prime Minister will be naked and run 10 miles in London.

Clearly, the outcome 'in office' contains little information; it is a highly probable outcome. The outcome 'naked run', however contains considerable information; it is a highly improbable event.

In information theory, an information source is a probability distribution, i.e. a set of probabilities assigned to a set of outcomes. This reflects the fact that the information contained in an outcome is determined not only by the outcome, but by how uncertain it is. An almost certain outcome contains little information.

A measure of the information contained in an outcome was introduced by Hartley in 1927. He defined the information contained in an outcome x_i as

$$I(x_i) = -\log_2 p(x_i)$$

This measure satisfied our requirement that the information contained in an outcome is proportional to its uncertainty. If $P(x_i) = 1$, then $I(x_i) = 0$, telling us that a certain event contains no information.

The definition above also satisfies the requirement that the total information in independent events should add. Clearly, our Prime Minister prediction for two days contains twice as much information as that for one day. For two independent outcomes x_i and x_j,

$$I(x_i \text{ and } x_j) = \log_2 P(x_i \text{ and } x_j) = \log_2 P(x_i)P(x_j) = I(x_i) + I(x_j)$$

Hartley's measure defines the information in a single outcome. The measure entropy $H(X)$ defines the information content of the source X as a whole. It is the mean information provided by the source. We have

$$H(X) = \sum_i P(x_i)I(x_i) = -\sum_i P(x_i)\log_2 P(x_i)$$

A binary symmetric source (BSS) is a source with two outputs whose probabilities are p and $1-p$ respectively. The prime minister discussed is a BSS. The entropy of the source is

$$H(X) = -p\log_2 p - (1-p)\log_2(1-p)$$

The function (Fig. 21) takes the value zero when $p = 0$. When one outcome is certain, so is the other, and the entropy is zero. As p increases, so too does the entropy, until it reaches a maximum when $p = 1 - p = 0.5$. When p is greater than 0.5, the curve declines symmetrically to zero, reached when $p = 1$. We conclude that the average information in the BSS is maximised when both outcomes are equally likely. The entropy is measuring the average uncertainty of the source.

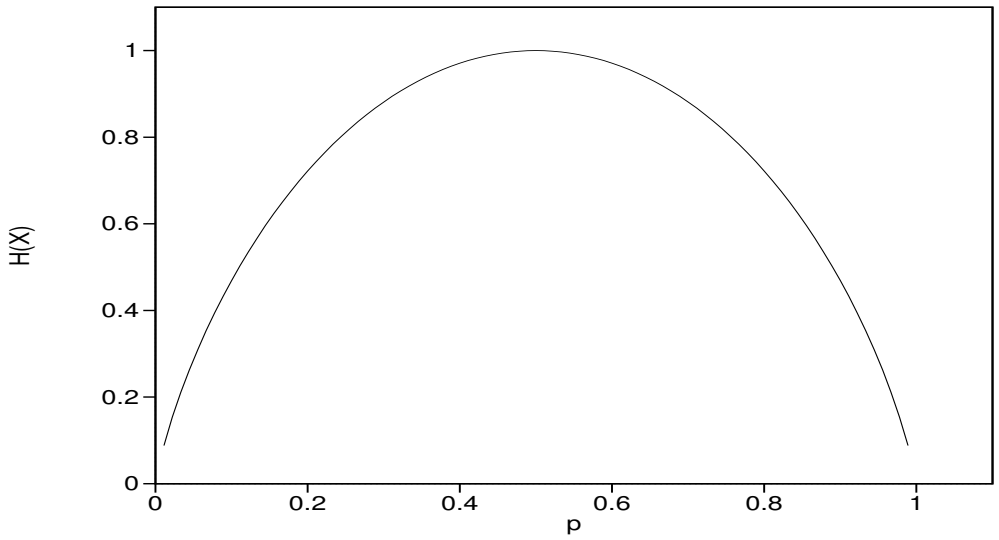

Figure 21: Entropy vs. p

(The term entropy is borrowed from thermodynamics. There too it is a measure of the uncertainty, or disorder of a system).

When $p = 0.5, H(X) = 1$. The unit of entropy is bits/symbol. An equally probable BSS has an entropy, or average information content per symbol, of 1 bit per symbol.

By long tradition, engineers have used the word bit to describe both the symbol, and its information content. A BSS whose output are 1 or 0 has an output we describe as a bit. The entropy of source is also measured in bits, so that we might say the equi-probable BSS has an information rate of 1 bit/bit. The numerator bit refers to the information content, while the denominator bit refers to the symbol 1 or 0. We can avoid this by writing it as 1 bit/symbol. When $p \neq 0.5$, the BSS information rate falls. When $p = 0.1, H(X) = 0.47$ bits/symbol. This means that on average, each symbol (1 or 0) of source output provides 0.47 bits of information.

3.2 Information source coding

It seems intuitively reasonable that an information source of entropy H needs on average only H binary bits to represent each symbol. Indeed, the equi-probable BSS generate on average 1 information bit per symbol bit. However, consider the prime minister example again. Suppose the probability of 'naked run' is 0.1 (N) and that of 'office' is 0.9 (O). We have already noted that this source has an entropy of 0.47 bits/symbol. Suppose we identify 'naked run' with 1 and 'office' with zero. This representation uses 1 binary bit per symbol, hence is using more binary bits per symbol than the entropy suggests is necessary.

The Shannon's first theorem states that an *instantaneous* code can be found that encodes a

Sequence	OOO	OON	ONO	NOO	NNO	NON	ONN	NNN
Probability	0.729	0.081	0.081	0.081	0.009	0.009	0.009	0.001
Codeword	0	1	01	10	11	00	000	111
Codeword Length (in bits)	1	1	2	2	2	2	3	3
Weighted length								
Entropy								

Table 1: Variable length source coding

source of entropy $H(X)$ with an average number of bits per symbol B_s such that

$$B_s \geq H(X)$$

Ordinarily, the longer the sequence of symbols, the closer B_s will be to $H(X)$.

The replacement of the symbols naked run/office with a binary representation is termed source coding. In any coding operation we replace the symbol with a codeword. The purpose of source coding is to reduce the number of bits required to convey the information provided by the information source: minimize the average length of codes.

Central to source coding is the use of sequence. By this, we mean that codewords are not simply associated to a single outcome, but to a sequence of outcomes. To see why this is useful, let us return to the problem of the Prime Minister. Suppose we group the outcomes in three, according to their probability, and assign binary codewords to these grouped outcomes. Table 1 shows such a code, and the probability of each codeword occurring. It is easy to compute that this code will on average use 1.4 bits/sequence

$$0.729*\log_2(0.729)+0.081*\log_2(0.081)*3+0.009*\log_2(0.009)*3+0.001*\log_2(0.001) = -1.4070$$

The average length of coding is given by

0.729*1+0.081*1+2*0.081*2+2*0.009*2+3*0.009+3*0.001=1.2

This example shows how using sequences permits us to decrease the average number of bits per symbol. Moreover, without difficulty, we have found a code that has an average bit usage less than the source entropy. However, there is a difficulty with the code in Table 1. Before a code can be decoded, it must be parsed. Parsing describes that activity of breaking the message string into its component codewords. After parsing, each codeword can be decoded into its symbol sequence. An instantaneously parsable code is one that can be parsed as soon as the last bit of a codeword is received. An instantaneous code must satisfy the prefix condition: that no codeword may be a prefix of any other codeword. This condition is not satisfied by the code in Table 1.

3.2.1 Huffman coding

The code in Table 2, however, is an instantaneously parsable code. It satisfies the prefix condition.

As a consequence of Shannon's Source coding theorem, the entropy is a measure of the smallest codeword length that is theoretically possible for the given alphabet with associated weights. In

Sequence	A	B	C	D	E	F	G	H
Probability	0.729	0.081	0.081	0.081	0.009	0.009	0.009	0.001
Codeword	1	011	010	001	00011	00010	00001	00000
Codeword Length (in bits)	1	3	3	3	5	5	5	5
Weighted length								
Entropy								

Table 2: OOO=A, OON=B, ONO=C, NOO=D, NNO=E, NON=F, ONN=G, NNN=H

this example, the weighted average codeword length is 1.59, only slightly larger than the calculated entropy of 1.6 bits per symbol. So not only is this code optimal in the sense that no other feasible code performs better, but it is very close to the theoretical limit established by Shannon.

The code in Table 2 uses

$$0.729 * 1 + 0.081 * 3 * 3 + 0.009 * 5 * 3 + 0.001 * 5 = 1.5980$$

bits per sequence. In fact, this is the Huffman code for the sequence set. We might conclude that there is little point in expending the effort in finding a code better than the Huffman code. The codeword for each sequence is found by generating the Huffman code tree for the sequence. A Huffman code tree is an unbalanced binary tree.

History

In 1951, David Huffman and his MIT information theory classmates were given the choice of a term paper or a final exam. The professor, Robert M. Fano, assigned a term paper on the problem of finding the most efficient binary code. Huffman, unable to prove any codes were the most efficient, was about to give up and start studying for the final when he hit upon the idea of using a frequency-sorted binary tree and quickly proved this method the most efficient.

In doing so, the student outdid his professor, who had worked with information theory inventor Claude Shannon to develop a similar code. Huffman avoided the major flaw of the suboptimal Shannon-Fano coding by building the tree from the bottom up instead of from the top down.

Problem definition

Given a set of symbols and their probabilities, find a prefix-free binary code with minimum expected codeword length.

The derivation of the Huffman code tree is shown in Fig. 22 and the tree itself is shown in Fig. 23. In both these figures, the letters A to H have been used in place of the sequences in Table 2 to make them easier to read.

Note that Huffman coding relies on the use of bit patterns of variable length. In most data communication systems, the data symbols are encoded as bit pattern of a fixed length, i.e. 8 bits. This is done for technical simplicity. Often, coding scheme such as Huffman coding are used on a source symbol set to produce variable bit length coding and are referred to as compression algorithms.

In Fig. 22 the sequences are ordered with respect to the probability of the sequence occurring, with the highest probability at the top of the list. The tree is derived bottom up, in terms of branch nodes and leaf nodes by combining probabilities and removing leaf nodes in progressive stages.

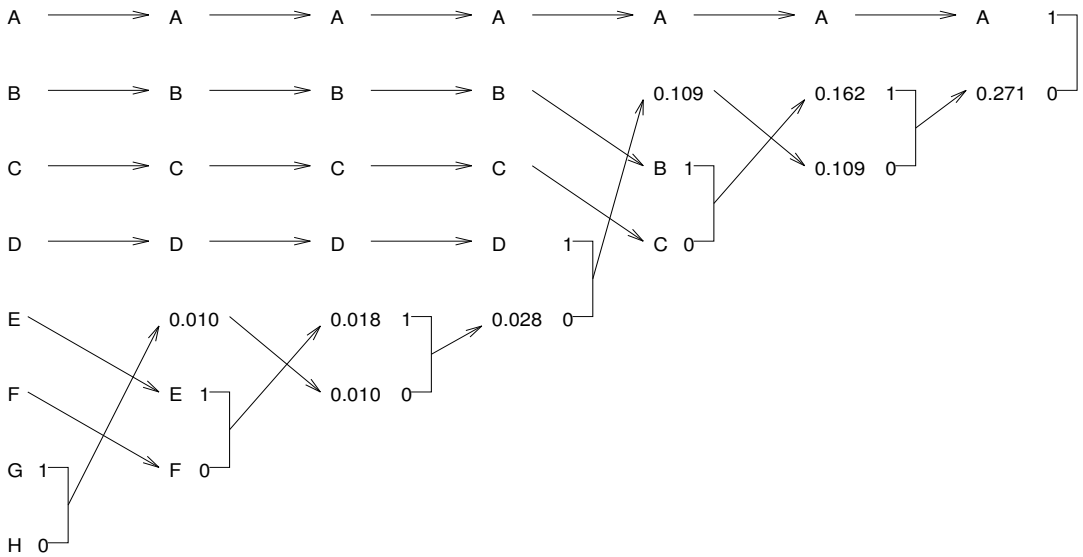

Figure 22: Example derivation of a Huffman code tree

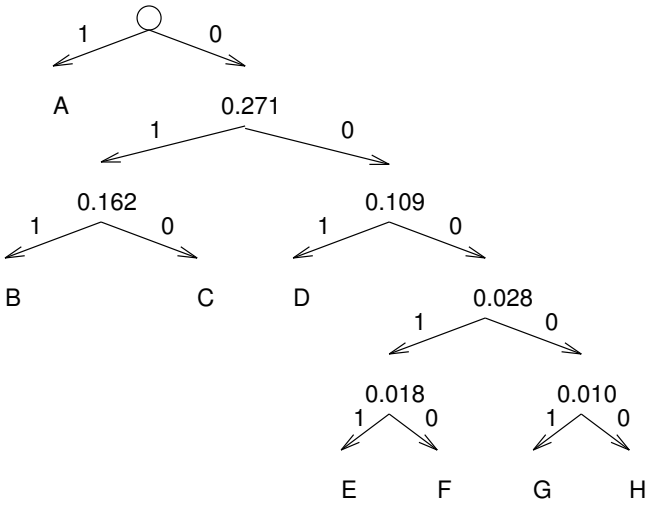

Figure 23: Example Huffman code tree

As shown in Fig. 22, the two lowest leaf nodes G and H have their weights added, and the topmost node is labelled with a 1 and the lower one with a 0. In the next stage the symbol G and H are represented by that with the lowest weight and the list is rewritten, again in order of the weights. The two lowest leaf nodes are now E and F, and they are labelled 1 and 0, respectively, and their weights are added to be taken onto the next stage. This continues until only two nodes remain. The Huffman tree shown in Fig. 23 is then produced by following backwards along the arrows. To derive the codewords from the tree, descend from the top node, and list the 1s and 0s in the order they appear until you reach the leaf node for one of the letters.

We summarize it here.

Creating the tree:

1. Start with as many leaves as there are symbols.
2. Queue all leaf nodes into the first queue (in order).
3. While there is more than one node in the queues:
   - Remove two nodes with the lowest weight from the queues.
   - Create a new internal node, with the two just-removed nodes as children (either node can be either child) and the sum of their weights as the new weight.
   - Update the parent links in the two just-removed nodes to point to the just-created parent node.
4. Queue the new node into the second queue.
5. The remaining node is the root node; the tree has now been generated.

3.3 Channel Capacity

One of the most famous results of information theory is Shannon's channel coding theorem. For a given channel there exists a code that will permit the error-free transmission across the channel at a rate R, provided $R < C$, the channel capacity. Equality is achieved only when the SNR is infinity.

As we have already noted, the astonishing part of the theory is the existence of a channel capacity. Shannon's theorem is both tantalizing and frustrating. It offers error-free transmission, but it makes no statements as to what code is required. In fact, all we may deduce from the proof of the theorem is that is must be a long one. No none has yet found a code that permits the use of a channel at its capacity. However, Shannon has thrown down the gauntlet, in as much as he has proved that the code exists.

We shall not give a description of how the capacity is calculated. However, an example is instructive. The binary channel is a channel with a binary input and output. Associated with each output is a probability p that the output is correct, and a probability $1 - p$ that it is not. For such a channel, the channel capacity turns out to be:

$$C = 1 + p \log_2 p + (1 - p) \log_2 (1 - p)$$

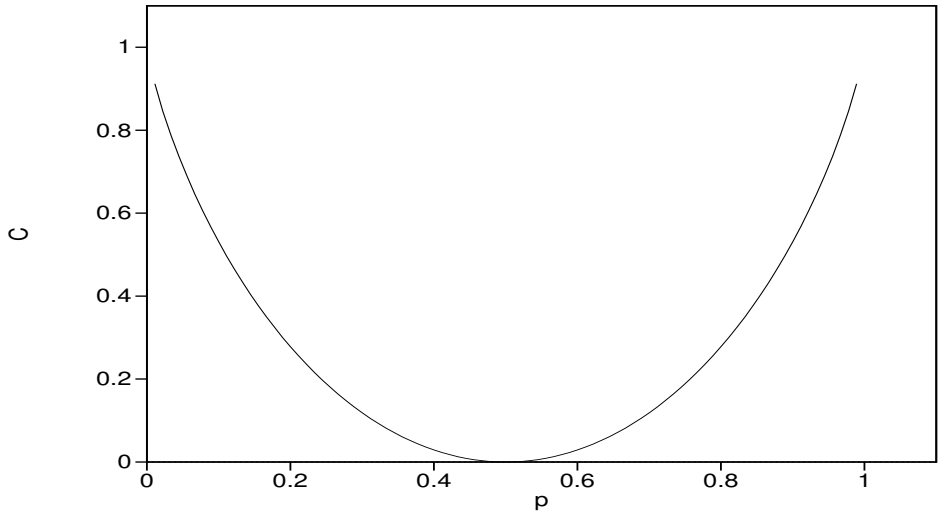

Figure 24: Channel capacity.

Here (Fig. 24), p is the bit error probability. If $p = 0$, then $C = 1$. If $p = 0.5$, then $C = 0$. Thus if there is an equal probability of receiving a 1 or a 0, irrespective of the signal sent, the channel is completely unreliable and no message can be sent across it.

So defined, the channel capacity is a non-dimensional number. We normally quote the capacity as a rate, in bits/second. To do this we relate each output to a change in the signal. For a channel of bandwidth B, we can transmit at most $2B$ changes per second. Thus the capacity in bits/second in $2BC$. For the binary channel we have

$$C = B[1 + p \log_2 p + (1 - p) \log_2(1 - p)]$$

For the binary channel the maximum bit rate W is $2B$. We note that $C < W$, i.e. the capacity is always less than the bit rate. The data rate D, or information rate, describes the rate of transfer of data bits across the channel. In theory we have

$$W > C > D$$

Shannon's channel coding theorem applies to the channel, not to the source. If the source is optimally coded, we can rephrase the channel coding theorem: A source of information with entropy $H(X)$ can be transmitted error free over a channel provided $H(x) \leq C$.

3.4 Error detection coding

3.4.1 Hamming distance

The task of source coding is to represent the source information with the minimum number of symbols. When a code is transmitted over a channel in the presence of noise, errors will occur.

The task of channel coding is to represent the source information in a manner that minimises the probability of errors in decoding.

It is apparent that channel coding requires the use of redundancy. If all possible outputs of the channel correspond uniquely to a source input, there is no possibility of detecting errors in the transmission. To detect, and possibly correct errors, the channel code sequence must be longer the the source sequence. The rate R of a channel code is the average ratio of the source sequence length to the channel code length. Thus $R < 1$.

A good channel code is designed so that, if a few errors occur in transmission, the output can still be decoded with the correct input. This is possible because although incorrect, the output is sufficiently similar to the input to be recognisable. The idea of similarity is made more firm by the definition of the Hamming distance. Let x and y be two binary sequences of same length. The Hamming distance between these two codes is the number of symbols that disagree.

For example:

- The Hamming distance between 1011101 and 1001001 is 2.
- The Hamming distance between 2143896 and 2233796 is 3.
- The Hamming distance between "toned" and "roses" is 3.

Suppose the code x is transmitted over the channel. Due to error, y is received. The decoder will assign to y the code x that minimises the Hamming distance between x and y. For example, consider the codewords:

$$a = (100000), b = (011000), c = (000111)$$

if the transmitter sends 10000 but there is a signal bit error and the receiver gets 10001, it can be seen that the nearest codeword is in fact 10000 and so the correct codeword is found.

It can be shown that to detect n bit errors, a coding scheme requires the use of codewords with a Hamming distance of at least $n + 1$. It can be also shown that to correct n bit errors requires a coding scheme with a least a Hamming distance of $2n + 1$ between the codewords.

By designing a good code, we try to ensure that the Hamming distance between the possible codewords x is larger than the Hamming distance arising from errors.

3.4.2 Parity Check Codes

The theoretical limitations of coding are placed by the results of information theory. These results are frustrating in that they offer little clue as to how the coding should be performed. Errors occur– they must, at the very least, be detected.

Error detection coding is designed to permit the detection of errors. Once detected, the receiver may ask for a retransmission of the erroneous bits, or it may simply inform the recipient that the transmission was corrupted. In a binary channel, error checking code are called *parity check codes*. Practical codes are normally block codes. A block code converts a fixed length of K data bits to a fixed length N code word, where $N > K$. The rate of the code is the ratio K/N, and

the redundancy of the code is $1 - K/N$. Our ability to detect errors depends on the rate. A low rate has a high detection probability, but a high redundancy. The receiver will assign to the received codeword the preassigned codeword that minimises the Hamming distance between the two words. If we wish to identify any pattern of n or less errors, the Hamming distance between the preassigned codes must be $n + 1$ or greater.

A very common code is the single parity check code. This code appends to each K data bits an additional bit whose value is taken to make the $K + 1$ word even or odd. Such a choice is said to have even (odd) parity. With even (odd) parity, a single bit error will make the received word odd (even). The preassigned code words are always even (odd), and hence are separated by a Hamming distance of 2 or more.

To see how the addition of a parity bit can improve error performance, consider the following example. A common choice of K is eight. Suppose that BER is $p = 10^{-4}$. Then

$$\begin{aligned} P(\text{ single bit error }) &= p \\ P(\text{ no error in single bit }) &= 1 - p \\ P(\text{ no error in 8 bits }) &= (1 - p)^8 \\ P(\text{ unseen error in 8 bits }) &= 1 - (1 - p)^8 = 7.9 \times 10^{-4} \end{aligned}$$

So, the probability of a transmission with an error is as above. With the additional of a parity error bit we can detect any single bit error. So:

$$\begin{aligned} P(\text{ no error single bit }) &= 1 - p \\ P(\text{ no error in 9 bits }) &= (1 - p)^9 \\ P(\text{ single error in 9 bits }) &= 9[P(\text{ single bit error })P(\text{no error in other 8 bits})] \\ &= 9p(1 - p)^8 \\ P(\text{ unseen error in 9 bits }) &= 1 - P(\text{no error in 9 bits}) - P(\text{single error in 9 bits}) \\ &= 1 - (1 - p)^9 - 9p(1 - p)^8 \\ &= 3.6 \times 10^{-7} \end{aligned}$$

As can be seen the addition of a parity bit has reduced the uncorrected error rate by three orders or magnitude.

Single parity bits are common in asynchronous, character oriented transmission. Where synchronous transmission is used, additional parity symbols are added that check not only the parity of each 8 bit row, but also the parity of each 8 bit column. The column is formed by listing each successive 8 bit word one beneath the other. This type of parity checking is called block sum checking, and it can correct any single 2 bit error in the transmitted block of rows and columns. However, there are some combinations of errors that will go undetected in such a scheme (see Table. 3)

Parity checking in this way provides good protection against single and multiple errors when the probability of the errors are independent. However, in many circumstances, errors occur in groups, or bursts. Parity checking of the kind just described then provides little protection. In these circumstances, a polynomial code is used.

	p1	B6	B5	B4	B3	B2	B1	B0
	0	1	0	0	0	0	0	0
	1	0	1	0	1	0	0	0
	0	1	0 (*)	0	0	1 (*)	1	0
	0	0	1	0	0	0	0	0
	1	0	1	0	1	1	0	1
	0	1	0	0	0	0	0	0
	1	1	1 (*)	0	0	0 (*)	1	1
	1	0	0	0	0	0	1	1
p2	1	1	0	0	0	0	0	1

Table 3: p1 is odd parity for rows; p2 is even parity for columns (*) mark undetected error combination.

The mechanism of polynomial codes is beyond the scope of this course. We shall not discuss it in details.

Error correction coding is more sophisticated than error detection coding. Its aim is to detect and locate errors in transmission. Once located, the correction is trivial: the bit is inverted. Error correction coding requires lower rate codes than error detection, often markedly so. It is therefore uncommon in terrestrial communication, where better performance is usually obtained with error detection and retransmission. However, in satellite communication, the propagation delay often means that many frames are transmitted before an instruction to retransmit is received. This can make the task of data handling very complex. Real-time transmission often precludes retransmission. It is necessary to get it right first time. In these special circumstances, the additional bandwidth required for the redundant check-bits is an acceptable price. There are two principle types: Hamming codes and convolutional codes. Again we will not discuss them in details here.

3.5 Encryption

In all our discussion of coding, we have not mentioned what is popularly supposed to be the purpose of coding: security. We have only considered coding as a mechanism for improving the integrity of the communication system in the presence of noise. The use of coding for security has a different name: encryption. The use of digital computers has made highly secure communication a normal occurrence. The basis for key based encryption is that is very much easier to encrypt with knowledge of the key than it is to decipher without knowledge of the key. The principle is just that of a combination lock. With a computer the number of the digits in the lock can be very large. Of course, one still has to keep the combination secure.

The most commonly used encryption algorithms are block ciphers. This means that the algorithm splits the plain text (message to be encrypted) into (usually) fixed size blocks which are then subjected to various functions to produce a block of cipher text. The most common functions are permutations based on expansion and compression and straight shuffling transformations. In a straight permutation, the bits of an n bit block are simply reordered. In expansion, as well as being

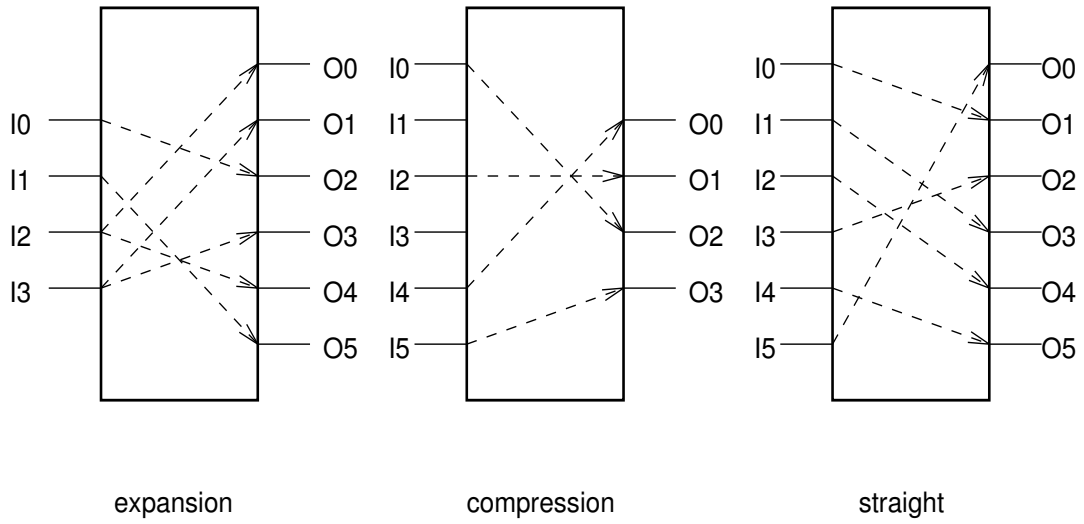

Figure 25: Examples of block cipher permutations.

reordered, the grouped of n bits is converted to m bits ($m > n$), with some bits being duplicated. In compression, the n bit block in converted to a p bits ($p < n$), with some of the original bits unused (see Fig. 25).

The most widely used form of encryption is defined by the National Bureau of Standards and is known as the data encryption standard (DES). The DES is a block cipher, splitting the data stream into 64-bit blocks that are enciphered separately. A unique key of 56 bits is then used to perform a succession of transposition and substitution operations. A 56 bit key has 7.2^{16} possible combinations. Assuming a powerful computer could attempt 10^8 a combinations per second, it would still take over 20 years to break the code. If the code is changed once per year, there is little possibility of it being broken, unless the code breaker had additional information. The DES converts 64 bits of of plain text into 64 bits of cipher text. The receiver uses the same key to decipher the cipher text into plain text.

The difficulty with this method is that each block is independent. This permits an interceptor in possession of the key to introduce additional blocks without the recipient being aware of this fact.

Like the combination of a lock, the system is only secure if the key is secure. If the key is changed often, the security of the key becomes a problem, because the transfer of the key between sender and receiver may not be secure. This is avoided by the use of matched keys. In a matched key scheme, the encryption is not reversible with the same key. The message is encrypted using one key, and decrypted with a second, matched key. The receiver makes available the first, public key. This key is use by the sender to encrypt the message. This message is unintelligible to anyone not in possession of the second, private key. In this way the private key needs not be transferred. The most famous of such scheme is the Public Key mechanism using the work of Rivest, Shamir and Adleman (RSA). It is based on the use of multiplying extremely large numbers and, with current

technology, is computationally very expensive.

RSA numbers are composite numbers having exactly two prime factors that have been listed in the Factoring Challenge of RSA Security? and have been particularly chosen to be difficult to factor. While RSA numbers are much smaller than the largest known primes, their factorization is significant because of the curious property of numbers that proving or disproving a number to be prime (primality testing) seems to be much easier than actually identifying the factors of a number (prime factorization).

Thus, while it is trivial to multiply two large numbers and together, it can be extremely difficult to determine the factors if only their product is given.

With some ingenuity, this property can be used to create practical and efficient encryption systems for electronic data. RSA Laboratories sponsors the RSA Factoring Challenge to encourage research into computational number theory and the practical difficulty of factoring large integers, and because it can be helpful for users of the RSA encryption public-key cryptography algorithm for choosing suitable key lengths for an appropriate level of security.

A cash prize is awarded to the first person to factor each challenge number.RSA numbers were originally spaced at intervals of 10 decimal digits between 100 and 500 digits, and prizes were awarded according to a complicated formula.

A list of the open Challenge numbers may be downloaded from RSA homepage.

Number digits	prize (USD)	factored (references)
$RSA-100$	100	Apr. 1991
$RSA-110$	110	Apr. 1992
$RSA-120$	120	Jun. 1993
$RSA-129$	129	Apr. 1994 (Leutwyler 1994, Cipra 1995)
$RSA-130$	130	Apr. 10, 1996
$RSA-140$	140	Feb. 2, 1999 (te Riele 1999)
$RSA-150$	150	Apr. 6, 2004 (Aoki 2004)
$RSA-155$	155	Aug. 22, 1999 (te Riele 1999, Peterson 1999)
$RSA-160$	160	Apr. 1, 2003 (Bahr et al. 2003)
$RSA-200$	200	May 9, 2005 (Weisstein 2005)
$RSA-576$	10000	Dec. 3, 2003 (Franke 2003; Weisstein 2003)
$RSA-640$	20000	Nov. 4, 2005 (Weisstein 2005)
$RSA-704$	30000	open
$RSA-768$	50000	open
$RSA-896$	75000	open
$RSA-102$	100000	open
$RSA-153$	150000	open
$RSA-204$	200000	open

Example 3 *In order to see all this in action, we want to stick with numbers that we can actually work with.*

Finding RSA numbers

So we have 7 for P, our public key, and 23 for Q, our private key (RSA number, very small).

Encoding

We create the following character set:

2	3	4	6	7	8	9	12	13	14	16	17	18
A	B	C	D	E	F	G	H	I	J	K	L	M
19	21	23	24	26	27	28	29	31	32	34	36	37
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
38	39	41	42	43	46	47	48	49	51	52	53	
sp	0	1	2	3	4	5	6	7	8	9	?	

The message we will encrypt is ”VENIO” (Latin for ”I come”):

V	E	N	I	O
31	7	19	13	21

To encode it, we simply need to raise each number to the power of P modulo R=55.

$$\begin{aligned}
V: &\quad 31^7 (\text{mod } 55) = 27512614111 (\text{mod } 55) = 26 \\
E: &\quad 7^7 (\text{mod } 55) = 823543 (\text{mod } 55) = 28 \\
N: &\quad 19^7 (\text{mod } 55) = 893871739 (\text{mod } 55) = 24 \\
I: &\quad 13^7 (\text{mod } 55) = 62748517 (\text{mod } 55) = 7 \\
O: &\quad 21^7 (\text{mod } 55) = 1801088541 (\text{mod } 55) = 21
\end{aligned}$$

So, our encrypted message is $26, 28, 24, 7, 21$ – or RTQEO in our personalized character set. When the message ”RTQEO” arrives on the other end of our insecure phone line, we can decrypt it simply by repeating the process – this time using Q, our private key, in place of P.

$$\begin{aligned}
R: &\quad 26^{23} (\text{mod } 55) = 350257144982200575261531309080576 (\text{mod } 55) = 31 \\
T: &\quad 28^{23} (\text{mod } 55) = 19259043800372760688541191131627 52 (\text{mod } 55) = 7 \\
Q: &\quad 24^{23} (\text{mod } 55) = 55572324035428505185378394701824 (\text{mod } 55) = 19 \\
E: &\quad 7^{23} (\text{mod } 55) = 2736874734008091 6343 (\text{mod } 55) = 13 \\
O: &\quad 21^{23} (\text{mod } 55) = 25765808751082182919290758696 61 (\text{mod } 55) = 21
\end{aligned}$$

The result is 31,7,19,13,21 – or ”VENIO”, our original message.

4 Signal Representation

In this Chapter, we are going to present a detailed discussion on Sampling theorem mentioned before: how fast to sample an analogous signal so that we could recover the signal perfectly. We will also introduce some basic tools which are essential for our later part of the module, including z transform, discrete time Fourier transform (DTFT) and discrete Fourier transform (DFT).

4.1 Sequences and their representation

A sequence is an infinite series of real numbers $\{x(n)\}$, which is written

$$\{x(n)\} = \{\cdots, x(-1), x(0), x(1), x(2), \cdots, x(n) \cdots\}$$

This can be used to represent a sampled signal, i.e. $x(n) = x(nT)$, where $x(t)$ is the original (continuous) function of time. Sometimes sequence elements are subscripted, x_n being used in place of $x(n)$.

The most basic tool in DCSP is the Z transform (ZT), which is related to the generating function used in the analysis of series. In mathematics and signal processing, the Z-transform converts a discrete time domain signal, which is a sequence of real numbers, into a complex frequency domain representation. The ZT of $\{x(n)\}$ is

$$X(z) = \sum x(n) z^{-n}$$

where the variable z is a complex number in general and $\sum$ is $\sum_{n=-\infty}^{\infty}$. The first point to note about ZT's is that some sequences have simple rational ZT's.

Example 4

$$\{x(n)\} = \{1, r, r^2, \cdots\}, n \geq 0 \textit{ and } x(n) = 0, n < 0$$

has ZT

$$X(z) = \sum r^n z^{-n}$$

which can be simplified if $r/z < 1$ *to give*

$$X(z) = 1/(1 - rz^{-1})$$

To check that this is correct, we simply invert the ZT by long division

$$X(z) = 1 + rz^{-1} + rz^{-2} + \cdots$$

which obviously corresponds to the original sequence.

The ZT's main property concerns the effects of a shift of the original sequence. Consider the ZT

$$Y(z) = \sum x(n) z^{-(m+n)}$$

which can be written

$$Y(z) = z^{-m}X(z)$$

This obviously corresponds to the sequence $\{y(n)\} = \{x(n-m)\}$.

Now if we add sequences $\{a(n)\}, \{b(n)\}$, we get a sequence

$$\{c(n)\} = \{\cdots, a(-1)+b(-1), a(0)+b(0), a(1)+b(1), \cdots\}$$

with ZT

$$C(z) = \sum (a(n)+b(n))z^{-n} = A(z) + B(z)$$

which is just the sum of the ZT's of $\{a(n)\}, \{b(n)\}$, i.e. the ZT is linear.

Now consider the product of two ZT's

$$C(z) = A(z)B(z)$$

for example. The question arises as to what sequence this represents. If we write it out in full

$$C(z) = \sum a(n)z^{-n} \sum b(m)z^{-m}$$

which can be rewritten

$$C(z) = \sum a(n)b(m)z^{-(m+n)}$$

which is the ZT of the sequence

$$\{c(n)\} = \sum a(m)b(n-m)$$

Sequences $\{c(n)\}$ of this form are called the convolution of the two component sequences $a(n), b(n)$, and are sometimes written as

$$c(n) = a(n) * b(n)$$

Convolution describes the operation of a digital filter, as we shall see in due course. The fundamental reason why we use ZT is that convolution is reduced to multiplication and this is a consequence of the even more basic shift property expressed in the equation above.

Example 5 *(Discrete time unit impulse)*

The unit impulse (Fig. 26) $\delta(n)$ is the most elementary signal, and it provides the simplest expansion. It is defined as

$$\delta(n) = \begin{cases} 1 & \textit{if } n = 0 \\ 0 & \textit{otherwise} \end{cases} \tag{4.1}$$

Any discrete signal can be expanded into the superposition of elementary shifted impulse, each one representing each of the samples. This is expressed as

$$x(n) = \sum x(k)\delta(n-k)$$

where each term $x(k)\delta(n-k)$ in the summation expresses the nth sample of the sequence.

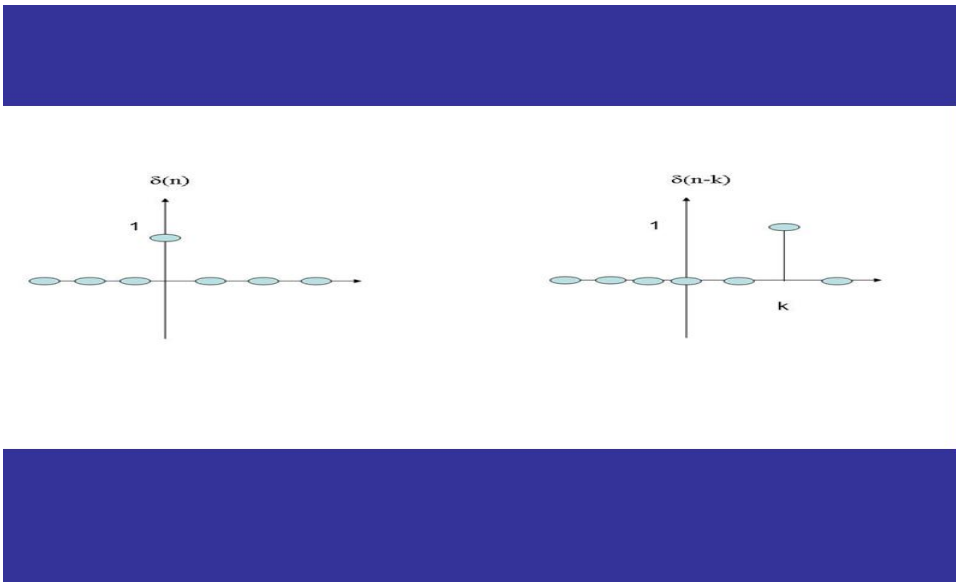

Figure 26: An impulse $\delta(n)$ and a shifted impulse $\delta(n-k)$

4.2 Discrete Time Fourier Transform (DTFT)

The basic tool of signal analysis is the Fourier transform, which is treated in detail in a number of references and revisited before. Although the Fourier transform is not the only transform, it is the one most widely used as a tool in signal analysis. Most likely, you have seen the Fourier transform in its symbolic formulation applied to signals expressed mathematically. For example, we know what the Fourier transform of a rectangular pulse, of a sinusoid, or of a decaying exponential is. However, in most applications of interest, the goal is to determine the frequency content of a signal from a finite set of samples stored on a disk or a tape.

The discrete Fourier transform (DFT), is the algorithm we use for numerical computation. With the DFT we compute and estimate of the frequency spectrum of any sort of data set stored as an array of numerical entries. In this chapter we quickly review the Fourier transform for discrete time signal (the DTFT, discrete time Fourier transform) and present the DFT in detail. In particular, we will be concentrating on two classes of applications: (1) spectral estimation, and (2) compression. The latter involves a variant of the DFT called the discrete cosine transform (or DCT), which is particularly well behaved when applied to approximation problems.

Definition and properties:

The DTFT gives the frequency representation of a discrete time sequence with infinite length. By definition

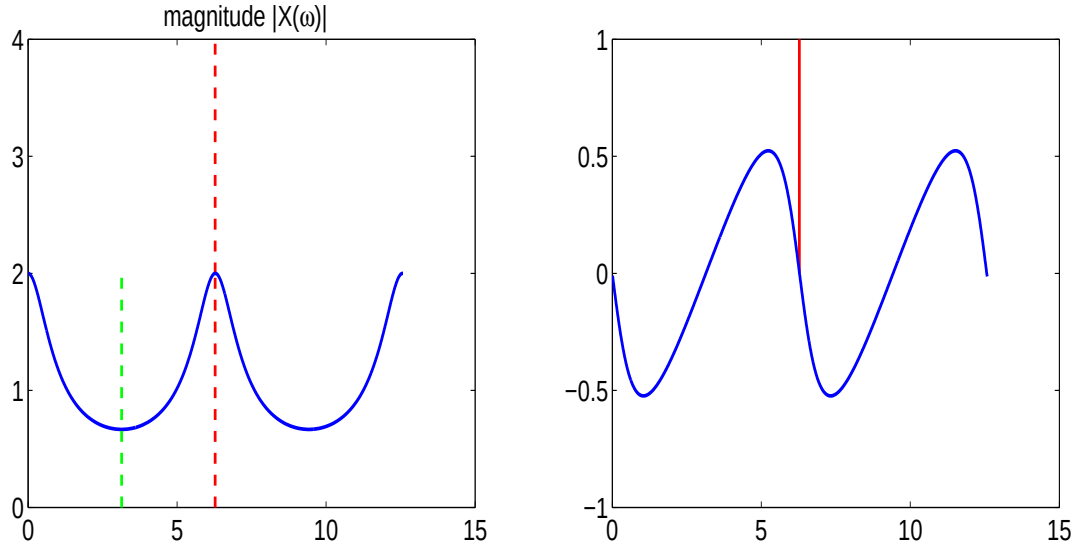

Figure 27: DTFT magnitude and phase (right).

$$\begin{cases} X(\omega) &= DTFT\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n)\exp(-j\omega n),\ \pi \leq \omega < \pi \\ x(n) &= IDTFT(X(\omega)) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\omega)\exp(j\omega n)d\omega,\ -\infty < n < \infty \end{cases} \tag{4.2}$$

The inverse discrete time Fourier transform (IDTFT) can be easily determined by substituting the expression of the DTFT, which yields

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} X(\omega)\exp(j\omega n)d\omega = \sum_{m=-\infty}^{\infty} x(m)\left(\frac{1}{2\pi}\int_{-\pi}^{\pi}\exp(j\omega(n-m))d\omega\right) = x(n)$$

where we used the fact that the term in brackets in the expression above is 1 when $n = m$ and 0 for all other cases.

By definition, $X(\omega)$ is always periodic with period 2π, since

$$X(\omega + 2\pi) = X(\omega)$$

and this is the reason why all the information is included within one period, say in the interval $-\pi \leq \omega < \pi$, as shown in Fig. (27)

Example 6 *(Fig. 28) consider the signal* $x(n) = 0.5^n, n = 0, 1, 2, \cdots$ *then*

$$X(\omega) = \sum_{n=-\infty}^{\infty} 0.5^n \exp(-j\omega n) = \frac{1}{1 - 0.5\exp(-j\omega)}$$

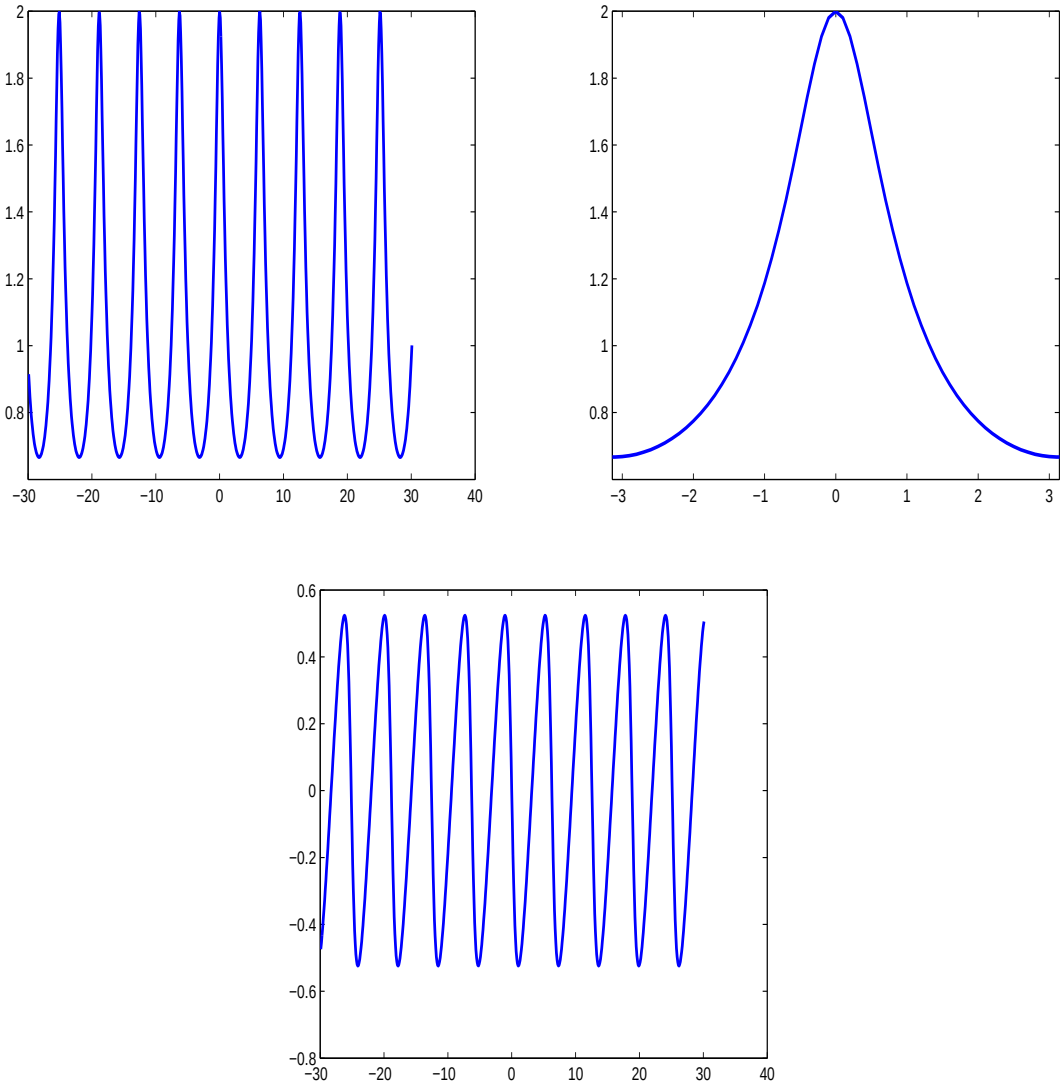

Figure 28: Upper panel (left), Magnitude of DTFT of $X(\omega)$, right, one period of it. Bottom panel, phase of DTFT of $X(\omega)$.

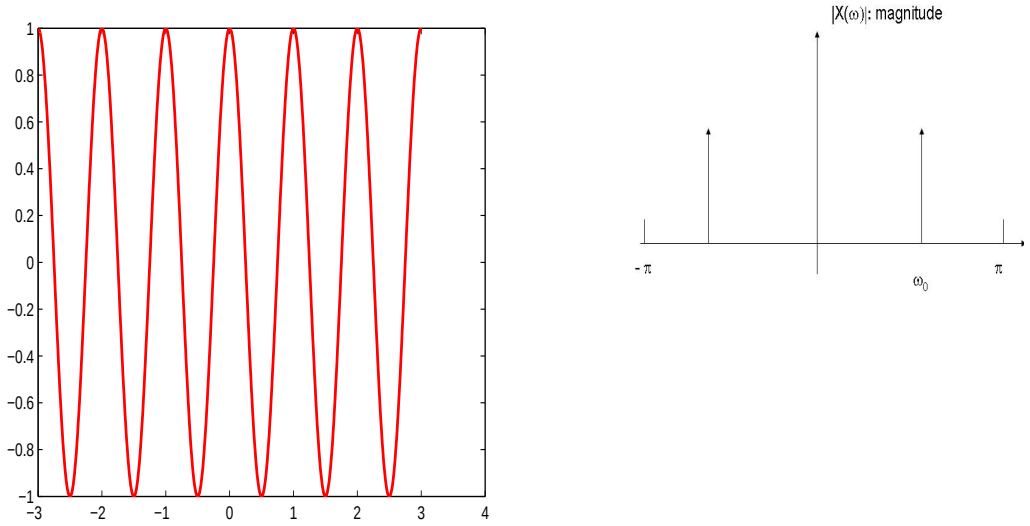

Figure 29: Sinusoid and its DTFT

Example 7 *Consider the signal* $x(n) = \delta(n)$ *then*

$$X(\omega) = \sum_{n=-\infty}^{\infty} \delta(n) \exp(-j\omega n) = 1$$

4.2.1 Computation of the DTFT

It is well known that the whole Fourier approach to signal analysis is based on the expansion of a signal in terms of sinusoids or, more precisely complex exponentials. In this approach we begin analyzing a signal by determining the frequencies contributing to its spectrum in terms of magnitudes and phases.

For example, if a sequence is a sinusoid $x(n) = \cos(\omega_0 n)$ of infinite length, its DTFT yields two 'delta' functions, $X(\omega) = \pi\delta(\omega - \omega_0) + \pi\delta(\omega + \omega_0)$, as in Fig. 29 where we assume all frequencies ω and ω_0 to be within the intervals $[-\pi, \pi)$. This shows that the DTFT gives perfect frequency localization as an exact concentration of energy at $\pm\omega_0$ provided (a) the sequence lasts from $-\infty$ to ∞ and we have an infinite amount of memory to actually collect all the data points, and (b) we compute $X(\omega)$ for all possible frequencies ω in the interval $[-\pi, \pi)$, again requiring an infinite amount of memory and an infinite computational time.

In practice, we do not have infinite memory, we do not have infinite time, and also any signal we want to analyze does not have infinite duration. In addition, the spectrum of the signal changes with time, just as music is composed of different notes that change with time. Consequently, the DTFT generally is not computable unless we have an analytical expression for the signal we analyze, in which case we can compute it symbolically. But most of the time this is not the case, especially when we want to determine the frequency spectrum of a signal measured from an experiment. In

this situation, we do not have an analytical expression for the signal, and we need to develop an algorithm that can be computed numerically in a finite number of steps, such as the discrete Fourier transform (DFT).

4.3 Discrete Fourier Transform (DFT)

Definition

The discrete Fourier transform (DFT) and its own inverse discrete Fourier transform (IDFT), associate a vector $X = (X(0), X(1), \cdots, X(N-1))$ to a vector $x = (x(0), x(1), \cdots, x(N-1))$ of N data points, as follows,

$$\begin{cases} X(k) = DFT\{x(n)\} &= \sum_{n=0}^{N-1} x(n) \exp(-j2\pi kn/N),\ k = 0, 1, \cdots, N-1 \\ x(n) = IDFT\{X(k)\} &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \exp(j\pi kn/N),\ n = 0, \cdots, N-1 \end{cases} \tag{4.3}$$

If we define $w_N = \exp(-j2\pi k/N)$, then

$$X(k) = DFT\{x(n)\} = \sum_{n=0}^{N-1} x(n)(w_N)^n$$

which implies that $X(k)$ is a weighted summation of $(W_N)^n$ (basis).

Example 8 *(see Appendix 6.8) Let $x = [1, 2, -1, -1]$ be a data vector of length $N = 4$. Then applying the definition, $w_4 = \exp(-j2\pi/4) = -j$, and therefore*

$$X(k) = 1 + 2(-j)^k - 1(-j)^{2k} - 1(-j)^{3k}$$

for $k = 0, 1, 2, 3$. This yields the DFT vector

$$X = DFT\{x\} = [1, 2 - 3j, -1, 2 + 3j]$$

4.3.1 The relationship between DFT and DTFT

In spite of their similarities, the DFT and DTFT are two quite different operations. Whereas the DFT is a numerical operation, which computers a finite number of coefficients $X(0), \cdots, X(N-1)$ from a finite set of data $x(0), \cdots, x(N-1)$, the DTFT is not computable numerically because it yields a continuous function, $X(\omega)$, based on an infinite sequence $x(n)$.

In this section we address the problem of estimating $X(\omega) = DTFT\{x(n)\}$ based on the DFT of a sample of finite length N. The finite sample can be given in terms of either an analytic expression or a set of observations stored in memory, such as a vector.

Before going any further, let us see how we can reconcile the fact that we defined the DFT for data in the form $x(0), \cdots, x(N-1)$, while in reality we might have data starting at any point in time, such as $x(n_0), \cdots, x(n_0+N-1)$. To simplify the arguments to follow, we assume the initial time to be $n_0 = -N/2$, with any choice of rounding for odd values of the data length N. Then we can write the DFT in terms of the finite sum

$$\hat{X}_N(\omega) = \sum_{n=-N/2}^{N/2-1} x(n) \exp(-j\omega n)$$

and $X(k) = \hat{X}_N(k2\pi/N)$. Also recall the DTFT as an infinite sum

$$X(\omega) = \sum_{-\infty}^{\infty} x(n) \exp(-j\omega n)$$

Now the question is, will $\hat{X}_N \to X$ as $N \to \infty$ for each value of the frequency variable ω?. If the answer is positive, the approximation makes sense and we can say that we can estimate the frequency spectrum of the whole signal based on a set of samples. The more data(i.e. the larger N is), the better the approximation.

What decides the convergence is how the signal $x(n)$ behaves as the index $n \to \infty$. In order to fully understand the arguments presented, recall that the infinite series $\sum 1/n^L$, with L an integer, is divergent for $L = 1$ and convergent for $L \geq 2$. In particular, we distinguish three different cases, all illustrated in Fig. 30.

- Absolutely summable signals. These signals are such that

  $$\sum_{n=-\infty}^{\infty} |x(n)| < \infty$$

  For these signals the DTFT always exists, is finite, and converges for every ω, as

  $$\lim_{N\to\infty} \hat{X}_N(\omega) = X(\omega) < \infty$$

  The fact that $X(\omega)$ is finite comes from the following inequality

  $$|X(\omega)| = |\sum x(n) \exp(-j\omega n)| \leq \sum |x(n)| < \infty$$

  These are the best-behaved sequences, in terms of convergence. This means that for every ω, we can approximated $X(\omega)$ infinitely closely by the finite sequence $\hat{X}_N(\omega)$, with N sufficiently large.

  Typical examples in this class are signals decaying as $1/n^2$ for which $|x(n)| \leq C/n^2$, for n sufficiently large. In particular, signals decaying exponentially fast.

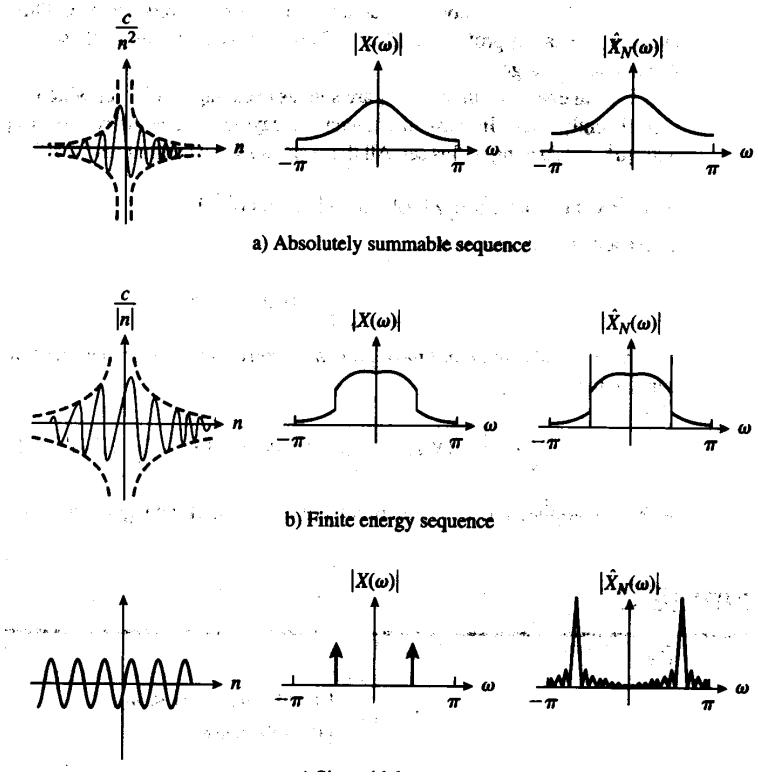

Figure 30: Three cases.

- Signals with Finite Energy (not infinitely summable). There are signals such that

$$\sum |x(n)|^2 < \infty$$

  For these signals we do not have pointwise convergence as in the preceding case, but we can say that

$$\lim_{N\to\infty} \int |X(\omega) - \hat{X}_N(\omega)|^2 d\omega \to 0$$

- This is one of the more important issues in DSP and we will treat the third case in the next subsection.

Example 9 *Consider the sequence*

$$x(n) = \frac{\sin(0.25\pi n)}{\pi n}$$

We want to estimate $X(\omega) = DTFT(x(n))$ using the DFT. Choose for example, finite sequences $x(n)$ for $-L \leq n \leq L$, of length $N = 2L + 1$. The plots of $\hat{X}_N(k)$ for several values of L are shown in Fig. 31, and the lack of convergence at the discontinuities ($\omega = \pm\pi/4$) is evident.

Example 10 *Consider the sequence*

$$x(n) = \frac{\sin^2(0.25\pi n)}{(\pi n)^2}$$

Again we want to estimate $X(\omega) = DTFT(x(n)$ using the DFT. Fig. 32 shows convergence for all values of ω by comparing two cases with $L = 32$ and $L = 1024$. This is expected because the signal is absolutely summable.

4.3.2 DFT for spectral estimation

One of the uses of the DFT is to estimate the frequency spectrum of a signal. In particular, given a continuous time signal $x(t)$, the goal is to determine its frequency components by taking a finite set of samples as a vector, $x(0), x(1), \cdots, x(N-1)$, and computing its DFT as in the previous section. This is shown in Fig 33.

The fact that we window the signal $x(n)$ (i.e. we take a finite set of data) will have an effect on the frequency spectrum we compute, and we have to be aware of that when we interpret the results of the DFT. This can be seen by defining a window sequence

$$w(n) = \begin{cases} 1 & \text{if } n = 0, 1, \cdots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

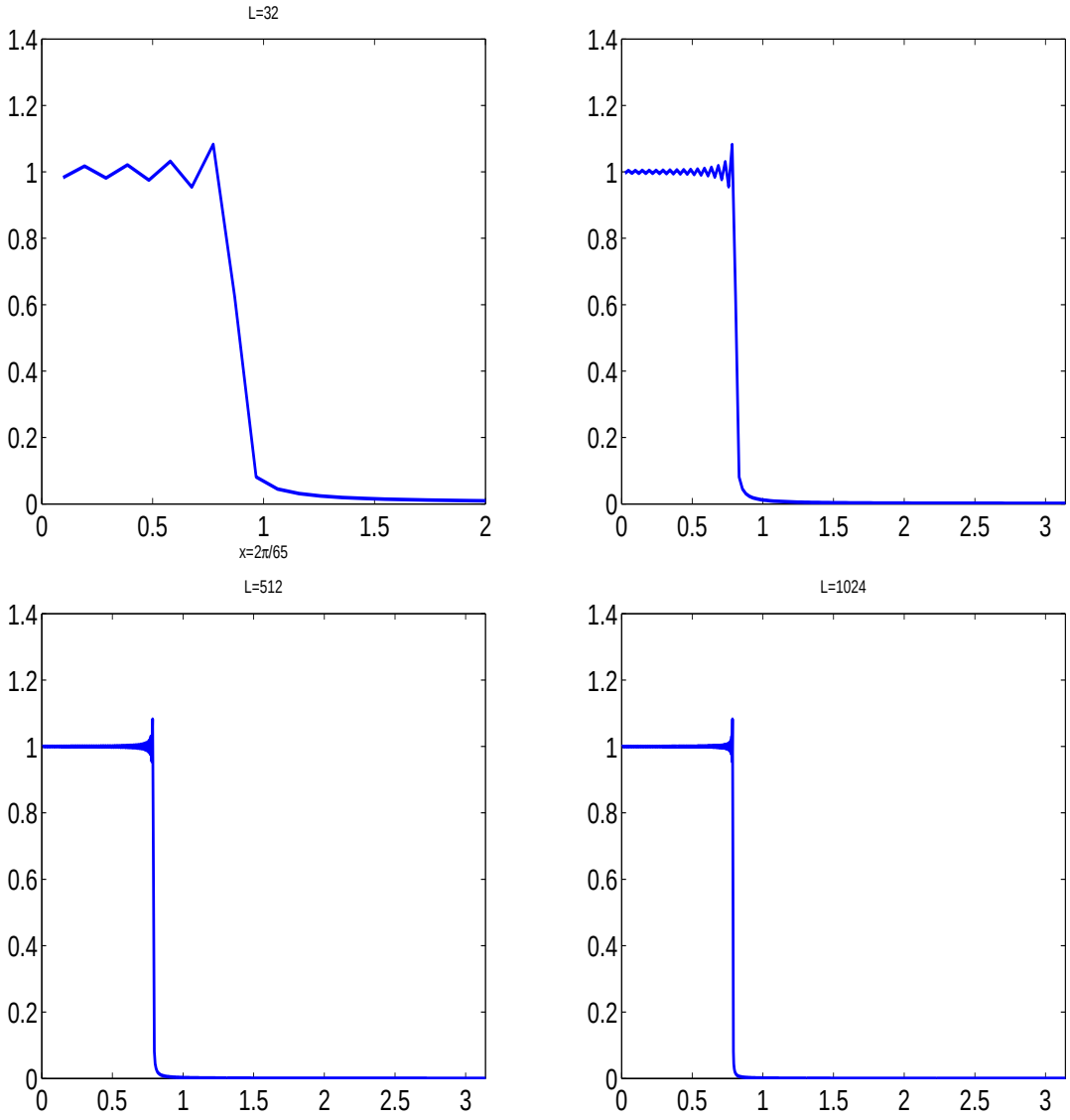

Figure 31: Discontinuity for $L = 32, 128, 512, 1024$

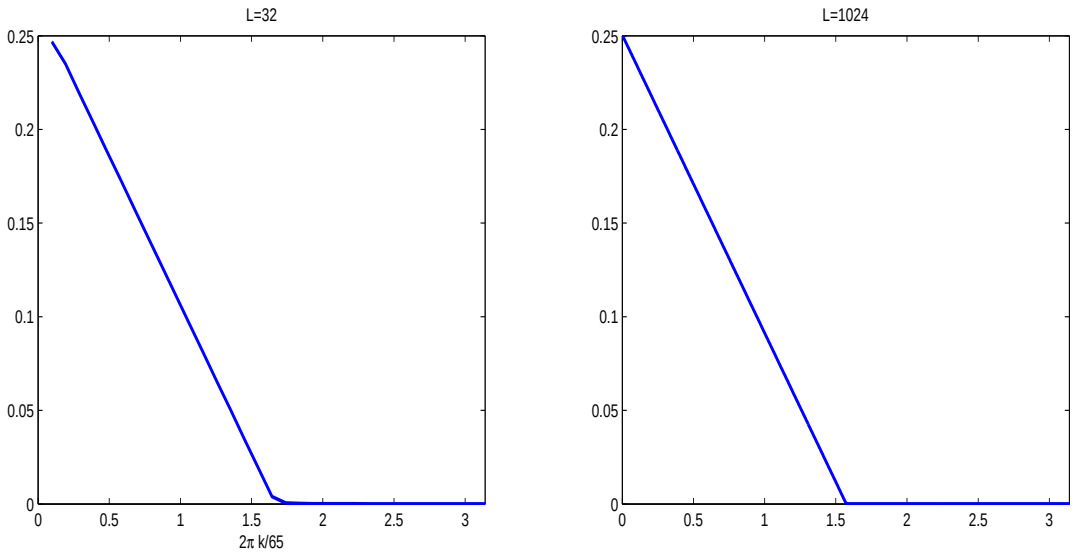

Figure 32: A summable case.

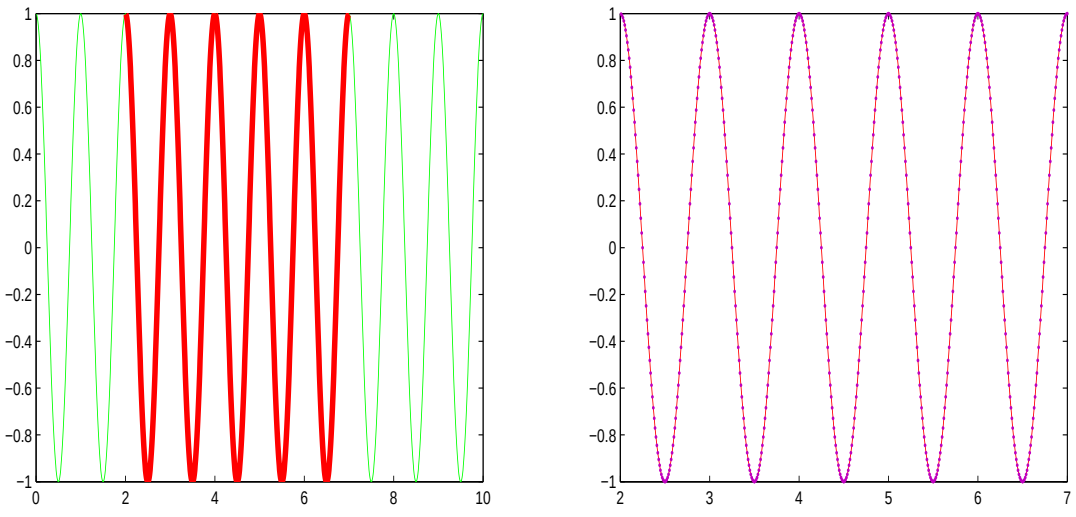

Figure 33: Windowing and then sampling.

and the DTFT of the windowed signal

$$X_w(\omega) = DTFT(w(n)x(n)) = \sum_{n=0}^{N-1} x(n)\exp(-j\omega n)$$

Now notice that we can relate the right-hand side of the preceding expression to the DFT of the finite sequence $x(0), \cdots, x(N-1)$ as

$$X(k) = X_w(\omega)|_{\omega=k2\pi/N}$$

In other words, by the DFT we compute samples of the frequency spectrum of the windowed signal $x(n)w(n)$. To see the effect of the windowing operation on the frequency spectrum, let us examine the case when the signal $x(n)$ is a complex exponential, such as $x(n) = \exp(j\omega_0 n)$. Then

$$X_w(\omega) = \sum_{-\infty}^{\infty} w(n)\exp(j\omega_0 n)\exp(-j\omega n) = W(\omega - \omega_0)$$

where the rightmost term is the frequency spectrum of the window function

$$W(\omega) = DTFT(w(n)) = \sum_{n=0}^{N-1} \frac{1-\exp(-j\omega N)}{1-\exp(j\omega)}$$

This expression can be easily manipulated to simpler form and we can write

$$W(\omega) = \exp(-j\omega(N-1)/2)\frac{\sin(\omega N/2)}{\sin(\omega/2)}$$

The magnitude of this expression is shown in Fig. 34; it repeats periodically with period 2π. As a result, the DTFT of the windowed complex exponential is given by

$$|X_w(\omega)| = |W(\omega - \omega_0)| = \frac{|\sin[(\omega-\omega_0)N/2]|}{|\sin[(\omega-\omega_0)/2]|}$$

Fig. 34 compares the frequency spectra of the complex exponential before and after the windowing operations. Without windowing, the complex exponential is perfectly localized in frequency, since

$$DTFT(\exp(j\omega_0 n)) = 2\pi\delta(\omega - \omega_0)$$

Once we window the signal, its frequency spectrum is not perfectly localized any more. The energy spreads around the frequency ω_0 within the mainlobe and trickles to other frequencies by the sidelobes of the sinc function.

A number of windows currently used are shown in Fig. 35. In all cases, they are designed with a smoother transition between the two regions (inside and outside the window), and as a consequence the sidelobes are lower. The drawback is that they all have a wider mainlobe. Given

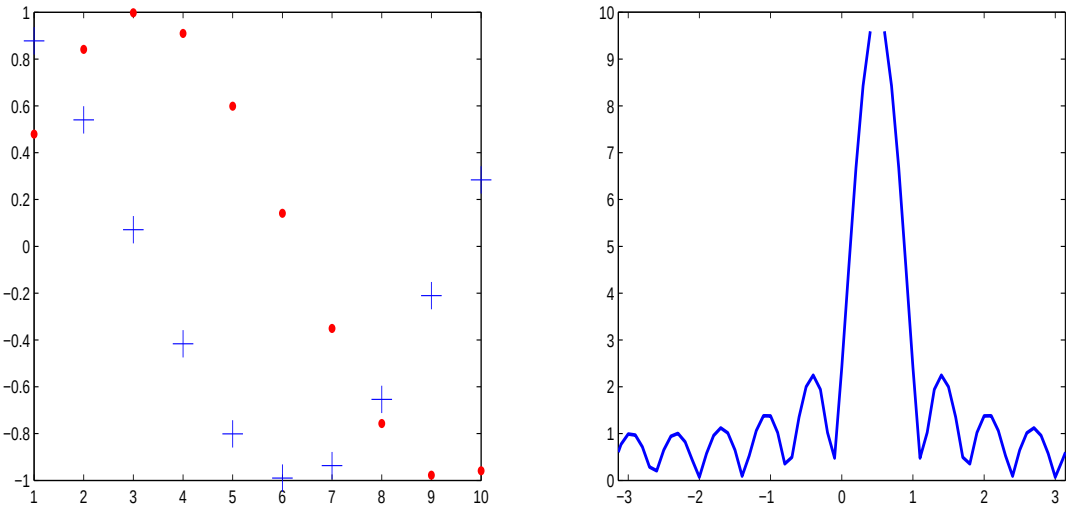

Figure 34: Left, $\sin(j\omega_0 n)$ and $\cos(j * \omega_0 n)$. Right: the spectrum of $\exp(j\omega_0 n)$ with $n = 1, 2, \cdots, 10$ (N=10) and $\omega_0 = 0.5$.

a fixed set of data points, we have to accept a compromise between frequency resolution (width of the mainlobe) and sidelobes.

As we have seen, the DFT is the basis of a number of applications, and is one of the most important tools in digital signal processing. The problem with this algorithm is that a brute-force implementation would hardly be feasible in real time, even for a data set of modest length. Fortunately, the DFT can be computed in a very efficient way, exploiting the very structure of the algorithm we have presented by using the fast Fourier transform (FFT), which is an efficient way of computing he DFT. It is know that for a data set of length N, the complexity of the FFT grows as $N \log_2 N$, the complexity of the DFT computed by brute force grow as N^2. FFT is widely used in many applications.

4.4 *Sampling and reconstruction*

We have assumed so far that the sequence of samples on which we are to operate is given to us. In practice, we shall totally have to obtain this by sampling a continuous time (or space) signal $x(t)$, say. The question we consider here is under what conditions we can completely reconstruct the original signal $x(t)$ from its discretely sampled signal $x(n)$.

To this end, let us introduce more notation. Consider a signal $x(t)$ and its samples $x(n) = x(nT_s)$, with T_s being the sampling interval and $F_s = 1/T_s$ the sampling frequency.

Now the question is how we can relate $X(\omega) = DTFT(x(n))$ to $X(F) = FT(x(t))$. In particular, we want to identify conditions under which we can determine $X(F)$ from $X(\omega)$, or in other words, if we can reconstruct the analog signal $x(t)$ form its samples $x(n)$.

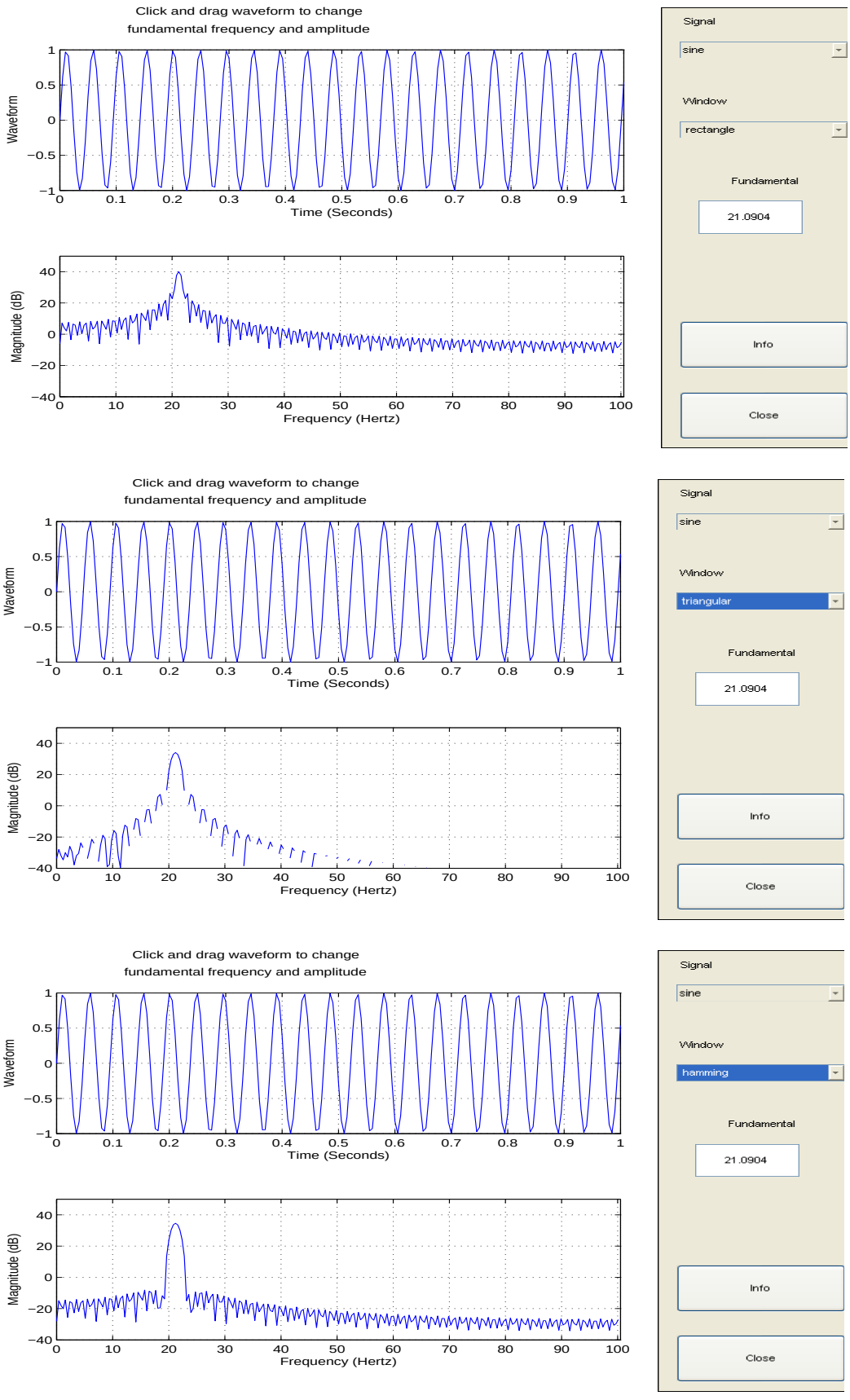

Figure 35: Various window. Created from Matlab demos: Signal Processing: transforms: DFT.

To achieve this goal, define

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) = x(t)\sum \delta(t - nT_s)$$

This signal shows the sampling operation; all the information within the sampling interval is lost but at the same time the signal is still defined in continuous time. Therefor we can take the Fourier transform, using both expressions on the right hand side, to obtain

$$X_s(F) = FT(x_s(t)) = \sum_{n=-\infty}^{\infty} x(n)\exp(-j2\pi(F/F)s)/n) = X(F) * F_s\sum_{-\infty}^{\infty}\delta(t - nT_s)$$

from the fact tat

$$FT(\sum \delta(t - nT_s)) = F_s\sum \delta(F - kF_s)$$

. The convolution with a delta function yields just a shift of the variable, so we can write

$$X_s(F) = X(\omega)|_{\omega=2\pi(F/F_s)} = F_s\sum X(F - kF_s)$$

where $X(\omega) = DTFT(x(n))$. In other words, when a signal $x(t)$ is sampled in time, its frequency spectrum $X(F)$ is repeated in frequency.

We can see that if the signal $x(t)$ is bandlimited, in the sense that $X(F) = 0$ for $|F| > F_B$, for some frequency F_B called the bandwidth of the signal, and we sample at a frequency $F_s > 2F_B$, then there is no overlapping between the repetitions on the frequency spectrum. In other words, if $F_s > 2F_B$,

$$X_s(F) = F_sX(F)$$

in the interval $-F_s/2 < F < F_s/2$ and $X(F)$ can be fully recovered from $X_s(F)$. This is very important because it states that the signal $x(t)$ can be fully recovered from its samples $x(n) = x(nT_s)$, provided we sample fast enough, $F_s > 2F_B$.

The way to recover the continuous time signal $x(t)$ from its samples $x(nT_s)$ is by an interpolating function $g(t)$. In fact, define $\hat{x}(t)$

$$\hat{x}(t) = \sum x(nT_s)g(t - nT_s)$$

Its Fourier transform is determined using simple properties

$$\hat{X}(F) = \sum x(n)\exp(-j2\pi FT_s)G(F)$$

where $G(F) = FG(g(t)$, the interpolating function. As we have seen, the summation is just $X_s(F)$. Therefore, if $G(F)$ is such that

$$G(F) = T_s\text{rect}(\frac{F}{F_s} = \begin{cases} T, & \text{if } -F_s/2 < F < F_s/2 \\ 0 & \text{otherwise} \end{cases}$$

Then $\hat{X}(F) = X(F)$ and $\hat{x}(t) = x(t)$. This implies that the ideal interpolating function is given by

$$g(t) = \text{IFT}\left(T_s\text{rect}\left(\frac{F}{F_s}\right)\right) = \text{sinc}(t/T_s)$$

and we can perfectly reconstruct the continuous time signal $x(t)$ from its samples $x(n)$:

$$X(t) = \sum x(n)\text{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

In summary, we have the following Nyquist–Shannon sampling theorem.

Theorem 2 *Exact reconstruction of a continuous-time baseband signal from its samples is possible if the signal is bandlimited and the sampling frequency is greater than twice the signal bandwidth.*

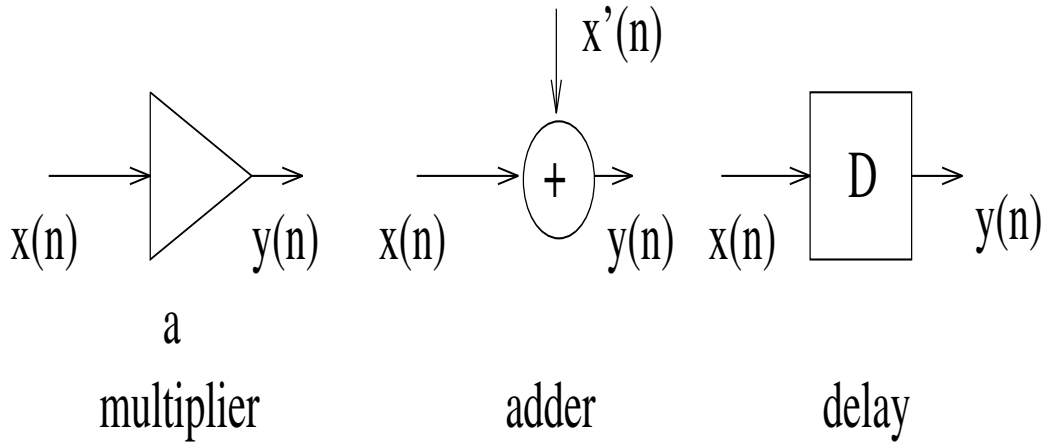

Figure 36: Multiplier, adder and delay units

5 Digital Filters

A digital filter is any electronic filter that works by performing digital mathematical operations on an intermediate form of a signal. Digital filters can achieve virtually any filtering effect that can be expressed as a mathematical function or algorithm. The two primary limitations of digital filters are their speed (the filter can't operate any faster than the computer at the heart of the filter), and their cost. However as the cost of integrated circuits has continued to drop over time, digital filters have become increasingly commonplace and are now an essential element of many everyday objects such as radios, cellphones, and stereo receivers.

A well-known signal processing wizard is said to have remarked, "When you think about it, everything is a filter."

5.1 Operations on Sequences

Digital filtering is all based on a small set of very simple linear shift invariant operation on sequence. These are

1. Multiplication by a constant coefficient. If $\{x(n)\}$ is the input sequence, the output is

$$\{y(n)\} = a\{x(n)\} = \{\cdots, ax(-1), ax(0), a(x), \cdots, ax(n)\cdots\}$$

   . Multipliers are normally depicted graphically by a triangle as in Fig. 36.

2. Adders do the obvious thing: if $\{x_1(n)\}, \{x_2(n)\}$ are the input sequences, the output is just

$$\{y(n)\} = \{\cdots, x_1(-1) + x_2(-1), x_1(0) + x_2(0), x_1(1) + x_2(1), \cdots\}$$

3. The other component is a delay unit, i.e. a single memory element. This operates on the input sequence $\{x(n)\}$ to give the output

$$\{y(n)\} = D\{x(n)\} = \{x(n-1)\}$$

Note that you must regard this as chning the sequence, rather than acting independently on the elements.

5.2 Filters

The commonly used digital filters are all based on combinations of these three operations. The general filter can be described by the operational equation

$$\{y(n)\} = \sum_{m=0}^{N} a(m)D^m\{x(n)\} + \sum_{m=1}^{N} b(m)D^m\{y(n)\}$$

Note that the sum over the outputs starts at a delay of 1, not 0. If we use the ZT, with $\{x(n)\} \leftrightarrow X(z)$,$\{y(n)\} \leftrightarrow Y(z)$, we must first note that the operator D has a ZT representation z^{-1}, from the shift result we established before. This mean that the filter above can be rewritten in ZT form

$$Y(z)(1 - \sum_{m=1}^{N} b(m)z^{-m}) = \sum_{m=0}^{N} a(m)z^{-m}X(z)$$

No the ratio

$$H(z) = Y(z)/X(z)$$

is known as the ZT transfer function of the filter

$$H(z) = \frac{\sum_{m=0}^{N} a(m)z^{-m}}{1 - \sum_{m=1}^{N} b(m)z^{-m}}$$

Now the transfer function $H(z)$ is the ZT of the filter output sequence when the input sequence has ZT $X(z) = 1$. This is the ZT of the unit impulse sequence

$$\{x(n)\} = \{1, 0, \cdots, \cdots\}$$

Thus the other way to characterise the filter is in terms of its impulse response since the response to any sequence is just the sum of the responses to the individual impulses of the sequence components.

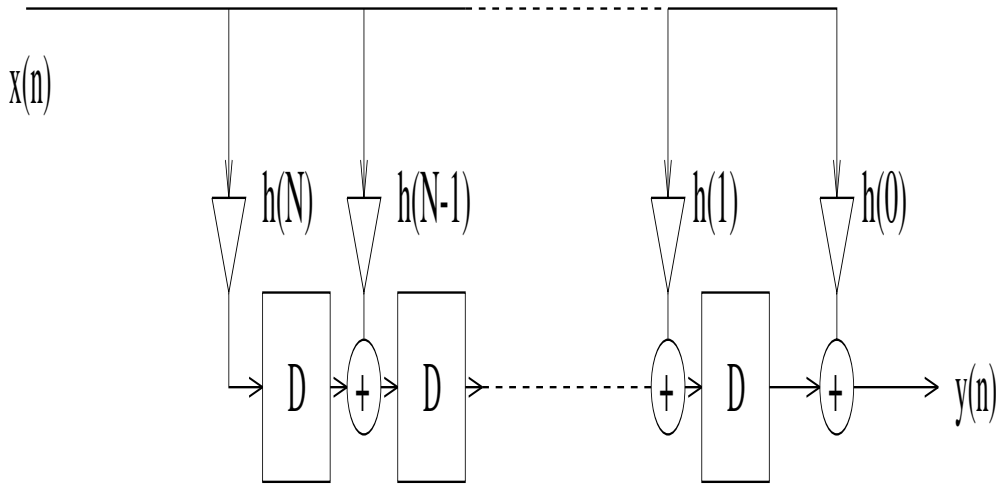

Figure 37: Direct nonrecursive filter implementation

5.3 Nonrecursive Filters

5.3.1 Operational Definition

When a filter is nonrecursive, its operational equation can be written

$$\{y(n)\} = \sum_{m=0}^{N} h(m) D^m \{x(n)\}$$

Such filters are also called finite impulse response filters, for the obvious reason that their IR contain only finitely many nonzero terms. Correspondingly, the ZT of a nonrecursive filter can be written as

$$H(z) = \sum_{m=0}^{N} h(m) z^{-m}$$

and the impulse response is just

$$\{h(n)\} = \{h(0), h(1), \cdots\}$$

Such a filter obviously has finite memory, in the sense that its response to a particular input element is zero after N time steps. This can also be seen from the block diagram of a direct implementation of the filter, Fig 37, which is a canonical form. FIR filters are all called moving average (MA) filters.

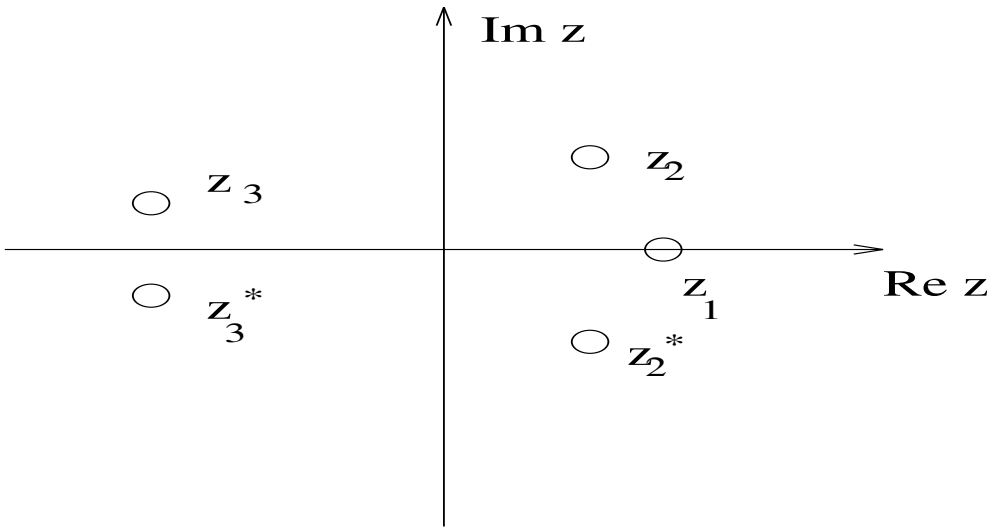

Figure 38: The zeros of an FIR filter transfer function lies on the real axis of occur in conjugate pairs

5.3.2 Zeros

Trying to figure out how an FIR filter will behave, given its IR, is note always so simple. Another way of looking at it is through its frequency domain behaviors. We can make a start on this by examining the zeros of its transfer function $H(z)$, i.e. those values of z for which

$$H(z) = 0$$

Since $H(z)$ is a polynomial of order N with real coefficients, it follows that the equation has N roots, which are either real or occur in complex conjugate pairs. We can express $H(z)$ in terms of the roots by writing

$$z^N H(z) = \prod_{m=1}^{N} (z - z_m)$$

where in general $z_m = |z_m| \exp(j \arg[z_m]]$ is the mth root, or zero of the transfer function. The zeros of a transfer function are usually denoted graphically in the complex z-plane by circles, as shown in Fig. 38.

The factorisation of $H(z)$ implies that $H(z)$ can be written as a product of quadratic polynomials in z^{-1} with real coefficients

$$H(z) = \prod_{m=1}^{M} (h_m(0) + h_m(1)z^{-1} + h_m(2)z^{-2})$$

This is effectively a series implementation of the transfer function. Series implementations have advantage in terms of both hardware and software modularity and sensitively to coefficient quantization. They are often sued in practice.

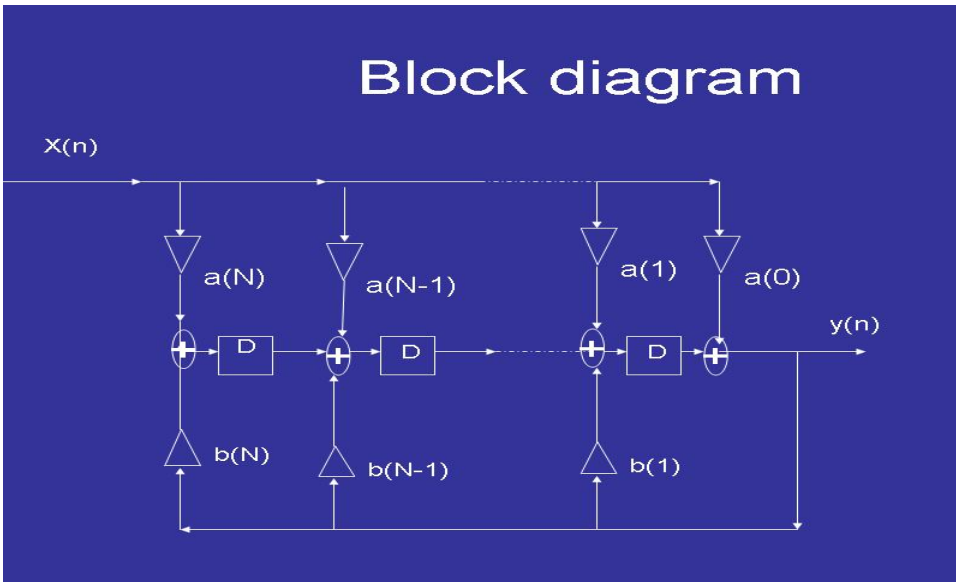

Figure 39: Block diagram of an ARMA filter.

5.4 Recursive Filters

5.4.1 Operational Definition

Of the many filter transfer function which are not FIR, the most commonly use in DSP are the recursive filters, so called because their current output depends not only on the last N inputs but also on the last N outputs. They have operational equation which are written as

$$(1 - \sum_{m=1}^{N} b(m)D^m)\{y(n)\} = \sum_{m=1}^{N} a(m)D^m)\{x(n)\}$$

and a ZT transfer function

$$H(z) = \frac{A(z)}{B(z)}$$

where

$$A(z) = \sum_{m=0}^{N} a(m)z^{-m}, B(z) = 1 - \sum_{m=1}^{N} b(m)z^{-m}$$

and if $B(z) \neq 1$, then their impulse response in infinity-they are IIR.

$$\{h(n)\} = \{h(0), h(1), \cdots\}$$

The block diagram (Fig. 39) of such a filter is an obvious generalisation of that for the nonrecursive Nth order filter. If the numerator coefficient $a(m) = 0$, then the filter is called an all pole filter(or sometimes an autoregressive (AR) filter).

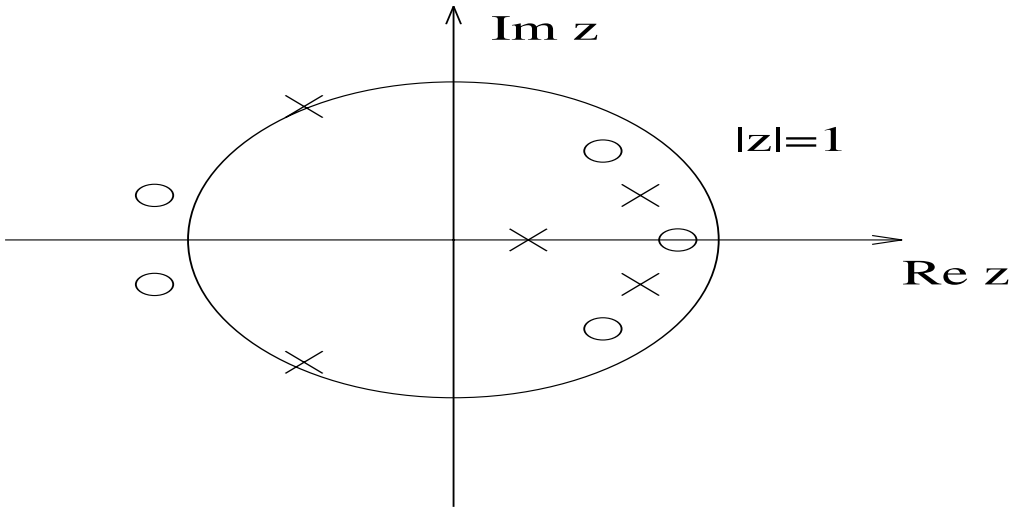

Figure 40: Poles and zeros of a filter transfer function.

5.4.2 Poles and Zeros

Writing he numerator and denominator of $H(z)$ in factors we get

$$H(z) = \frac{A(z)}{B(z)} = (\prod_{n=1}^{N}(z - \alpha_n))/(\prod_{n=1}^{N}(z - \beta_n))$$

We know that the roots α_n are the zeros of the transfer function. The roots of the equation $B(z) = 0$ are called the poles of the transfer function. They have greater significance for the behaviour of $H(z)$: it is singular at the points $z = \beta_n$. Poles are drawn on the z-plane as crosses, as shown in Fig. 40.

ON this figure, the unit circle has been shown on the z-plane. In order for a recursive filter to be bounded input bounded output stable, all of the poles of its transfer function must lie inside the unit circle. A filter is BIBO stable if any bounded input sequence gives rise to a bounded output sequence. Now if the pole of the transfer lie insider the unit circle, then they represent geometric series with a coefficient whose magnitude $|\beta_m| < 1$, i.e. a sequence $\{1, \beta_m, \beta_m^2, \cdots\}$ which is convergent. Consequently, the filter output, which is a sum of such sequences weighted by the appropriate input terms, is bounded if the input is. If, on the other hand, $\beta_m| > 1$, the geometric series diverges and the filter output will grow without bound as $n \to \infty$. If $\beta_m| = 1$, the filter is said to be conditionally stable: some input sequence will lead to bounded output sequence and some will not.

Since FR filters have no poles, they are always BIBO stable: zeros have no effect on stability.

5.5 Frequency and digital filters

5.5.1 Poles, Zeros and Frequency Response

Now suppose we have the ZT transfer function of a filter

$$H(z) = \frac{A(z)}{B(z)}$$

We can formally represent the frequency response of the filter by substituting $z = \exp(j\omega)$ and obtain

$$H(\omega) = \frac{A(\omega)}{B(\omega)}$$

Obviously, $H(\omega)$ depends on the locations of the poles and zeros of the transfer function, a fact which ca n be made more explicit by factoring the numerator and denominator polynomials to write

$$H(\omega) = K\frac{\prod_{n=1}^{N}(\exp(j\omega) - \alpha_n)}{\prod_{n=1}^{N}(\exp(j\omega) - \beta_n)}$$

where K is a constant, α_n, β_n are respectively the nth zero and pole of the transfer function. Each factor in the numerator or denominator is a complex function of frequency, which has a graphical interpretation in the rms of the location of the corresponding root in relation to the unit circle This is illustrated in Fig. 41. The factors are simply the difference vectors between the root location and the point on the unit circle at angle ω. This is true for any stable filter (i.e poles inside unit circle).

Writing, for example, the nth zero as

$$\alpha_n = |\alpha_n| \exp(j\theta_n)$$

we find that it makes a contribution with magnitude

$$|A_n(\omega)|^2 = 1 + |\alpha_n|^2 - 2|\alpha_n| \cdot (\omega - \theta_n)$$

to the overall frequency response. This has a minimum at $\omega = \theta_n$ and a maximum at $\omega = \theta_n + \pi$. In other words, the contribution of a zero to the filter frequency response is a minimum at the angle of the zero and a maximum at the diametrically opposite frequency. Similarly, the contribution from a pole is a maximum at eh angle of the pole and a minimum at the opposite angle. Thus we can make a reasonable guess about the filter frequency response imply by looking at the pole-zero diagram: lowpass filters have poles near the original $\omega = 0$ and zeros at high frequencies and so on.

To get a more detailed picture of the frequency response, it is always useful to calculate it at frequencies $\omega = 0, \pi/2, \pi$, which are simple to calculate, but even with these few points and the locations of poles and zeros, we can get a good idea of how the response will look. Note also that, because they are in the denominator, the contribution of a pole to the response grows as $|\beta_n| \to 1$, while a zero on the unit circle implies a null in the response at that frequency.

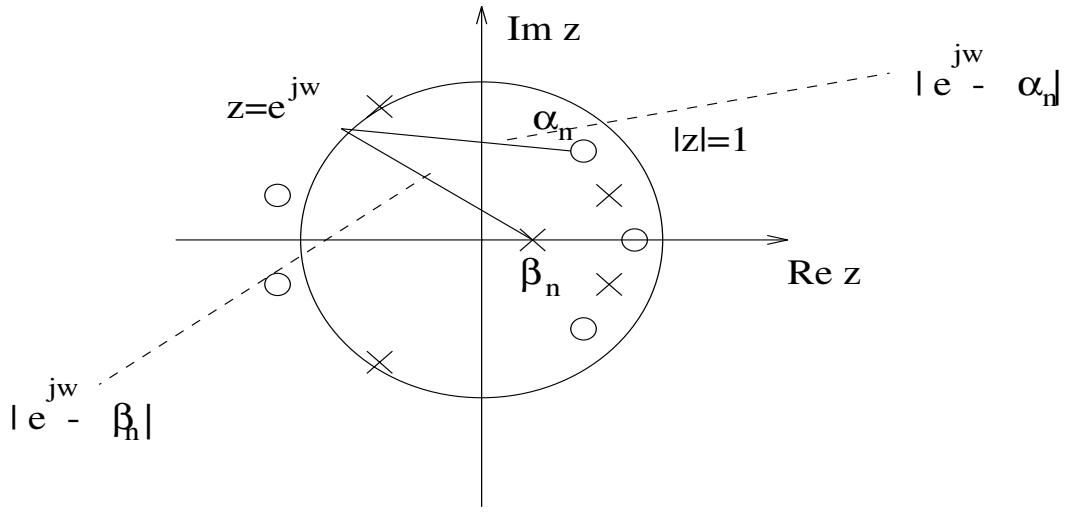

Figure 41: The contribution of a pole or zero to a filter frequency response

To make matters more concrete, Fig. 42 shows what effect varying the magnitude and angle of a simple zero has on the frequency response magnitude, while Fig. 43 does the same thing for a pole. A significant point to note in comparing to the diagrams is how even a single pole can give a sharp peak in the frequency response magnitude, where a zero inevitable gives a rather broad one. This is a consequence of the poles belonging to the denominator of the transfer function, of course. It is primarily for this reason that recursive designs are preferred in many applications.

5.5.2 Filter Types

There are four main classes of filter in widespread use: lowpass, highpass, bandpass and bandstop filters. The name are self-explanatory, but he extent to which the ideal frequency responses can be achieved in practice is limited. This four types are shown in Fig. 44.

In practice, these ideal shapes require IIR filters which do not have a good approximation in terms of a rational transfer function. To achieve a given magnitude response, a recursive filter of a given order is generally better than a nonrecursive one, but this is often at the expense of an unpleasant (nonlinear) phase response: linear phase implies the filter simply delays the input signals, as well as shaping its frequency content. The resulting filter will be inadequate in several respects compared with the ideal:

1. Finite stopband attenuation. A practical filter will not have a null response to frequency outside its passband: this would require zeros at all such frequencies.

2. Finite transition bandwidth. Any realisation will not have a discontinuous frequency response: there will be a band of frequencies width are neither stopped not passed, but are transitional.

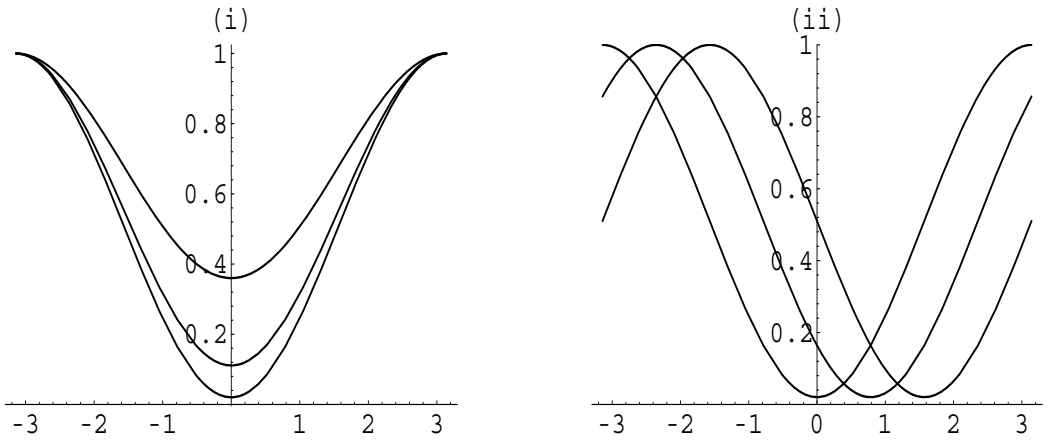

Figure 42: The effect of varying (1) the magnitude and (ii) the angle of a simple zero. The squared magnitude is plotted vs. radial frequency.

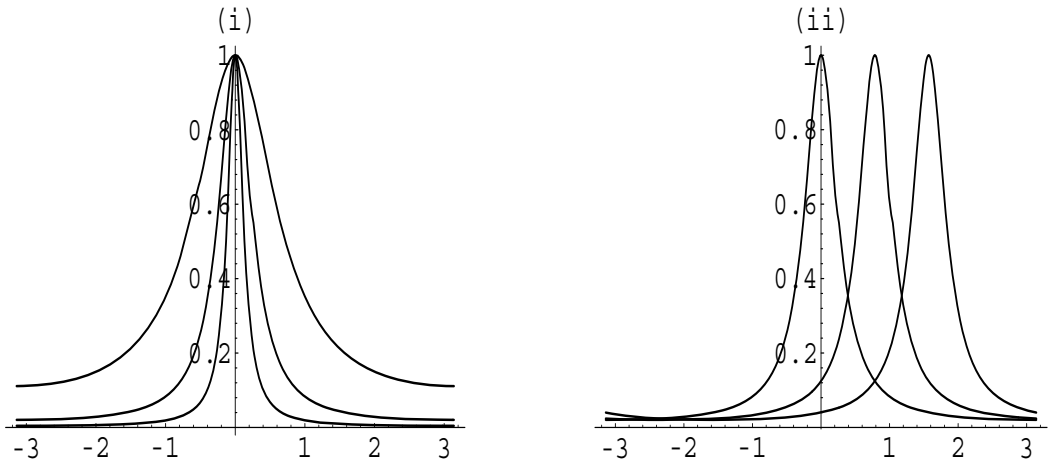

Figure 43: The effect of varying (1) the magnitude and (ii) the angle of a simple pole. The squared magnitude is plotted vs. radial frequency.

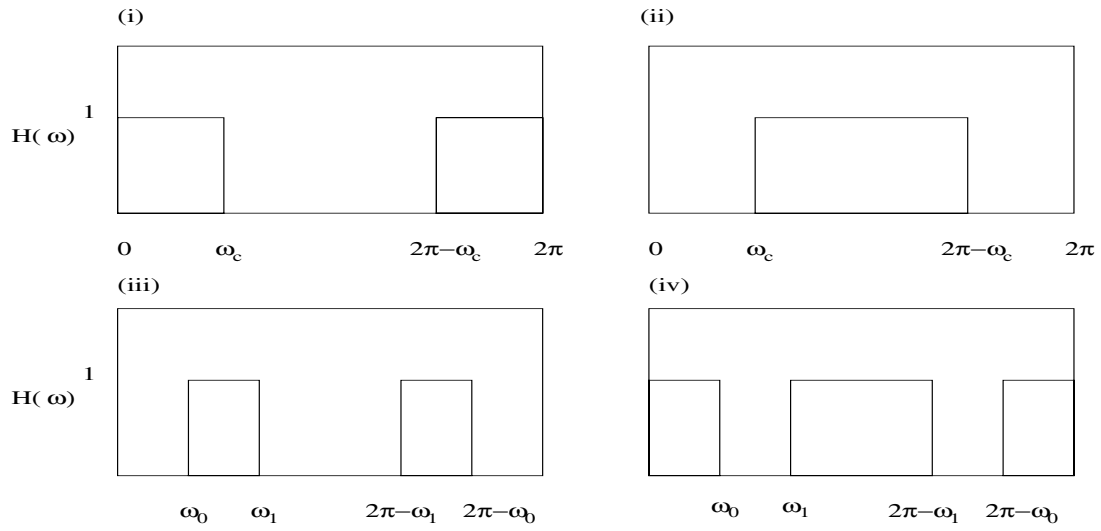

Figure 44: The common types of digital filter frequency response: (i) lowpass, (ii) highpass, (iii) bandpass, (iv) bandstop

3. Many filters also have a nonuniform response in the passband: passband ripple

Different designs, such Butterworth, Chebyshev or Elliptic, have different trade-off between these artifacts. Originally, much digital filter design was based on the transformation of analogue designs, using a variety of techniques including he impulse-invariant of bilinear transforms.

5.6 Simple Filter Design

In a number of cases, we can design a linear filter almost by inspection, by moving poles and zeros like pieces on a chessboard. This is not just a simple exercise designed for an introductory course, for in many cases the use of more sophisticated techniques might not yield a significant improvement in performance.

In this example consider the audio signal $s(n)$, digitized with a sampling frequency $F_s kHZ$. The signal is affected by a narrowband (i.e., very close to sinusoidal) disturbance $w(n)$. Fig. 45 show the frequency spectrum of the overall signal plus noise, $x(n) = s(n) + w(n)$, which can be easily determined.

Notice two facts:

1. The signal has a frequency spectrum concentrated within the interval 0 to 20+ kHz.

2. The disturbance is at frequency F_0 kHz.

Now the goal is to design and implement a simple filter that rejects the disturbance without affecting the signal excessively. We will follow these steps:

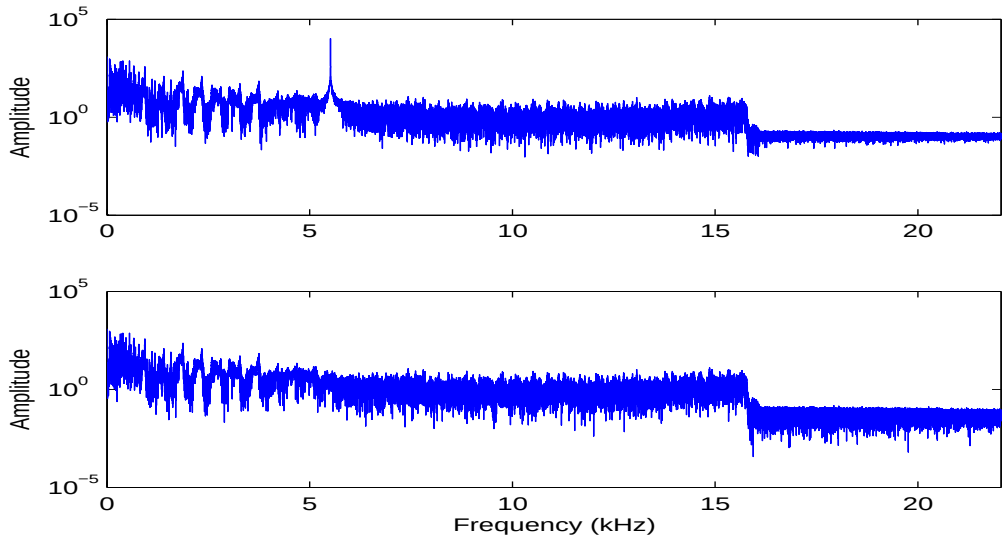

Figure 45: A signal with a disturbance. 'Say it' by John Coltrane Quartet.

Step 1: Frequency domain specifications. We need to reject the signal at the frequency of the disturbance. Ideally, we would like to have the following frequency response:

$$H(\omega) = \begin{cases} 0 & \text{if } \omega = \omega_0 \\ 1 & \text{Otherwise} \end{cases}$$

where $\omega_0 = 2\pi(F_0/F_s)$ (let us say, it is $\pi/4$) radians, the digital frequency of the disturbance. Recall that the digital frequency is a relative frequency, the therefore has no dimensions.
Step 2: Determine poles and zeros We need to place two zeros on the unit circle $z_1 = \exp(j\omega_0) = \exp(j\pi/4)$ and $z_2 = \exp(-j\omega_0) = \exp(-j\pi/4)$. This would yield the transfer function

$$H(z) = K(z^{-1} - z_1)(z^{-1} - z_2) = K(1 - 1.414z^{-1} + z^{-2})$$

If we choose, say $K = 1$, the frequency response is shown in Fig. 46. As we can see, it rejects the desired frequencies, as expected, but greatly distorts the signal.

A better choice would be to select the poles close to the zeros, within the unit circle, for stability. For example, let the poles be $p_1 = \rho\exp(j\omega_0)$ and $p_2 = \rho\exp(-j\omega_0)$. With $\rho = 0.95$, for example, we obtain the transfer function

$$H(z) = K\frac{(z - z_1)(z - z_2)}{(z - p_1)(z - p_2)} = 0.9543\frac{z^2 - 1.414z + 1}{z^2 - 1.343z + 0.9025}$$

and we choose the overall gain K so that $H(z)|_{z=1} = 1$. This yields the frequency response shown in Fig. 47, which is more selective than the previous choice. The only caveat is that the poles have to be within the unit circle, even in the presence of numerical uncertainties.

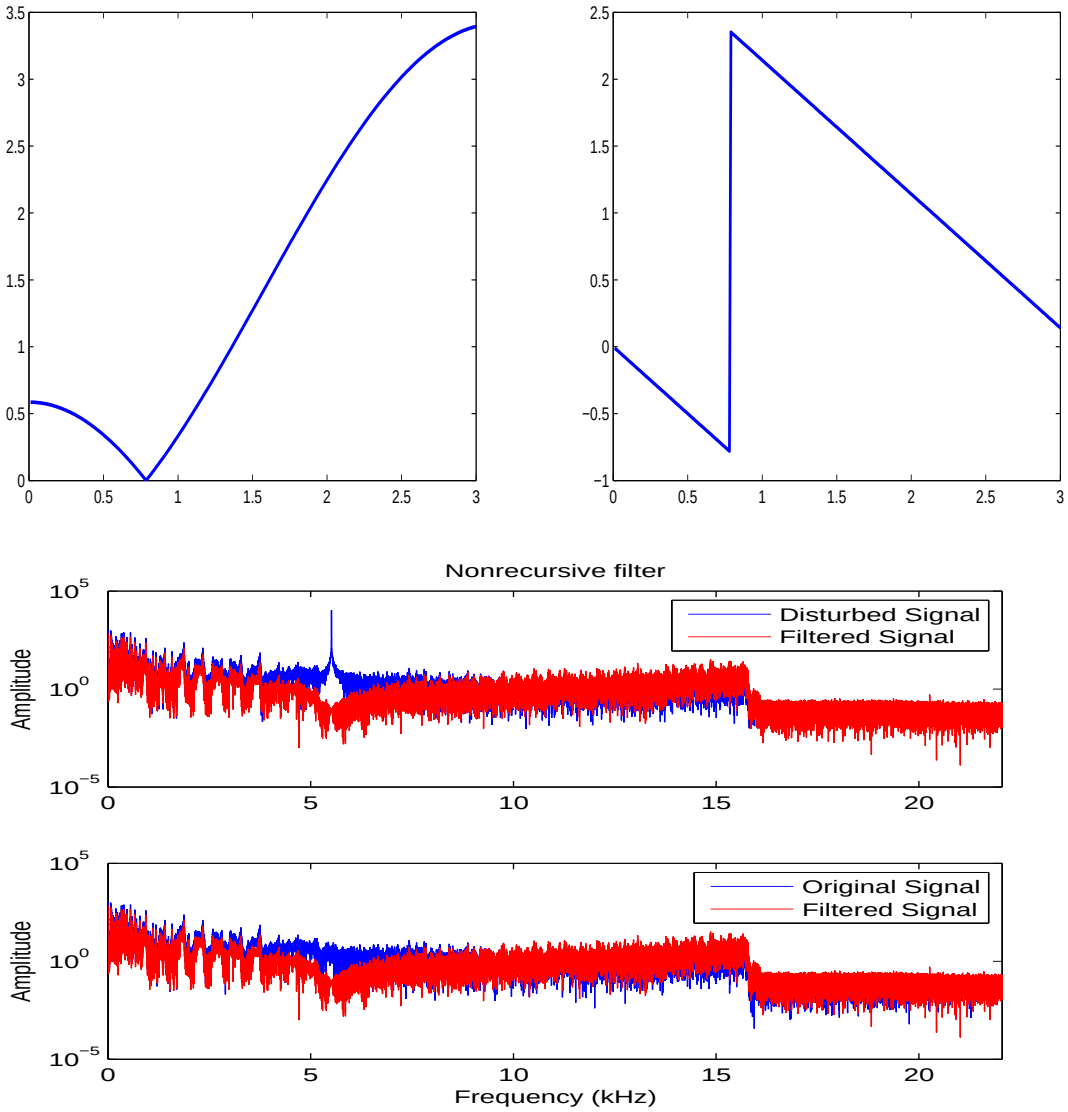

Figure 46: Frequency response with zeros and actual output of filtered signal.

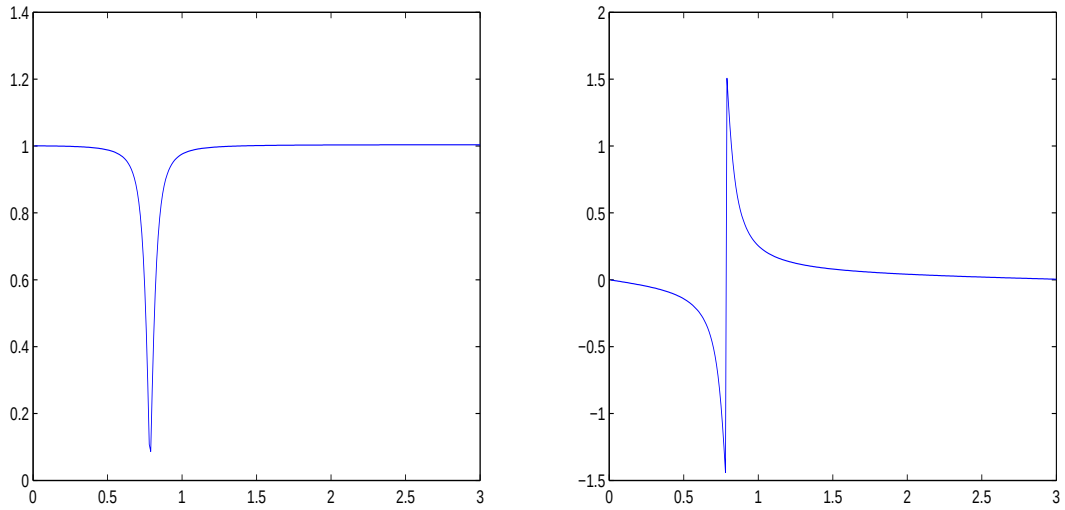

Figure 47: Frequency response with poles close to the unit circle

Step 3: Determine the difference equation in the time domain From the transfer function, the difference equation is determined by inspection:

$$y(n) = 1.343y(n-1) - 0.9025y(n-2) + 0.954x(n) - 1.3495x(n-1) + 0.9543x(n-2)$$

The difference equation can be easily implemented as a recursion in a high-level language. The final result is the signal $y(n)$ with the frequency spectrum shown in Fig. 48, where we notice the absence of the disturbance.

A Matlab program is available in Appendix.

5.7 Matched Filter

One common problem in signal processing is that of detecting the presence of a known signal against a background of noise. This has applications in radar, sonar, communications and pattern recognition. The problem is to choose the filter IR $\{h(n)\}$ which maximises the filter output $\{y(n)\}$, then we want to maximise the output at time 0

$$y(0) = \sum_{n=-\infty}^{\infty} h(n)x(-n)$$

As it stands, however, this is not a well defined problem: if we double all the filter coefficients, the output will also double. We must therefore constrain the filter coefficients to have finite energy, i.e.

$$\sum h^2(n) = 1$$

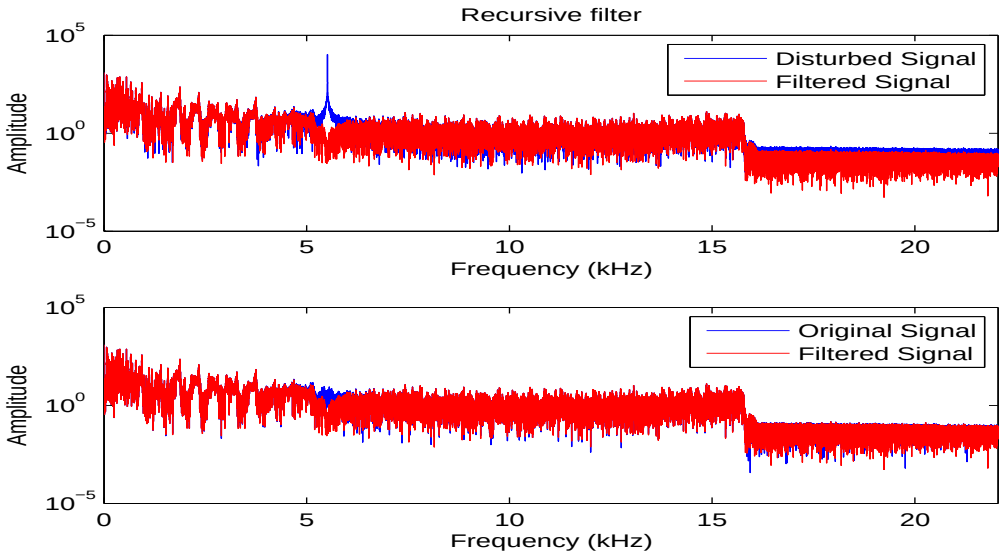

Figure 48: Frequency spectrum of filtered signal

We can solve this constrained maximisation using the technique of Langrange multipliers, which tells us to maximise the function

$$E(h) = y(0) - \lambda(\sum h^2(n) - 1)$$

where the second term is the constraint and λ is called the Lagrange multiplier, which is a number we shall select so that the constraint equation above is satisfied. Differentiating the equation above with respect to $h(n)$ gives

$$\frac{\partial E(h)}{\partial h(n)} = x(-n) + 2\lambda h(n)$$

and setting this to 0 gives the solution

$$h(n) = \epsilon x(-n)$$

where the constant $\epsilon = 1/2\lambda$ is chosen so that the energy is in unity

$$\epsilon = \frac{1}{\sqrt{\sum x^2(n)}}$$

In other words, the best filter, which is called hte matched filter for the signal, has an IR which is just the time-reverse of the signal sequence. For this filter, the output at time 0 is

$$y(n) = \sum x(-m)x(n-m) = \sum x(m)x(n+m)$$

In this case, the sequence $\{y(n)\}$ is called the autocorrelation sequence of the signal $\{x(n)\}$ and is defined as

$$r_x(n) = \sum x(m)x(n+m)$$

The autocorrelation sequence $\{r_x(n)\}$ has two important properties

1. Its is maximum is at time 0, $r_x(0) > r_x(n), n \neq 0$
2. It is symmetrical about 0, $r_x(-n) = r_x(n)$

The ZT of $\{r_{x(n)}\}$ is

$$R_x(z) = X(z)X(z^{-1})$$

Similarly, we can define the cross-correlation between two sequence $\{a(n)\}, \{b(n)\}$ as

$$r_{ab}(n) = \sum a(m)b(m-n)$$

Note that here we have a $-$, i.e. $\{b(n)\}$ is delayed with respect to $\{a(n)\}$. This is important because in general cross-correlation is not symmetric. In fact

$$r_{ab}(-n) = r_{ba}(n)$$

unless of course the two sequence are the same, $a(n) = b(n)$, in which case we get the autocorrelation.

Among random processes, the most random are the white noise processes, so called because of their spectral properties. If $\{x(n)\}$ is a zero mean stationary white noise process with variance σ^2, then it satisfies

$$Ex(n) = 0, R_x(n) = \sigma^2\delta_{n0}$$

5.8 Noise in Communication Systems: Stochastic Processes

The main application for matched filtering techniques is in the detection of known (deterministic) signals in a background of random disturbances. In order to understand how well they work, it is necessary to find a way of modelling random noise: a sample from a stochastic process $x(t)$ which is random variable for each fixed t.

Except the mean and variance we defined before, the third important statical description of a process is second order: the autocorrelation sequence. Unlike that defined for deterministic signals in the previous section, this must be defined in an appropriate way for a random process, i.e. as as expected value.

$$R(n) = E(x(m) - Ex(m))(x(m-n) - Ex(m-n))$$

Here we have assumed stationarity, i.e. the quantity above depends on only the difference $m, m-n$. Although it is defined in a completely different way from the deterministic autocorrelation before, the autocorrelation for random sequences shares the two properties of the deterministic autocorrelation: symmetry and a maximum at the origin. Like the process mean, it can also be estimated by an appropriate sample average, but getting a reasonable estimate is a complex task, which is known as spectrum estimation, which we do not have space for. It is essential to be clear about the distinction between these two types of correlation sequence: one applies to known, finite

energy signals, the other to random, finite power signals. The ZT of the autocorrelation sequence is defined as

$$S(z) = \sum R(n) z^{-n}$$

The autocorrelation of a process describes how similar it is to its time shifts: how dependent the value at a given time n is on that at a different time. If the process has nonzero mean, however, a large value of the autocorrelation will result even if the variation of the signal is completely random Consequently, it is generally preferable to remove the mean before calculating the correlation. The resulting measure is called the autocovariance. Among random processes, the most random are the white noise processes, so called because of their spectral properties. If $\{\xi(n)\}$ is a sample from a zero mean stationary white noise process with variance σ^2, then it satisfies

$$E\xi(n) = 0, R_\xi(n) = \sigma^2 \delta_{n0}$$

In other words, its autocorrelation sequence is an impulse at the origin. This means that values $\xi(n)\xi(m)$ taken at different times are completely unrelated. The whiteness of the power spectrum can be see by considering either $S(x)$ or equivalently the Fourier transform

$$S(\omega) = S(\exp(j\omega)) = \sigma^2$$

which is a constant dependent on the noise power: the spectrum is flat.

By any large, we shall not deal explicitly with the probability densities, but only work with the expected values, which are a sufficient characterisation of the signal processes for much of the DCSP used at present, from audio analysis to computer vision or radar. Although various probability densities are used, the most widespread is the Gaussian or normal density, as we have discussed before. This has a number of significant features.

1. it is completely characterised by its first and second order statistics. For a single variate, the normal density is given as in Chapter II.

2. Any linear combination of normal variables is also normal. This means that if we filter a normal process, we get another normal process (with different mean and correlation).

3. The sum of a large number of independent random variables is generally approximately normally distributed. This result is called the Central limit Theorem.

Normal processes have been very widely studied and are very useful tool in designing DCSP algorithms of all sorts.

5.8.1 Detection of known signals in noise

We can now turn to the problem which motivated the above summary of the elements of the theory of stochastic processes, namely the detection of a known, finite energy signal $\{x(n)\}$ in noise,

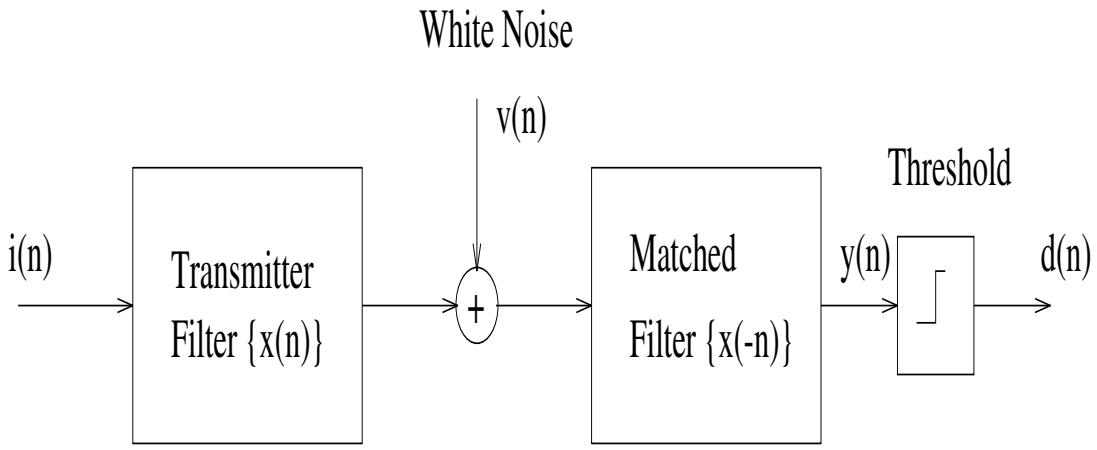

Figure 49: Model of a typical signal detection application, such as radar or communication

which we shall assume is a sample form a zero mean white noise process $\{\xi(n)\}$. The noisy signal is just

$$w(n) = x(n) + \xi(n)$$

with

$$E(\xi(n)x(m)) = x(m)E\xi(n) = 0, E(\xi(n)\xi(m) = \sigma^2\delta_{mn}$$

The overall system is modelled by the example in Fig. 49 in which a sequence of impulse $\{i(n)\}$, which might be at regular or random intervals, drives the transmitter filter, which generates the signal sequence $\{x(n)\}$ in response to each impulse. The detector operates on the stream of signals, corrupted by the channel noise process, using a matched filter, whose output is thresholded to determine whether the signal is present at a given time. The system output is the binary decision sequence $\{d(n)\}$. I na radar system, of course, the input sequence $\{i(n)\}$ would represent the returns from different targets, which would be randomly spaced in time, while in communication system, the impulses would occur at regular intervals, with data being used to modulate their phase or amplitude.

Now, according to our previous calculation, the filter which maximises the output in response to the known signal is just the matched filter for the signal

$$h(n) = \epsilon x(N - n)$$

but the signal in the present case is corrupted by noise, whose effects we must take into account. First, suppose that no signal is present, so $w(n) = \xi(n)$. In signal detection theory, this known as the null hypothesis. In this case, the filter optput at time 0 has statistics

$$E(y(N) = 0$$

and

$$Ey^2(N) = \sigma^2 \sum_{n=0}^{N} h^2(n)$$

To proceed, now consider the case where the signal is present. Then

$$Ey(N) = \epsilon \sum_{n=0}^{N} x^2(n)$$

and

$$Ey^2(N) = (\epsilon \sum_{n=0}^{N} x^2(n))^2 + \sigma^2 \sum_{n=0} N h^2(n)$$

Since the matched filter maximises the mean output when the signal is present and the mean output is 0 when the signal is absent, the matched filter is the best filter for detecting the signal when the noise is white.

5.9 Wiener Filter

If there is a significant amount of noise in the image, then it is better to find a way to deal with the randomness directly than to rely on methods which are essentially ad hoc. The simple way to deal with noise is just to lowpass filter the signal, since as we have already seen most of its energy is at low frequencies. Unfortunately, this will blur the edges too, leading to undesirable results. To do any better, we must use a signal model.

The statistical approach to the problem is based on stochastic models of the image and noise processes. The simplest one to describe is Wiener filtering, named after its inventer. To understand it, we shall start with a very simple problem. Suppose we have a single sample $y = x + \xi$, where x is the desired variable and ξ is a sample of white noise, independent of x with variance σ^2. We assume that the variance of x is 1 and that both signal and noise have zero mean

$$Ex = E\xi = Ex\xi = 0, Ex^2 = 1, E\xi^2 = \sigma^2$$

We want to choose a coefficient a so that the difference between the estimate $\hat{x} = ay$ and x is as low as possible

$$\min_a E[x - ay]^2$$

Differentiating the error with respect to a and equating to zero

$$2E(xy) = 2aEy^2$$

or

$$a = \frac{E(xy)}{Ey^2}$$

at which value the error, $x - \hat{x}$ is orthogonal to the data y, in the sense that

$$E(x - \hat{x})y = Exy - aEy^2 = 0$$

This orthogonality principle can readily be extended to solve the problem of estimating images distorted by noise.

The Wiener formulation works for any type of noise or signal spectrum and any filter. If it has a weakness, it is that the AR signal models which typically get used to not do a good job of representing image features like edges, so that the Wiener filter tends to blur them. This problem can be solved by using a non-stationary formulation, possibly using a state space model to give an optimum recursive non-stationary filter. It goes beyond the scope of this note to go into details.

5.10 Having Fun: Contrast Enhancement

Many images are captured in low light conditions, which can cause then to have low contrast between the darker and lighter regions. This problem can be solved by contrast stretching, in which a nonlinear function such as $f(x) = x^2$ or $f(x) = exp[ax]$ is used to increase the dynamic range of the image. The most sophisticated form of contrast enhancement is known as histogram equalisation, which works by trying to spread out the intensities so that their probability density is as uniform as possible.

Fig. 50 shows a comparison between two figures. The left one is generated from the positive Gaussian random variable, the right is from the uniformly distributed random variable. It is clearly demonstrated that the uniform one has a stronger contrast to us.

It os based on the observation that if u is a random variable with probability density $p_u(x)$, the cumulative distribution of u is defined to be the total probability that u is less than x, ie.

$$P_u(x) = \int_{-\infty}^{x} P_u(y)dy$$

If we define the function

$$f(u) = P_u(u)$$

which is just the cumulative distribution of u, then $f(u)$ itself has a cumulative distribution

$$P_f(x) = P_u(P_u^{-1}(x)) = \begin{cases} x & 0 < x < 1 \\ 0 & x \leq 0 \\ 1 & x \geq 1 \end{cases}$$

which corresponds to a uniform density $pf(x) = 1, 0 \leq x \leq 1$, as intended. Because the intensities are normally integers, rather than real numbers, this equalisation can only be achieved approximately in practice, however. In fig. 51, we show another example.

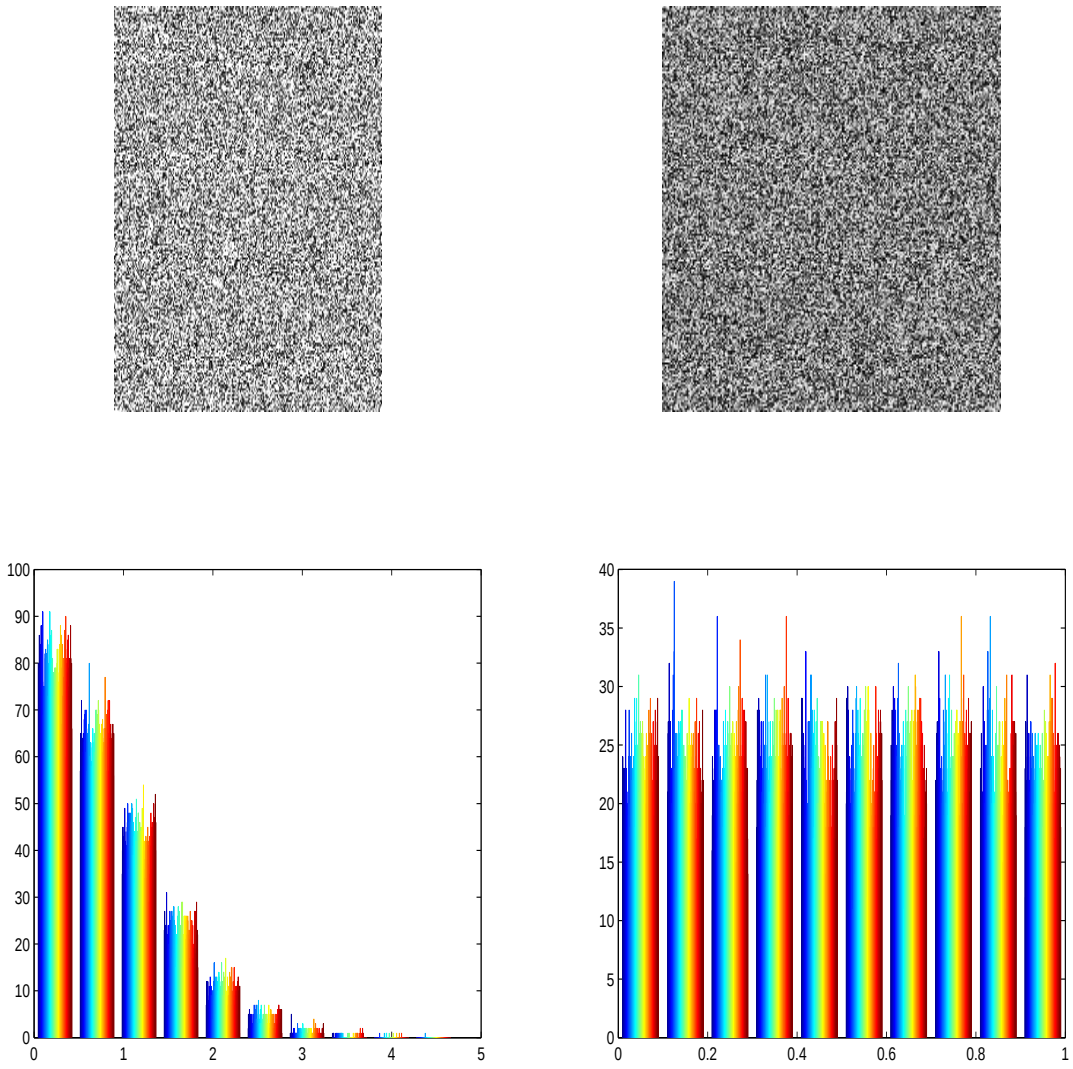

Figure 50: Upper panel: left is from Gaussian, right from uniform. Bottom panel: left is the histogram of Gaussian, right is the histogram of uniform.

Figure 51: Another application.

6 Appendix: Mathematics

6.1 Sinusoids

The term sinusoid is used here to designate a waveform of the type

$$A\cos(2\pi ft + \phi) = A\cos(\omega t + \phi)$$

Thus, a sinusoid is defined as a cosine at amplitude A, frequency ω, and phase ϕ. We could also have defined a sinusoid based on the sine function as $A\sin(2\pi ft + \phi)$, in which case ϕ would be different by 90^o ($\pi/2$ radians). A sinusoid's phase is typically denoted by ϕ, and is in radian units. The instantaneous frequency of a sinusoid is defined as the derivative of the instantaneous phase with respect to time:

$$\omega = \frac{d}{dt}[\omega t + \phi]$$

A discrete-time sinusoid is simply obtained from a continuous-time sinusoid by replacing t by nT:

$$A\cos(2\pi fnT + \phi) = A\cos(\omega nT + \phi)$$

6.2 Complex Numbers

$$
\begin{aligned}
j &= \sqrt{-1} \\
z &= x + jy = r\exp(j\theta) \\
x &= r\cos(\theta) \\
y &= r\sin(\theta) \\
r &= |z| = \sqrt{x^2 + y^2} \\
\theta &= \angle z = \tan^{-1}(y/x) \\
|z_1 z_2| &= |z_1||z_2| \\
|z_1/z_2| &= |z_1|/|z_2| \\
|\exp(j\theta)| &= 1 \\
\angle r &= 0 \\
z_1 z_2 &= (x_1 x_2 - y_1 y_2) + j(x_1 x_2 + y_1 y_2) \\
z_1 z_2 &= r_1 r_2 \exp(j(\theta_1 + \theta_2) \\
\bar{z} &= x - jy = r\exp(-j\theta) \\
z\bar{z} &= |z|^2 = x^2 + y^2 = r^2
\end{aligned}
$$

6.3 The Exponential Function

$$
\begin{aligned}
e^x &= \lim_{n\to\infty}(1 + x/n)^n \\
e^x &= \sum_{n=0}^{\infty} \frac{x^n}{n!} \\
e^{j\theta} &= cos(\theta) + j\sin(\theta) \\
e^{jn\theta} &= cos(n\theta) + j\sin(n\theta) \\
\sin(\theta) &= \frac{e^{j\theta} - e^{-j\theta}}{2j} \\
\cos(\theta) &= \frac{e^{j\theta} + e^{-j\theta}}{2} \\
e &= 2.71828182...
\end{aligned}
$$

6.4 Trigonometric identities

$$
\begin{aligned}
\sin(-\theta) &= -\sin(\theta) \\
\cos(-\theta) &= \cos(\theta) \\
\sin(A+B) &= \sin(A)\cos(B) + \cos(A)\sin(B) \\
\cos(A+B) &= \cos(A)\cos(B) - \sin(A)\sin(B) \\
\sin^2(\theta) &= \frac{1-\cos(2\theta)}{2} \\
\cos^2(\theta) &= \frac{1+\cos(2\theta)}{2} \\
\sin(A)\sin(B) &= \frac{\cos(A-B)-\cos(A+B)}{2} \\
\cos(A)\cos(B) &= \frac{\cos(A+B)+\cos(A-B)}{2} \\
\sin(A)\cos(B) &= \frac{\sin(A+B)+\sin(A-B)}{2} \\
\sin(A)+\sin(B) &= 2\sin\left(\frac{A+B}{2}\right)\cos\left(\frac{A-B}{2}\right) \\
\cos(A)+\cos(B) &= -2\cos\left(\frac{A+B}{2}\right)\cos\left(\frac{A-B}{2}\right)
\end{aligned}
$$

6.5 Spectrum

In this module, we think of filters primarily in terms of their effect on the spectrum of a signal. This is appropriate because the ear (to a first approximation) converts the time-waveform at the eardrum into a neurologically encoded spectrum. Intuitively, a spectrum (a complex function of frequency ω) gives the amplitude and phase of the sinusoidal signal-component at frequency ω. Mathematically, the spectrum of a signal x is the Fourier transform of its time-waveform. Equivalently, the spectrum is the z transform evaluated on the unit circle $z = \exp(j\omega T)$.

We denote both the spectrum and the z transform of a signal by uppercase letters. For example, if the time-waveform is denoted $x(n)$, its z transform is called $X(z)$ and its spectrum is therefore $X(\exp(j\omega T))$. The time-waveform $x(n)$ is said to "correspond" to its z transform $X(z)$, meaning they are transform pairs. This correspondence is often denoted $x(n) \leftrightarrows X(z)$, or $x(n) \leftrightarrows X(\exp(j\omega T)$. Both the z transform and its special case, the (discrete-time) Fourier transform, are said to transform from the time domain to the frequency domain.

We deal most often with discrete time nT (or simply n) but continuous frequency f (or $\omega = 2\pi f$). This is because the computer can represent only digital signals, and digital time-waveforms are discrete in time but may have energy at any frequency. On the other hand, if we were going to talk about FFTs (Fast Fourier Transforms–efficient implementations of the Discrete Fourier Transform, or DFT), then we would have to discretize the frequency variable also in order to represent spectra inside the computer. In this book, however, we use spectra only for conceptual insights into the perceptual effects of digital filtering; therefore, we avoid discrete frequency for simplicity.

When we wish to consider an entire signal as a "thing in itself," we write $x(\cdot)$, meaning the whole time-waveform ($x(n)$ for all n), or $X(\cdot)$, to mean the entire spectrum taken as a whole. Imagine, for example, that we have plotted $x(n)$ on a strip of paper that is infinitely long. Then $x(\cdot)$ refers to the complete picture, while $x(n)$ refers to the nth sample point on the plot.

6.5.1 Fourier's Song

Integrate your function times a complex exponential
It's really not so hard you can do it with your pencil
And when you're done with this calculation
You've got a brand new function - the Fourier Transformation
What a prism does to sunlight, what the ear does to sound
Fourier does to signals, it's the coolest trick around
Now filtering is easy, you don't need to convolve
All you do is multiply in order to solve.
From time into frequency–from frequency to time
Every operation in the time domain
Has a Fourier analog - that's what I claim
Think of a delay, a simple shift in time
It becomes a phase rotation - now that's truly sublime!
And to differentiate, here's a simple trick
Just multiply by J omega, ain't that slick?
Integration is the inverse, what you gonna do?
Divide instead of multiply–you can do it too.
From time into frequency–from frequency to time
Let's do some examples... consider a sine
It's mapped to a delta, in frequency - not time
Now take that same delta as a function of time
Mapped into frequency –of course –it's a sine!
Sine x on x is handy, let's call it a sinc.
Its Fourier Transform is simpler than you think.
You get a pulse that's shaped just like a top hat...
Squeeze the pulse thin, and the sinc grows fat.
Or make the pulse wide, and the sinc grows dense,
The uncertainty principle is just common sense.

6.6 Matlab program for simple filter design

(From Enrico Rossoni)

```
clear all; close all;clc;
Fs = 11025;
Fnoise = Fs/8;
```

```
SNR = -5;
load jazz signal;
samplelength = length(signal);
SAMPLINGFREQUENCY = Fs;
NFFT = min([256, length(signal)]);
NOVERLAP = 0;
WINDOW = HANNING(NFFT);
[Ps,F] = spectrum(signal,NFFT,NOVERLAP,WINDOW,SAMPLINGFREQUENCY);
spectrumsignal = Ps(:,1); Psignal = mean(spectrumsignal);
deltats = 1/Fs;
inoise = min(find(F¿=Fnoise));
fprintf(1,'Signal-to-Noise ratio =
Pnoise = spectrumsignal(inoise) * 10^-SNR/10;
ampnoise = sqrt(Pnoise);
noise = ampnoise*sin(2*pi*Fnoise*(1:samplelength)'*deltats);
Pn = spectrum(noise,NFFT,NOVERLAP,WINDOW,SAMPLINGFREQUENCY);
spectrumnoise = Pn(:,1); Pnoise = mean(spectrumnoise)
noisy = signal + noise;
Pns = spectrum(noisy, NFFT, NOVERLAP,WINDOW, SAMPLINGFREQUENCY);
a1 = 1.3435; a2 = -0.9025;
b0 = 0.9543; b1 = 0.9543*(-1.4142); b2 = 0.9543;
y = zeros(size(noisy)); y(1) = 0; y(2) = 0;
for ii = 3:samplelength
y(ii) = a1*y(ii-1) + a2*y(ii-2) + b0*noisy(ii) + b1*noisy(ii-1) + b2*noisy(ii-2);
end
Pfiltered = spectrum(y, NFFT, NOVERLAP, WINDOW, SAMPLINGFREQUENCY);
figure(2); plot(F*0.001, Pns(:,1), F*0.001, Pfiltered(:,1), 'r');
xlabel('Frequency (kHz)'); ylabel('—X(F)—'); legend('Disturbed Signal','Filtered Signal');
fprintf('Playing filtered audio sample n'); soundsc(y)
```

6.7 Fourier Transform: From Real to Complex Variables

For a signal $x(t)$ with a period T, we have

$$x(t) = \frac{A_0}{2} + \sum_{n=1}^{\infty}[A_n \cos(n\omega t) + B_n \sin(n\omega t)] \tag{6.1}$$

where $\omega = (2\pi)/T$ and A_i, B_i are defined as in Chapter II.

Using the Euler formula as in this Appendix, we have

$$\exp(jn\omega t) = \cos(n\omega t) + j \sin(n\omega t) \tag{6.2}$$

and

$$\begin{aligned} \exp(-jn\omega t) &= \cos(-n\omega t) + j\sin(-n\omega t) \\ &= \cos(n\omega t) - j\sin(n\omega t) \end{aligned} \tag{6.3}$$

where $j^2 = -1$. Eq. (6.2)+Eq. (6.3) gives us

$$\cos(n\omega t) = \frac{[\exp(jn\omega t) + \exp(-jn\omega t)]}{2} \tag{6.4}$$

Eq. (6.2)-Eq. (6.3) yields

$$\sin(n\omega t) = \frac{[\exp(jn\omega t) - \exp(-jn\omega t)]}{2j} \tag{6.5}$$

Substituting Eq. (6.4) and Eq. (6.5) into Eq. (6.1), we obtain

$$\begin{aligned} x(t) &= \frac{A_0}{2} + \sum_{n=1}^{\infty}[A_n\cos(n\omega t) + B_n\sin(n\omega t)] \\ &= \frac{A_0}{2} + \sum_{n=1}^{\infty}\left[A_n\frac{[\exp(jn\omega t) + \exp(-jn\omega t)]}{2} + B_n\frac{[\exp(jn\omega t) - \exp(-jn\omega t)]}{2j}\right] \\ &= \frac{A_0}{2} + \sum_{n=1}^{\infty}\left\{\left[A_n\frac{\exp(jn\omega t)}{2} + B_n\frac{\exp(jn\omega t)}{2j}\right] + \left[A_n\frac{\exp(-jn\omega t)}{2} - B_n\frac{\exp(-jn\omega t)]}{2j}\right]\right\} \\ &= \frac{A_0}{2} + \sum_{n=1}^{\infty}\left\{\left[\frac{A_n}{2} + \frac{B_n}{2j}\right]\exp(jn\omega t) + \left[\frac{A_n}{2} - \frac{B_n}{2j}\right]\exp(-jn\omega t)\right\} \end{aligned} \tag{6.6}$$

If we define

$$C_0 = A_0/2,\ c_n = \frac{A_n}{2} + \frac{B_n}{2j},\ c_{-n} = \frac{A_n}{2} - \frac{B_n}{2j}$$

Eq. (6.6) becomes

$$\begin{aligned} x(t) &= c_0 + \sum_{n=1}^{\infty} c_n\exp(jn\omega t) + \sum_{n=1}^{\infty} c_{-n}\exp(-jn\omega t) \\ &= \sum_{n=-\infty}^{\infty} c_n\exp(jn\omega t) \end{aligned} \tag{6.7}$$

6.8 More Details on Example 8

Given a finite sequence $x[n]$, its Discrete Fourier Transform (DFT) $X(k)$ is defined as

$$X(k) = \text{DFT}\{x[n]\} = \sum_{n=0}^{N-1} x[n]\exp\left[-j(2\pi/N)kn\right], \quad k = 0, \ldots, N-1$$

where N is the lenght of the sequence. Note that by defining the set of angular frequencies $\omega_k = (2\pi/N)k$, $k = 0, \ldots, N-1$, we can rewrite Equation 1 as

$$X(k) = \sum_{n=0}^{N-1} x[n] \exp\left[-j\omega_k n\right] \tag{6.8}$$

The meaning of this formula is clear if you think that the sequences $e_k[n] = \exp(j\omega_k n)$, $k = 0, \ldots, N-1$, form a 'basis' for representing any sequence of N numbers (you can easily check these N sequences are all independent). Therefore, we can interpret $X(k)$ as the 'projection' of our original signal onto the kth sequence of our basis. In conclusion, we can regard the Fourier transform as a simple 'change of coordinate' (from time to frequency, from frequency to time..).

Let us consider for example the sequence $x[n] = (1, 2, -1, -1)$. In this case we have $N = 4$ and $\omega_k = (\pi/2)k$. From Eq. (6.8) we then obtain

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x[n] \exp\left[-j\omega_k n\right] \\ &= x(0)\exp(-j(\pi/2)k \cdot 0) + x(1)\exp(-j(\pi/2)k \cdot 1) + \\ &+ x(2)\exp(-j(\pi/2)k \cdot 2) + x(3)\exp(-j(\pi/2)k \cdot 3) \\ &= 1 + 2 \cdot (-j)^k - 1 \cdot (-j)^{2k} - 1 \cdot (-j)^{3k} \end{aligned}$$

where we have used the Euler formula to get

$$\exp(-j\pi/2) = \cos(-\pi/2) + j\sin(-\pi/2) = -j,$$

and the basic property of the exponential $\exp(nz) = [\exp(z)]^n$. Finally, from the equation above we get $X[k] = (1, 2-3j, -1, 2+3j)$.

As we mentioned above, these complex numbers can be though as coordinates to represent our signal. In fact, for any given finite sequence $x[n]$ we have[1]

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e_k[n]$$

The coefficients in this expansion can be more conveniently rewritten as $X(k) = \rho_k \exp(j\phi_k)$ where $\rho_k = |X(k)|$ is the modulus and $\phi_k = arg(X(k))$ is the phase of $X(k)$. Thus, in order to reconstruct our sequence, we just need to scale each basis sequence by ρ_k, rotate its elements by ϕ_k in the complex plane and finally sum all the resulting sequences up together.

In Figure 52 we plot the signal x respectively in the time (top) and in the z domain (right). In Figure 6.8 we plot the signal in the frequency domain, i.e. the amplitude and the phase of its DFT coefficients $X(k)$ are plotted as a function of the frequency index k. We will usually refer to this as to the *spectrum* of x.

[1]The factor $1/N$ is due to the fact that our basis sequences $\{e_k\}$ are not normalized.

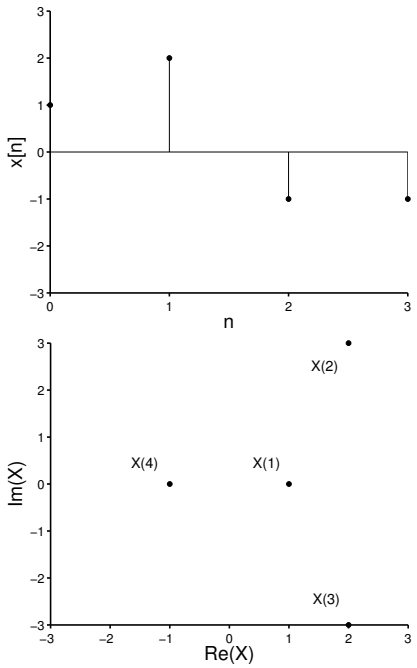

Figure 52: The signal x plotted in the time domain (top) and in the z-domain (bottom).

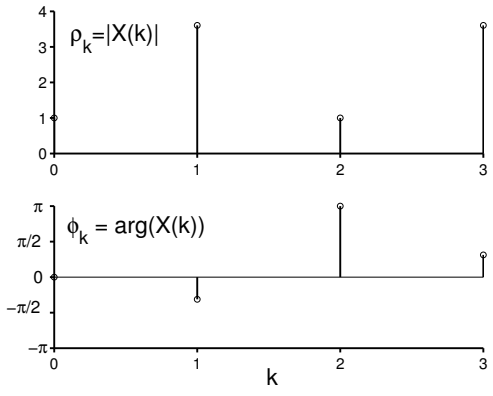

Figure 53: The signal x in the frequency domain.