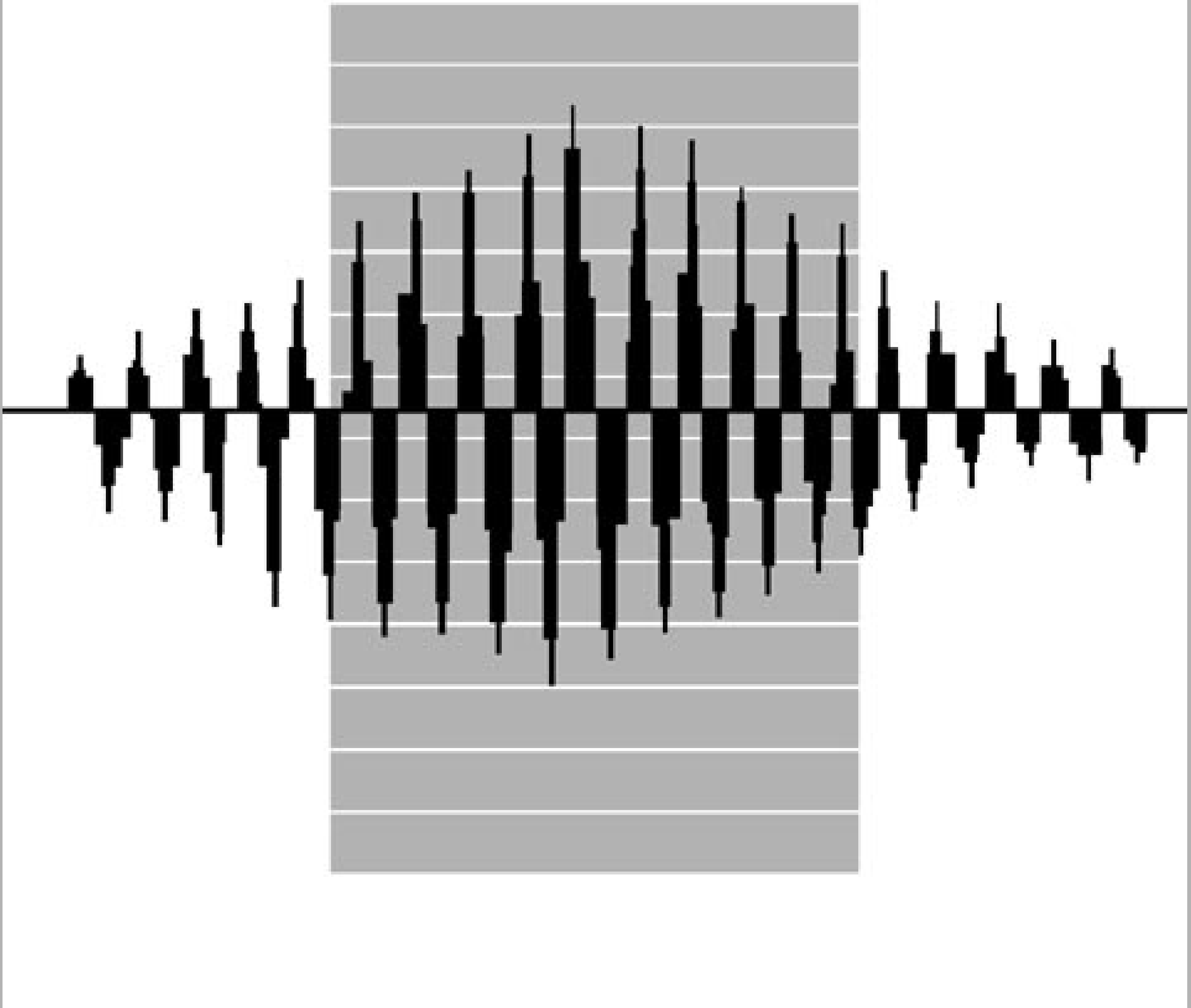
INTRODUCTORY DIGITAL SIGNAL PROCESSING

WITH COMPUTERAPPLICATIONS

STUDENT EDITION



Roelien van Waas
Sven Warris
Tjeerd Andringa

Table of contents

1 Introduction

Introduction of the terms are found in the right margin. The paragraph next to it will introduce and explain the terms.

1.1 Digital signal processing

Digital Signal Processing

Digital signal processing is concerned with the numerical manipulation of signals and data in sampled form. Using elementary operations like digital storage and delay, addition, subtraction, and multiplication by constants, a wide variety of useful functions can be produced.

discrete time signal
sampling instant
sampled-data signal
digital signal

Discrete time signal, a signal which is defined only for a particular set of instants in time, or sampling instants. A discrete time signal can be subdivided into two categories: sampled-data signals, which display a continuous range of amplitude values, and digital signals, in which the amplitude values are quantized in a series of finite steps.

DSP scheme

A typical DSP scheme:

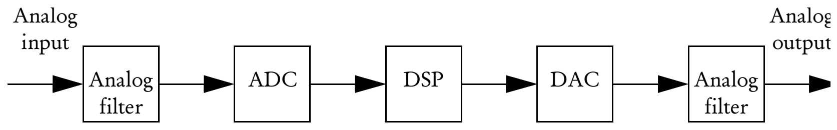


Figure 1
The signal flows from the left to the right. It will pass the filter so the AnalogToDigitalConvertor will work properly. Then some processing is done on the signal, it is converted back to a analog signal and will pass the last filter.
The two filters are necessary to reduce or avoid artifacts due to ADC and DAC. What these are will become apparent in the following chapters

Four reasons for converting signals into digital form and back again:

1. Signals and data of many types are increasingly stored in digital computers, and transmitted in digital form from place to place. In many cases it makes sense to process them digitally as well
2. Digital processing is inherently stable and reliable. It also offers certain technical possibilities not available with analog methods
3. Rapid advances in integrated circuit design and manufacture are producing ever more powerful DSP devices at decreasing cost
4. In many cases DSP is used to process a number of signals simultaneously

1.2 Properties of DSP's

linear

All the DSP's in this book are linear. A linear system, or processor, may be defined as one which obeys the Principle of Superposition:

Definition 1
Principle of Superposition

> *If an input consisting of the weighted sum of a number of signals is applied to a linear system, then the output is the weighted sum, or superposition, of the systems responses to each signal considered separately.*

frequency-preservation

A major property of linear systems, which is closely relate to the Principle of Superposition, is known as frequency-preservation. It means that if an input signal containing certain frequencies is applied to a linear system, the output can contain only the same frequencies and no others.

$$y_1[n] = x_1[n], y_2[n] = x_2[n] \Leftrightarrow ax_1[n] + bx_2[n] = ay_1[n] + by_2[n]$$

Eq. 1.1
linear system

time-invariant

All the DSP's in this book are not only linear but also time-invariant, which means that the system doesn't vary with time. The only effect of a time-shift in the input signal to the system is a corresponding time-shift in its output.

$$y[n] = x[n] \Leftrightarrow y[n-N] = x[n-N]$$

Eq. 1.2
time invariant system

linear time-invariant
LTI

The systems in this book are so-called linear time-invariant (LTI) systems, which involve the following types of operations on input and/or output samples:

- storage/delay
- addition/subtraction
- multiplication by constants

Other properties are, see Example 1.3 (pg. 28):

- **Causality**: the output signal depends only on present and/or previous values of the input
  $y[n] = 2x[n]$
- **Stability**: in response to a bounded input a bounded output is produced (see Figure 2)
- **Invertibility**: if input *x[n]* gives an output *y[n]*, then there is an inverse which would produce *x[n]* if given input *y[n]*
- **Memory**: the present output *y[n]* depends upon one or more previous (but a finite number of) input values *x[n-1]*, *x[n-2]*, .. , *x[0]*
  $y[n] = x[n] + x[n-1]$

causality
stability
invertibility
memory

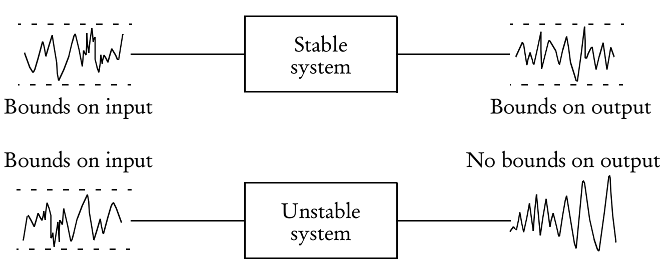


Figure 2
Bounded input, bounded output stability

1.3 Sampling

aliasing

Sampling a analog signal causes the original spectrum to repeat around multiples of the sampling frequency. If the sampling frequency is chosen to small, the successive spectral repetitions overlap, this is called aliasing. In these cases the original signal cannot be fully reconstructed.

> *A analog signal containing components up to some maximum frequency f_1 Hz may be completely represented by regularly-spaced samples, provided the sampling rate is at least $2f_1$ samples per second.*

Theorem 1
Sampling theorem

Nyquist frequency
Nyquist rate
folding frequency

The maximum frequency f_1 contained in an analog signal is widely referred to as the Nyquist frequency. The minimum sampling rate ($2f_1$ samples per second) is known as the Nyquist rate. And the so-called, folding frequency, which equals half the sampling frequency actually used, is the highest frequency which can be adequately represented according to the Sampling theorem. When the theorem is obeyed justly, the Nyquist and folding frequencies are equal.

1.3.1 Quantization

quantization levels
quantization noise

A digital signal is represented by binary numbers. If the binary is of length N it will allow 2^N separate numbers. So if an analog number, which has a continuous range of amplitudes, is sampled errors are introduced. The total range of possible amplitudes is divided into quantization levels (or slots). A 3-bit code has therefore $2^3 = 8$ quantization levels.When more bits are used, a greater precision can be reached. The noise introduced by this coding process is called quantization noise and has a maxima of plus or minus half a quantization level. This noise can not be removed from the new digital signal (e.g. the analog signal cannot be reconstructed).

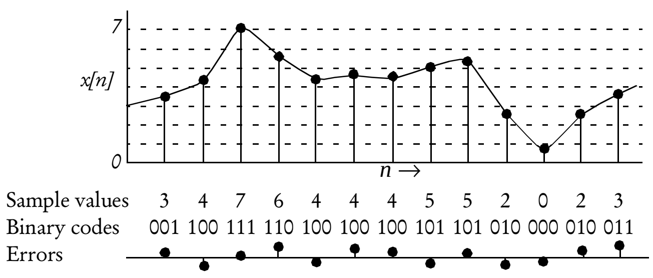


Figure 3
Converting an analog signal into a binary code.

1.4 Basic types of digital signals

The three types of digital signals are the unit impulse, unit step and unit ramp function. These are defined as follows:

$$u[n] = 0, n < 0$$
$$u[n] = 1, n \geq 0$$

Eq. 1.3
unit step function

$$\delta[n] = 0, n \neq 0$$
$$\delta[n] = 1, n = 0$$

Eq. 1.4
unit impulse function

The relationship between these two can be written formally, using linearity, as:

$$u[n] = \sum_{m=-\infty}^{n} \delta[m]$$

Eq. 1.5

$$\delta[n] = u[n] - u[n-1]$$

$$r[n] = n \cdot u[n]$$

Eq. 1.6
unit ramp function

Other simple digital signals may be built up by summation of these basic functions, see Example 1.1 (pg. 15).

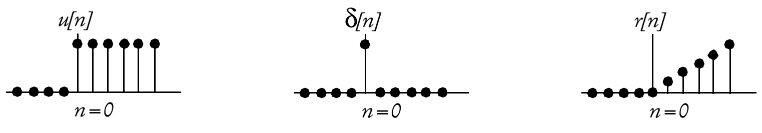


Figure 4
From the left to the right: unit step, unit impulse and unit ramp

The exponential:

$$x[n] = Ae^{\beta n} \qquad \textbf{Eq. 1.7}$$

exponential

where A and β are constants.

If β is purely imaginary the sine and cosine signals can be generated.

Suppose

$$\beta = j\Omega \qquad \textbf{Eq. 1.8}$$

then

$$x_1[n] = Ae^{jn\Omega} = A\cos(n\Omega) + jA\sin(n\Omega) \qquad \textbf{Eq. 1.9}$$

Suppose

$$\beta = -j\Omega \qquad \textbf{Eq. 1.10}$$

then

$$\begin{aligned} x_2[n] &= Ae^{-jn\Omega} = A\cos(-n\Omega) + jA\sin(-n\Omega) \\ \cos(-n\Omega) &= \cos(n\Omega) \\ \sin(-n\Omega) &= -\sin(n\Omega) \end{aligned} \qquad \textbf{Eq. 1.11}$$

Using

$$x_1[n] + x_2[n] = 2A\cos(n\Omega) \qquad \textbf{Eq. 1.12}$$

gives

$$A\cos(n\Omega) = \frac{A}{2}e^{jn\Omega} + \frac{A}{2}e^{-jn\Omega} \qquad \textbf{Eq. 1.13}$$

Using

$$x_1[n] - x_2[n] = 2jA\sin(n\Omega) \qquad \textbf{Eq. 1.14}$$

gives

$$A\sin(n\Omega) = \frac{A}{2j}e^{jn\Omega} - \frac{A}{2j}e^{-jn\Omega} \qquad \textbf{Eq. 1.15}$$

1.5 Filters

- low-pass filter, a smoothing, or averaging, filter used for removing unwanted interference or noise, from a relatively slowly varying signal — low-pass filter
- bandstop or notch filter, a filter that rejects a narrow band of frequencies — bandstop or notch filter
- bandpass filter, a filter that passes a particular band of frequencies relatively strongly — bandpass filter
- anti-aliasing filter, a filter with low-pass characteristics, which is designed to ensure that the maximum frequency $f_1 = \frac{1}{2T}$ cannot be exceeded — anti-aliasing filter

A widely used technique for smoothing is the moving average, where each smoothed value is computed as the average of a number of preceding raw-data values, and the process is repeated sample-by-sample through the record.

smoothing
moving average

2 Time-domain analysis

Basic techniques are developed for describing digital signals and digital processors in the time domain. Foremost among them is the convolution process, which allows us to compute the output signal from a LTI processor in response to an arbitrary input signal.

2.1 Convolution

Every signal *x[n]* can be written as the superposition, or summation of the more basic impulse signal, weighted with the appropriate value of *x[n]* and shifted by a number of sampling intervals.

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

Eq. 2.1

impulse response *h[n]*

A very important role in linear DSP is played by the impulse response *h[n]*. If the input to a linear processor is δ*[n]*, the output must be *h[n]*. The impulse response can have a wide variety of different forms, depending upon the processing, or filtering, action of the system. See Example 2.1 (pg. 36).

step response *s[n]*

Just as the unit step function is the running sum of the unit impulse δ*[n]*, so is the output of a LTI processor the running sum of its impulse response. See Example 2.2 (pg. 39).

$$s[n] = \sum_{m=-\infty}^{\infty} h[m]$$

Eq. 2.2

Alternatively $h[n] = s[n] - s[n-1]$.

It is shown how digital signals are described as sets of impulse functions, and how to characterize LTI processors by their impulse or step responses. These properties enable the description of a LTI systems response to any nontrivial input. This is done by the convolution sum:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

Eq. 2.3
convolution sum

Thus because the input signal is a superposition of impulses, the output can be described as the superposition of the impulse response.

The process of convolution is often denoted by an asterisk or a ⊗, and is commutative:

$$x[n] \otimes h[n] = h[n] \otimes x[n]$$

Eq. 2.4
commutative property

Two further basic aspects of convolution are its associative and distributive properties.

The associative property may be summarised as:

$$x[n] \otimes (h_1[n] \otimes h_2[n]) = x[n] \otimes h_1[n] \otimes h_2[n]$$

Eq. 2.5
associative property

It implies that a cascaded combination of two or more LTI systems can be condensed into a single LTI system. Where the associative property has important implication for cascaded LTI systems, the distributive property has important implication for LTI systems in parallel. The property can be summarised as:

$$x[n] \otimes (h_1[n] + h_2[n]) = (x[n] \otimes h_1[n]) + (x[n] \otimes h_2[n])$$

Eq. 2.6
distributive property

This means that two (or more) parallel systems are equivalent to a single system whose impulse response equals the sum of the individual impulse responses.

2.2 Transients in LTI processors

start-up transient
stop transient

No real-life signal continues forever. Hence practical DSP is concerned with signals which are effectively 'switched on' at one instant, and 'switched of' again later. When an input signal is first switched on, and applied to a digital processor with memory, it generates a start-up transient. When the signal is switched off again, a stop transient is produced. These transient should not be regarded as unnecessary, avoidable nuisance.

Transients are important for several reasons:

- A start-up transient may mask the initial portion of an output signal, preventing us from seeing the desired response
- We often assume that the output signal of a digital processor is initially zero. But this will only be true if it has 'settled' following any previous input, in other words, any stop transient must have died away
- Transients are closely related to the 'natural' response of a system, and to impulse responses. The give further valuable insights into the behaviour of linear processors

steady-state response

In between start-up and stop transients a system shows its steady-state response.

2.3 Difference equations

difference equation

The operation of a digital processor in the time-domain can be described as a difference equation. The most general form of this equation is given by:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]$$

Eq. 2.7
difference equation

The complexity of an LTI processor depends on the number of terms on each side of the equation. The value of N, which indicates the highest-order difference of the output signal, is generally referred to as the order of the system.

auxiliary conditions

In order to solve this equation, however, some auxiliary conditions, also called boundary conditions, are needed.

homogeneous solution
particular solution

The output signal can be obtained by separation of the equation into an homogeneous and particular solution. The homogeneous solution accounts for any nonzero auxiliary conditions, and for transients caused by switching an input signal on and off. The particular solution represents the steady-state response of the system to a continuing input signal.

This particular solution is the solution of the difference equation with all the auxiliary conditions put to zero.

The homogeneous solution is the solution of the difference equation using the auxiliary conditions and under the additional condition of zero input. See Example 2.4 (pg. 57).

The superposition of this homogeneous and particular solution will give the complete output of the system.

3 Frequency Domain Analysis

Three reasons why a frequency-domain approach is needed:

- Sinusoidal and exponential signals occur in the natural world, and in the world of technology. Even when a signal is not of this type, it can be analysed into component frequencies. The response of an LTI processor to each such component is quit simple. It can only alter the amplitude and the phase, not the frequency. The overall output signal can then be calculated by superposition
- If an input signal is described by its frequency spectrum, and an LTI processor by its frequency response, then the output signal spectrum is found by multiplication. This is generally simpler to perform, and to visualise, then the equivalent time-domain convolution
- The design of DSP algorithms and systems often starts with a frequency domain specification. In other words, it specifies which frequency ranges in an input signal are to be enhanced, and which suppressed

3.1 The discrete Fourier series

A vector can always be described by the sum of a number of linearly independent vectors. For example: the most obvious description of a vector $\vec{x}$ in $\Re^3$ is:

$$\vec{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = a\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + b\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + c\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Eq. 3.1
Basevectors of $\Re^3$

Before we can use this, some linear algebra is introduced. The definitions and theorems should be known to the students, so no proof is given.

Definition 2
Linearly dependent

A subset S *of vector space* V *is said to be linearly dependent if there exist a finite number of distinct vectors* $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$ *in* S *and scalars* $a_1, a_2, \ldots, a_n$, *not all zero, such that* $a_1\vec{x}_1 + a_2\vec{x}_2 + \ldots a_n\vec{x}_n = 0$. *In this case we will also say that the elements of* S *are linearly dependent.*

This gives the following definition of linearly independent:

Definition 3
Linearly independent

A subset S *of a vector space that is not linearly dependent is said to be linearly independent. As before, we will also say that the elements of* S *are linearly independent in this case.*

A subset *S* of a vector space *V* that is linearly independent and generates *V* possesses a very useful property: every element of *V* can be expressed in one and only one way as a linear combination of elements of *S*. It is this result that makes linearly independent generating sets the building blocks of vector spaces.

Definition 4
Vector space basis

A basis β *for a vector space* V *is a linearly independent subset of* V *that generates* V. *(If* β *is a basis for* V, *we also say that the elements of* β *form a basis for* V.*)*

The following theorem shows the most significant property of a basis:

Theorem 2

Let V *be a vector space and* $\beta = \{\vec{x}_1 \ldots \vec{x}_n\}$ *be a subset of* V. *Then* β *is a basis for* V *if and only if each vector* $\vec{y}$ *in* V *can be uniquely expressed as a linear combination of vectors in* β, *i.e., can be expressed in the form* $\vec{y} = a_1\vec{x}_1 + \ldots a_n\vec{x}_n$ *for unique scalars* $a_1, \ldots, a_n$.

3.1.1 Trigonometric form of the Fourier series

Fourier Theorem
Fourier Series

Now why go through all this trouble? Well, maybe this algebra helps us to find a better way of describing periodic signals. It is important to see that a periodic digital signal can be seen as a vector in $\Re^N$. Harmonic sinuses and cosinuses are linearly independent (e.g. the set of functions $\sin n\omega_0 t$ and $\cos n\omega_0 t$ for $n = 0, 1, 2, 3, \ldots$ are orthogonal over any time interval equal to one complete period of the fundamental frequency ω_0), so if we create a basis consisting of an infinite number of harmonic sinuses and cosinuses it can be shown (Fouriers Theorem) that any periodic signal can be described as the weighted sum of these sinuses and cosinuses (the Fourier series):

$$x[n] = \frac{a_0}{2} + a_1 \cos \omega n + \ldots + a_k \cos k\omega n + \ldots + b_1 \sin \omega n + \ldots + b_k \sin k\omega n + \ldots$$

Eq. 3.2
Fourier Series

Or, as a sum:

$$x[n] = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos k\omega n + \sum_{k=1}^{\infty} b_k \sin k\omega n$$

Eq. 3.3

where:

$$a_k = \frac{2}{N} \sum_{n=0}^{N-1} x[n] \cos k\omega n$$

$$b_k = \frac{2}{N} \sum_{n=0}^{N-1} x[n] \sin k\omega n$$

Eq. 3.4

and:

$$\omega = \frac{2\pi}{N}$$

Eq. 3.5

Fourier Coefficients

Thus the Fourier series are the linearly independent basis that generates the vector space $\Re^N$. Any periodic signal of length N can be described using this basis by an unique set of scalars (Fourier Coefficients).

harmonic

In Eq. 3.2 $k\omega$ is called the kth harmonic with a_k (or b_k) its amplitude.

3.1.2 Exponential form of the Fourier Series

It is now obvious that the trigonometric form of the Fourier Series can be written in the exponential form:

$$x[n] = \sum_{k=-\infty}^{\infty} d_k e^{jk\omega n}$$

Eq. 3.6
Exponential Fourier Series

in which

$$d_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk\omega n}$$

Eq. 3.7

3.1.3 Properties of the Fourier Series

A double arrow is used to denote the relationship between a signal *x[n]* and its spectrum a_k (*x[n]* $\leftrightarrow a_k$).

linearity

- **linearity:**
  if $x_1[n] \leftrightarrow a_k$ and $x_2[n] \leftrightarrow b_k$
  then $Ax_1[n] + Bx_2[n] \leftrightarrow Aa_k + Bb_k$

time-shifting

- **time-shifting:**
  if $x[n] \leftrightarrow a_k$
  then $x[n - n_0] \leftrightarrow a_k e^{-\frac{2j\pi k n_0}{N}}$

differentiation

- **differentiation:**
  if $x[n] \leftrightarrow a_k$
  then $x[n] - x[n-1] \leftrightarrow a_k\left(1 - e^{-\frac{2j\pi k}{N}}\right)$

- **integration**: integration
  if $x[n] \leftrightarrow a_k$
  then $\sum_{k=-\infty}^{n} x[k] \leftrightarrow a_k\left(1 - e^{-\frac{2j\pi k}{N}}\right)^{-1}$
- **convolution**: convolution
  if $x_1[n]$ and $x_2[n]$ have the same period
  and if $x_1[n] \leftrightarrow a_k$ and $x_2[n] \leftrightarrow b_k$
  then $\sum_{m=0}^{N-1} x_1[m]x_2[n-m] \leftrightarrow Na_kb_k$
  Convolution in the time-domain is equivalent to multiplication in the frequency-domain.
- **modulation**: modulation
  if $x_1[n] \leftrightarrow a_k$ and $x_2[n] \leftrightarrow b_k$
  then $x_1[n]x_2[n] \leftrightarrow \sum_{m=0}^{N-1} a_m b_{k-m}$
  Multiplication in the time-domain is equivalent to convolution in the frequency-domain

3.2 The Fourier Transform

Most practical digital signals are a-periodic, that is they are not strictly repetitive. In practise, signals are of finite length therefore the FT's of these signals are approximations. To obtain the spectrum of an a-periodic signal the Fourier Transform is to be used:

$$\mathcal{X}(\Omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega n}$$

Eq. 3.8
Fourier transform

The inverse transform is:

$$x[n] = \frac{1}{2\pi}\int_{2\pi} \mathcal{X}(\Omega)e^{j\Omega n}d\Omega$$

Eq. 3.9
inverse Fourier transform

The Fourier Transform is much like the Fourier Series. The only difference is that the Fourier Transform has an infinite long basis and thus contains the

Block 1: The Fourier Transform for basic signals

- unit impulse δ*[n]*,

$$\mathcal{X}(\Omega) = \sum_{n=-\infty}^{\infty} \delta[n]e^{-i\Omega n} = \delta[0]e^{0} = 1$$

basis an infinite number of vectors.

3.2.4 Properties of the Fourier Transform

- **linearity**: linearity
  if $x_1[n] \leftrightarrow \mathcal{X}_1(\Omega)$ and $x_2[n] \leftrightarrow \mathcal{X}_2(\Omega)$
  then $ax_1[n] + bx_2[n] \leftrightarrow a\mathcal{X}_1(\Omega) + b\mathcal{X}_2(\Omega)$

- **time-shifting:** time-shifting
  if $x[n] \leftrightarrow \mathcal{X}(\Omega)$
  then $x[n-n_0] \leftrightarrow \mathcal{X}(\Omega)e^{-j\Omega n_0}$
- **convolution:** convolution
  if $x_1[n] \leftrightarrow \mathcal{X}_1(\Omega)$ and $x_2[n] \leftrightarrow \mathcal{X}_2(\Omega)$
  then $x_1[n] \otimes x_2[n] \leftrightarrow \mathcal{X}_1(\Omega)\mathcal{X}_2(\Omega)$

3.3 Frequency Response

In the time domain, the input signal *x[n]* is convolved with the impulse response *h[n]* to produce the output signal *y[n].* The convolution property of the transform dictates that the equivalent in the frequency-domain must be a multiplication. The output signal spectrum *$\mathcal{Y}(\Omega)$* is simply the product of the input spectrum *$\mathcal{X}(\Omega)$* and a function *$\mathcal{H}(\Omega)$* representing the system.

$$\mathcal{Y}(\Omega) = \mathcal{X}(\Omega)\mathcal{H}(\Omega)$$

Eq. 3.10
output signal spectrum

$\mathcal{H}(\Omega)$ is known as the frequency response:

$$\mathcal{H}(\Omega) = \frac{\mathcal{Y}(\Omega)}{\mathcal{X}(\Omega)} = \frac{\sum_{k=0}^{M} b_k e^{-jk\Omega}}{\sum_{k=0}^{N} a_k e^{-jk\Omega}}$$

Eq. 3.11
frequency response

4 The Z-Transform

The z-transform offers a valuable set of techniques for the frequency analysis of digital signals and processors. It is also very useful in design.

4.1 The Z-Transform

The Z-transform is defined as:

$$X(z) = \sum_{n=0}^{\infty} x[n]z^{-n}$$

Eq. 4.1
z-transform

unilateral

This is the unilateral version, which is normally adequate. The bilateral version with summation limits of ±∞ is only needed occasionally and has very strict convergence conditions.

The z-transform is quite easy to visualise. *X(z)* is essentially a power series in z^{-1}, with coefficients equal to successive values of the time-domain signal *x[n]*. See example 4.1 (pg. 99).

time-shift operator

A very simple way of thinking of *z* is as a time-shift operator. Multiplication by *z* is equivalent to a time advance by one sampling interval. Division by *z* is equivalent to a time delay by the same amount.

Block 2: The z-transform on basic signals

- unit impulse:

  $X(z) = \delta[n]z^{-n} = 1$
- unit step:

  $$X(z) = \sum_{n=0}^{\infty} 1z^{-n} = 1 + z^{-1} + z^{-2} + \ldots = \frac{1}{1-z^{-1}} = \frac{z}{z-1}$$
- unit ramp:

  $$X(z) = \sum_{n=0}^{\infty} nz^{-n}$$

transfer function

The equivalent of the frequency response function *H(Ω)* is the, so called, transfer function *H(z)*, where $Y(z) = X(z)H(z)$.

4.1.1 Properties of the z-transform

- **linearity**:

  linearity

  if $x_1[n] \leftrightarrow X_1(z)$ and $x_2[n] \leftrightarrow X_2(z)$

  then $ax_1[n] + bx_2[n] \leftrightarrow aX_1(z) + bX_2(z)$
- **time-shifting**:

  time-shifting

  if $x[n] \leftrightarrow X(z)$

  then $x[n-n_0] \leftrightarrow X(z)z^{-n_0}$
- **convolution**:

  convolution

  if $x_1[n] \leftrightarrow X_1(z)$ and $x_2[n] \leftrightarrow X_2(z)$

  then $x_1[n] \otimes x_2[n] \leftrightarrow X_1(z)X_2(z)$

- **final-value theorem:**

  final-value theorem

  if $x[n] \leftrightarrow \mathbf{X}(z)$

  then $\lim_{n \to \infty} x[n] = \lim_{z \to 1} \frac{z-1}{z}\mathbf{H}(z)$

With the final-value theorem the final, steady-state, response of a system to a step input can be calculated (see page 106 of the book).

4.1.2 Z-plane poles and zeros

rational function

A z-transform used to describe a real digital signal or an LTI system is always a rational function of the frequency variable z. In other words it can always be written as the ratio of numerator and denominator polynomials in z:

$$\mathbf{X}(z) = \frac{\mathbf{N}(z)}{\mathbf{D}(z)}$$

Eq. 4.2
rational description of $\mathbf{X}(z)$

This is apart from a gain factor K, the transform can be completely specified by the roots of $\mathbf{N}(z)$ and $\mathbf{D}(z)$.

$$\mathbf{X}(z) = \frac{\mathbf{N}(z)}{\mathbf{D}(z)} = \frac{K(z-z_1)(z-z_2)(z-z_3)(z-z_4)\ldots}{(z-p_1)(z-p_2)(z-p_3)(z-p_4)\ldots}$$

Eq. 4.3
zeros and poles

zeros
poles

The constants $z_1, z_2, z_3, \ldots$ are called zeros of $\mathbf{X}(z)$, because they are the values of z for which $\mathbf{X}(z)$ is zero. Conversely $p_1, p_2, p_3, \ldots$ are known as the poles of $\mathbf{X}(z)$, giving values of z for which $\mathbf{X}(z)$ tends to infinity.

A very useful representation of a z-transform is obtained by plotting its poles and zeros in the complex plane (Argand diagram). Zeros are shown as an open circular symbol, and a pole as a cross or asterisk.

stability

- A circle of unit radius centred at the z-plane origin is drawn, as this indicates the stability of a system.
- A system or signal is only stable if all the poles lie inside the unit circle.
- Zeros or poles at the origin of the z-plane produce a pure time advance (or delay), but have no other effect of the characteristics of the processor or signal.
- Usually a minimum-delay system is preferred, which can be achieved by ensuring that there is an equal number of poles and zeros.

About the frequency variable z can be noted that it is equivalent to $e^{j\Omega}$ in Fourier notation. Therefore if we put $z = e^{j\Omega}$, where Ω is real, effectively converting a z-transform to the exponentials (or sines and cosines) of Fourier analysis.

As $e^{j\Omega}$ always has unit magnitude, irrespective of the value of Ω, it always lies on the unit circle. The actual place on this circle depends on the frequency (as Ω increases, the point moves anti-clockwise around the unit circle, the angle made with the positive real axis equals the value of Ω).

This representation can also provide information about the magnitude and phase of the spectral function. A vector has to be drawn form each pole-vector and zero-vector to the point on the unit circle representing the sinusoidal frequency of interest.

magnitude

- The magnitude of the spectral function equals the product of all zero-vector lengths, divided by the product of all pole-vector lengths. (An additional gain vector may have to be taken into account, but this only alters the scale of the function, not the shape)

phase

- The phase equals the sum of all zero-vector phases, minus the sum of all pole-vector phases

See Example 4.4 on page 113 of the book.

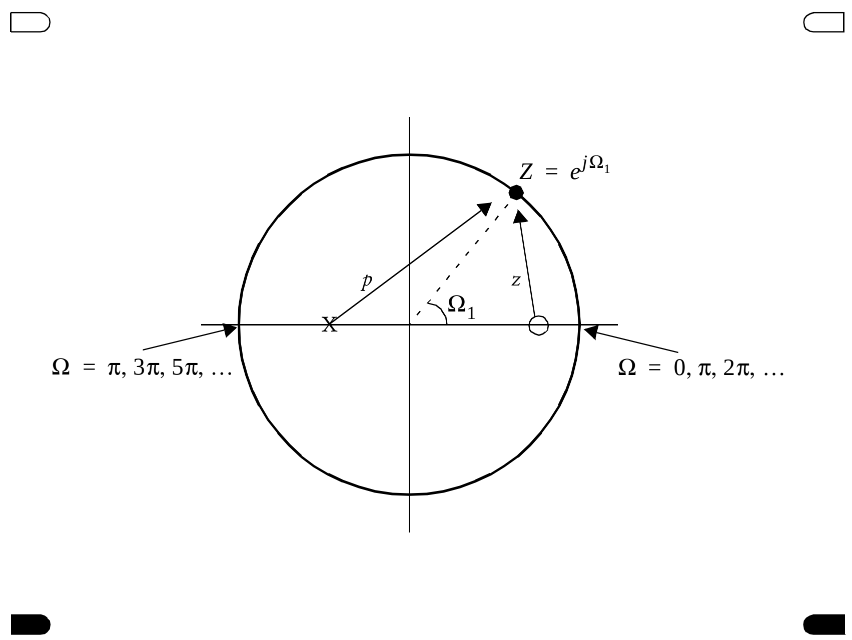

$Z = e^{j\Omega_1}$
p
z
Ω_1
$\Omega = \pi, 3\pi, 5\pi, \dots$
$\Omega = 0, \pi, 2\pi, \dots$

Figure 5
The effect of poles and zeros close to the unit circle can be visualized. As the frequency varies, and we move around the unit circle, the spectral magnitude function peaks whenever we pass close to a pole. It goes through a minimum whenever we pass close to a zero. Note that zeros can occur actually on the unit circle, giving rise to true nulls at the corresponding frequencies. However, the poles of a stable system are always inside the unit circle, so its response must be finite at all frequencies.

4.2 First- and second-order LTI systems

The transfer function of a high-order system in terms of its poles and zeros is written as:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{K(z-z_1)(z-z_2)(z-z_3)\dots}{(z-p_1)(z-p_2)(z-p_3)}$$

Eq. 4.4
transfer function

$H(z)$ may in principle be built up by cascading a number of first and second-order subsystems defined by:

$$H_1(z) = \frac{z-z_1}{z-p_1}$$

Eq. 4.5
first-order system

$$H_2(z) = \frac{(z-z_2)(z-z_3)}{(z-p_2)(z-p_3)}$$

Eq. 4.6
second-order system

If we consider only first and second-order systems with their zeros at the origin, the location of a first-order real pole is then written as $z = \alpha$; and of a complex-conjugate pole-pair as $z = re^{j\theta}$ and $z = re^{-j\theta}$, so

$$H_1(z) = \frac{z}{z-\alpha}$$

Eq. 4.7

$$H_2(z) = \frac{z^2}{(z-re^{j\theta})(z-re^{-j\theta})} = \frac{z^2}{z^2 - 2rz\cos\theta + r^2}$$

Eq. 4.8

4.2.3 First-order systems.

low-pass filter
high-pass filter

The single real pole must be inside the unit circle to ensure stability. If it is on the positive real axis, we get an elementary recursive low-pass filter with a peak response at $\Omega = 0$ (see Eq. 4.7). If the pole is on the negative real axis, the equivalent high-pass filter is obtained with a peak response at $\Omega = \pi$. The pole-zero configuration, frequency response (magnitude and phase), and impulse response of typical low-pass and high-pass systems are illustrated in the book on pg. 116).

4.2.4 Second-order systems

center frequency
bandwidth

There is a second-order zero at the origin and a complex-conjugate pole pair at $z = re^{\pm j\theta}$ (see Eq. 4.8). The frequency at which the peak gain occurs (the center-frequency) is determined by the parameter θ; and the selectivity, or bandwidth, of the system by the parameter r.

Some general observations:

- when $\theta = 0$, it is a typical low-pass filter
- when r is moving towards 1 (the poles are moving towards the unit circle), this gives a very selective frequency-domain characteristic
- when $\theta = \pi$, it is a typical high-pass filter

4.3 Nonzero auxiliary conditions

The unilateral z-transform can also cope with nonzero auxiliary conditions. From the DSP point of view this is useful in two closely-related situations:

- A processor may not have settled following application of a previous input signal. In other words the stop transient has not died away
- The input signal was in fact applied prior to $n = 0$ and we need to assess its subsequent effects on the output

The transform cannot take detailed account of past history, but it can summarise its effects by incorporating nonzero initial conditions. A first-order processors past history can be summarized in terms of a single initial condition. In the case of a second-order system, two initial condition terms $y[-1]$ and $y[-2]$ are needed.

However an LTI systems response to a unit impulse is only its ‘true’ impulse response $h[n]$ if the initial conditions are zero. And likewise, the ratio of output to input z-transforms only equals the transfer function $H(z)$ for zero initial conditions.

5 Non-recursive digital filters

5.1 General design information

General remark: almost any DSP algorithm or processor can reasonably be described as a ‘filter’. However, the term is most commonly used for systems that transmit (or reject) well-defined frequency ranges.

The general form of difference equation for a causal LTI processor is given by:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]$$

Eq. 5.1 causal LTI processor

In a nonrecursive filter the output depends only on present and previous inputs, so the difference equation may be written in the form:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k]$$

Eq. 5.2 nonrecursive filter

The transfer function and frequency response function of a nonrecursive filter are given by respectively:

$$H(z) = \sum_{k=0}^{M} b_k z^{-k}$$

Eq. 5.3 transfer function

$$H(\Omega) = \sum_{k=0}^{M} b_k e^{-jk\Omega}$$

Eq. 5.4 frequency response

Finite Impulse Response FIR

Such a filter directly implements the convolution sum, and the coefficient b_k are simply equal to successive terms in its impulse response. As the number of coefficients must be finite, a practical nonrecursive filter is often referred to as FIR (finite impulse response). The art of designing a nonrecursive filter is to achieve an acceptable performance using as few coefficients b_k as possible. Practical filters typically need between (say) 10 and 150 coefficients.

Nonrecursive filters are slower in operation that most recursive designs but there are two major compensating advantages:

- A nonrecursive filter is inherently stable. Its transfer functions is specified in terms of z-plane zeros only, so there is no danger that inaccuracies in the coefficients may lead to instability (stability)
- Since a nonrecursive filter has a finite impulse response (FIR), the latter can be made symmetrical in form. This produces an ideal linear-phase characteristic, equivalent to a pure time-delay of all frequency components passing through the filter. There is said to be no phase-distortion (phase distortion)
- A FIR leads to finite start-up and stop transients

High-pass and band-pass designs can be readily derived from a low-pass prototype.

5.2 The Fourier Transform method

Eq. 3.8 descripes the Fourier transform and Eq. 3.9 the inverse Fourier transform. Only the second equation is relevant here. If we rewrite this equation to descibe a LTI processor rather than a signal, we have:

$$h[n] = \frac{1}{2\pi}\int_{2\pi} H(\Omega)e^{j\Omega n}d\Omega$$

Eq. 5.5 impulse response filter

If we start with a desired frequency response $H(\Omega)$, Eq. 5.5 shows how to derive the corresponding impulse response *h[n]*. The sample values of *h[n]* give the required multiplier coefficients b_k for out nonrecursive filter.

The approach is conceptually straightforward, but there are two main practical difficulties:

1. The integral in Eq. 5.5 can be hard to solve, therefore we will assume linear phase responses and use a computer to estimate the values of *h[n]*
2. The choice of $H(\Omega)$ may result in an impulse response with a large number of terms, giving a very uneconomic filter, so a comprise has to be settled between time-domain and frequency-domain performance

See Example 5.1 in the book: an infinitely sharp cut-off filter is designed, giving an infinite time-domain response.

6 Recursive Digital Filters

6.1 General design information

A recursive filter depends on one or more previous outputs and on inputs: feedback. It has great computational advantages: in contrast to a nonrecursive filter a recursive filter requires just a few coefficients. This approach has also two distinct disadvantages:

feedback

- a recursive filter may become unstable if the coefficients are chosen badly
- a recursive design cannot provide a linear-phase response

A recursive filter usually has a infinite impulse response (IIR). This means the impulse response *h[n]* decays to zero as $n \to \infty$. hence theoretically it continues forever.

infinite impulse response
IIR

Recursive filter design not only focuses on z-plane zeros, but also on z-plane poles:

$$\boldsymbol{H}(z) = \frac{K(z-z_1)(z-z_2)(z-z_3)\dots}{(z-p_1)(z-p_2)(z-p_3)\dots}$$ **Eq. 6.1**

Hence giving a frequency response as:

$$\boldsymbol{H}(\Omega) = \frac{K(e^{j\Omega}-z_1)(e^{j\Omega}-z_2)(e^{j\Omega}-z_3)\dots}{(e^{j\Omega}-p_1)(e^{j\Omega}-p_2)(e^{j\Omega}-p_3)\dots}$$ **Eq. 6.2**

which can be readily computed. Eq. 6.2 gives separate control over numerator and denominator.

A fast way of designing recursive filters is by making calculated guesses for the value of the coefficients: a pole close to the unit circle gives a response peak; a zero close to or on the unit circle produces a trough (or null). See figure 6.1 (page 170) and figure 6.2 (page 171) for some examples.

Usually only the magnitude of $\boldsymbol{H}(\Omega)$ is of interest. It is therefore convenient to produce a decibel plot of $|\boldsymbol{H}(\Omega)|$ against frequency. Suppose we specify a real pole at $z = \alpha$. It contributes the following factor to the denominator of $\boldsymbol{H}(\Omega)$:

$$\boldsymbol{F}(\Omega) = e^{j\Omega} - \alpha = \cos\Omega - \alpha + j\sin\Omega$$ **Eq. 6.3**

with magnitude:

$$|\boldsymbol{F}(\Omega)| = \sqrt{1 - 2\alpha\cos\Omega + \alpha^2}$$ **Eq. 6.4**

A real zero at $z = \alpha$ would give an identical contribution to the numerator of $|\boldsymbol{H}(\Omega)|$. An *n*th-order pole or zero would give a contribution $|\boldsymbol{F}(\Omega)|^n$.

6.2 Digital integrators

Integration, like differentiation, is a common signal processing operation and can be considered as a special kind of recursive filter. Several different approaches to calculate the integration of signal exist, three of them are:

- **running sum**: this process simply adds up the samples of a digital ($y[n] = y[n-1] + x[n]$) and has the following transfer function and frequency response:

running sum

$$\boldsymbol{H}(z) = \frac{\boldsymbol{Y}(z)}{\boldsymbol{X}(z)} = \frac{1}{1-z^{-1}} = \frac{z}{z-1}$$ **Eq. 6.5**
running sum transfer function

$$\boldsymbol{H}(\Omega) = \frac{1}{1-e^{-j\Omega}} = \frac{e^{j\Omega}}{e^{i\Omega}-1}$$ **Eq. 6.6**
running sum frequency response

- **trapezoid rule**: the output is updated by the average of the two adjacent inputs ($y[n] = y[n-1] + \frac{x[n] + x[n-1]}{2}$)

trapezoid rule

$$\mathbf{H}(z) = \frac{z+1}{2(z-1)}$$

Eq. 6.7
trapezoid transfer function

$$\mathbf{H}(\Omega) = \frac{e^{j\Omega}+1}{2(e^{j\Omega}-1)}$$

Eq. 6.8
trapezoid frequency response

- **Simpson's rule**: $y[n] = y[n-2] + \frac{x[n] + 4x[n-1] + x[n-2]}{3}$

Simpson's rule

$$\mathbf{H}(z) = \frac{z^2+4z+1}{3(z^2-1)}$$

Eq. 6.9
Simpson's transfer function

$$\mathbf{H}(\Omega) = \frac{e^{2j\Omega}+4e^{j\Omega}+1}{3(e^{2j\Omega}-1)}$$

Eq. 6.10
Simpson's frequency response

7 Discrete and Fast Fourier Transforms

The Discrete Fourier Transform (DFT) is one of the most important signal-operations used in DSP. When implemented using a Fast Fourier Transform (FFT) algorithm the DFT offers a rapid frequency-domain analysis and processing of digital signals and investigations of LTI systems. This chapter will explain the fundaments of the DFT, its computational problems and how these are solved by implementing a FFT.

discrete Fourier transform
DFT
fast Fourier transform
FFT

7.1 The discrete Fourier transform

The DFT of an aperiodic signal *x[n]*, with a finite number of non-zero samples, defined over a range $0 \le n \le N-1$ is given by:

$$\mathcal{X}[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi kn}{N}} = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

Eq. 7.1
Discrete Fourier Transform

where $W_N = e^{\frac{-j2\pi}{N}}$, and the spectral coefficients $\mathcal{X}[k]$ are evaluated for $0 \le k \le N-1$. To recover the original signal the inverse DFT (IDFT) is used, which is given by:

inverse DFT
IDFT

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} \mathcal{X}[k] e^{\frac{j2\pi kn}{N}} = \frac{1}{N} \sum_{k=0}^{N-1} \mathcal{X}[k] W_N^{-kn}$$

Eq. 7.2
Inverse DFT

Eq. 7.1 has no limitations on *k*. If we use this equation to calculate $\mathcal{X}[k]$ for value of *k* outside the range $0 \le k \le N-1$, we find that $\mathcal{X}[k]$ is periodic. The same phenomenon occurs when the IDFT is used to recalculate *x[n]* from $\mathcal{X}[k]$. Additional values of *x[n]* outside $0 \le n \le N-1$ yields a periodic version of *x[n]*. This means the DFT and IDFT consider the signal to be periodic.

The number of samples required for a reliable DFT-computation needs a theoretical foundation. The frequency-domain version of the Sampling theorem provides a good base to start from. Because of the symmetry of the Fourier transformation the frequency-equivalent of this theorem is stated as follows:

> *The continuous spectrum of a signal with limited duration T_0 seconds may be completely represented by regularly-spaced frequency-domain samples, provided the samples are spaced not more than $T_0^{-1} Hz$ apart.*

Theorem 3
Fourier-Sampling theorem

With *N* finite samples and with *T* as the sampling interval the spectrum can be represented with samples spaced $(NT)^{-1} Hz$ or $2\pi(NT)^{-1}$ radians per second apart. Hence with $\Omega = \omega T$ the samples in Ω must be taking at intervals of $2\pi N^{-1}$ or less.

7.1.1 Properties of the DFT

periodicity

- **periodicity**: with ranges $0 \le n \le N-1$ and $0 \le k \le N-1$ for respectively *x[n]* and $\mathcal{X}[k]$

  $x[n] = x[n+N]$ and $\mathcal{X}[k] = \mathcal{X}[k+N]$

- **linearity:**
  if $x_1[n] \leftrightarrow X_1[k]$ and $x_2[n] \leftrightarrow X_2[k]$
  then $Ax_1[n] + Bx_2[n] \leftrightarrow AX_1[k] + BX_2[k]$

linearity

- **time-shifting:**
  if $x[n] \leftrightarrow X[k]$
  then $x[n - n_0] \leftrightarrow X[k]e^{-\frac{j2\pi k n_0}{N}} = X[k]W_N^{kn_0}$

time-shifting

- **convolution:**
  if $x_1[n] \leftrightarrow X_1[k]$ and $x_2[n] \leftrightarrow X_2[k]$
  then $x_1[n] \otimes x_2[n] \leftrightarrow X_1[k]X_2[k]$

convolution

- **modulation:**
  if $x_1[n] \leftrightarrow X_1[k]$ and $x_2[n] \leftrightarrow X_2[k]$
  then $x_1[n]x_2[n] \leftrightarrow X_1[k] \otimes X_2[k]$

modulation

- **symmetry table:**

symmetry table

x[n]		X[k]	
$\Re$	$\Im$	$\Re$	$\Im$
even	0	even	0
odd	0	0	odd
0	even	0	even
0	odd	odd	0

7.1.2 Computing the DFT

A major practical consideration when computing the DFT is its speed. It may seem obvious to use Eq. 7.1 and Eq. 7.2, but these are relatively slow to compute on normal computers. By expressing the equations in terms of sines and cosines we have:

$$X[k] = \sum_{n=0}^{N-1} x[n]\left(\cos\left(\frac{2\pi kn}{N}\right) - j\sin\left(\frac{2xkn}{N}\right)\right)$$

Eq. 7.3
DFT using sin and cos

$$x[n] = \frac{1}{N}\sum_{m=0}^{N-1} X[k]\left(\cos\left(\frac{2\pi kn}{N}\right) + j\sin\left(\frac{2xkn}{N}\right)\right)$$

Eq. 7.4
IDFT using sin and cos

Except for the scaling factor and the sign, the calculations are the same. By making use of symmetry properties, calculations can be speed up even more. But even then the number of floating point calculations range from $4N^2$ (worst case) to N^2 (even/odd real signal). It is also a very inefficient algorithm: many calculations are done many times, thus consuming computer time with redundant work. FFT-algorithms try to minimise the amount of redundancy, thus speeding up the process.

redundancy

7.2 The Fast Fourier Transform (FFT)

Due to the periodic nature of W_N^{kn} and the limited number of distinct values it has, much of the redundancy can be eliminated. Take for example a short DFT ($N = 8$). Both k and n vary between 0 and 7 inclusive so the product kn varies between 0 and 49. However, there are only eight distinct values for W_8^{kn} (see Figure 6). The table forms a matrix with diagonal symmetry and many of the values are ± 1 implying simple additions and substractions. All these feature contribute to the redundancy of the standard DFT.

Figure 6
Tabulation of W_8^{kn}

		Value of n							
		0	1	2	3	4	5	6	7
	0	1	1	1	1	1	1	1	1
	1	1	$\frac{(1-j)}{\sqrt{2}}$	$-j$	$-\frac{(1+j)}{\sqrt{2}}$	-1	$-\frac{(1-j)}{\sqrt{2}}$	j	$\frac{(1+j)}{\sqrt{2}}$
value of k	2	1	$-j$	-1	j	1	$-j$	-1	j
	3	1	$-\frac{(1+j)}{\sqrt{2}}$	j	$\frac{(1-j)}{\sqrt{2}}$	-1	$\frac{(1+j)}{\sqrt{2}}$	$-j$	$-\frac{(1-j)}{\sqrt{2}}$
	4	1	-1	1	-1	1	-1	1	-1
	5	1	$-\frac{(1-j)}{\sqrt{2}}$	$-j$	$\frac{(1+j)}{\sqrt{2}}$	-1	$\frac{(1-j)}{\sqrt{2}}$	j	$-\frac{(1+j)}{\sqrt{2}}$
	6	1	j	-1	$-j$	1	j	-1	$-j$
	7	1	$\frac{(1+j)}{\sqrt{2}}$	j	$-\frac{(1-j)}{\sqrt{2}}$	-1	$-\frac{(1+j)}{\sqrt{2}}$	$-j$	$\frac{(1-j)}{\sqrt{2}}$

7.2.3 Decimation-in-time

conventional decomposition

In this section we will show how the DFT can be expressed in terms of shorter, simpler, DFT's by dividing the signal into subsequences.The method is known as conventional decomposition.

decimation-in-time

The conventional decomposition which will be explained in this section is called decimation-in-time. Suppose we have a signal with N sample values, where N is an integer power of 2. We first separate x[n] into two subsequences, each with $N/2$ samples. The first subsequence consists of the even-numbered points in *x[n]*, and the second consists of the odd-numbered points. Writing $n = 2r$ when n is even, and $n = 2r + 1$ when n is odd, the DFT may be recast as:

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{kn}, \qquad 0 \le k \le N-1 \\ &= \sum_{r=0}^{\frac{N}{2}-1} x[2r]W_N^{2rk} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]W_N^{(2r+1)k} \\ &= \sum_{r=0}^{\frac{N}{2}-1} x[2r](W_N^2)^{rk} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1](W_N^2)^{rk} \end{aligned} \qquad \text{Eq. 7.5}$$

We note that:

$$W_N^2 = e^{-\frac{2j2\pi}{N}} = W_{N/2} \qquad \text{Eq. 7.6}$$

and we may therefore write:

$$\begin{aligned} \mathcal{X}[k] &= \sum_{r=0}^{\frac{N}{2}-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]W_{N/2}^{rk} \\ &= \mathcal{G}[k] + W_N^k \mathcal{H}[k] \end{aligned} \qquad \text{Eq. 7.7}$$

We have now expressed the original *N*-point DFT in terms of two *N/2*-point DFTs, $\mathcal{G}[k]$ and $\mathcal{H}[k]$. $\mathcal{G}[k]$ is the transform of the even-numbered points in *x[n]*, and $\mathcal{H}[k]$ is the transform of the odd-numbered points. Note that we must multiply $\mathcal{H}[k]$ by an additional term W_N^k before adding it to $\mathcal{G}[k]$.

radix-2

If we assume that the transform length *N* is an integer power of 2, it follows that *N/2* is even. Therefore we can take the decomposition further, by breaking each *N/2*-point susequence down into two shorter, *N/4*-point subsequences. The process can continue until, in the limit, we are left with a series of 2-point subsequences, each of which requires a very simple 2-point DFT. A complete decomposition of this type gives rise to one of the commonly-used radix-2, decimation-in-time, FFT algorithms.

8 FFT Processing

In this chapter we will focus on two of the main application areas of FFT processing:

1. **Digital spectral analyse**. The use of the FFT to explore the properties of digital signals and systems
2. **Digital filtering by fast convolution**. This application offers a frequency-domain alternative to the time-domain filtering techniques discussed in chapter 5 and chapter 6

In chapter 5 and chapter 6 the time-domain methods of signal filtering are descibed. An alternative approach, based on the FFT, can be summarized as follows. First, the input signal is transformed into the frequency-domain. The various spectral coefficients are then modified in magnitude and phase according to the desired filter characteristics. This means that filtering is accomplished by frequency-domain multiplication rather than time-domain convolution. The approach is very flexible and may in principle be used to implement any desired frequency response specification.

The effectiveness of this method relies on the speed of the FFT algorithms. We should be careful not to assume that it is necessarily superior to time-domain filtering. Its flexibility is not always required. Futhermore a time-domain filter with a few recursive coefficients may well be faster an FFT implementation. In practice the choice between them depends not only on speed, but also on the amount of storage needed, and whether or not true on-line working is required. If a filtered output sample is required to be generated every time a new input sample becomes available, then a time-domain filter has to be specified, because effective use of the FFT demands that data is stored and processed in substantial blocks.

8.1 Spectral analysis

digital spectral analysis

Digital spectral analysis is, unlike digital filtering, investigative: it is not necessarily concerned with modifying the signal. Spectral analysis is used in two different sets of applications:

1. **Naturally-occuring signals**. Examples arise in the analysis of speech, biomedic signals
2. **Complete systems**. A system is disturbed with a suitable input and the output is analysed. Examples arise in the analysis of electronic circuits and filters, vibrations in buildings, radar, sonar

spectral function

An FFT algorithm yields a set of spectral coeffients which may be regarded as samples of the underlying continous spectral function (or as harmonics of a periodic version of the same signal). Just as a digital signal *x[n]* is sampled in the time domain, so its DFT is sampled in the frequency domain.

spectral spreading
spectral leakage

If all frequency components of a signal have a integral number of periods within the transform length no discontinuities arise with end-to-end repetion. Each component then occupies a definite harmonic frequency, corresponding to a single spectral coefficient. If this is not the case, the discontinuities in the time domain will lead to spectral spreading or leakage in the frequency domain. This effects is of much significance! (see Figure 7)

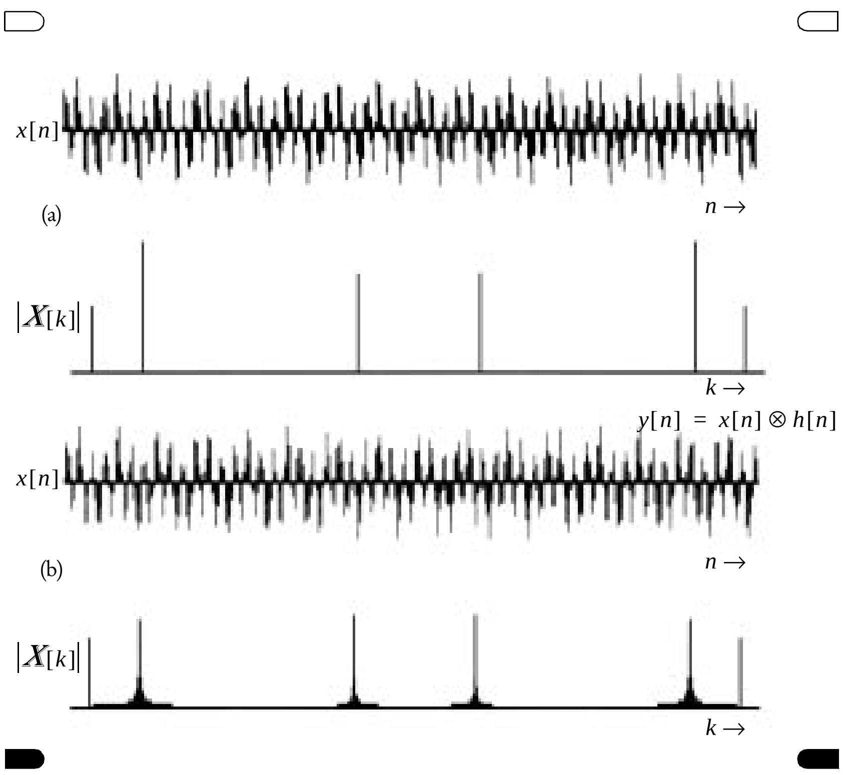


Figure 7
Fourier transformation of (a) a signal containing three exact Fourier harmonics and (b) a signal containing both harmonic and non-harmonic components.

In Figure 7, the following two signals are plotted:

$$x[n] = 0.1\sin\left(\frac{2\pi n}{512}16\right) + 0.2\sin\left(\frac{2\pi n}{512}53\right) + 0.15\cos\left(\frac{2\pi n}{512}211\right)$$

Eq. 8.1
Signal in Figure 7 (a)

$$x[n] = 0.1\sin\left(\frac{2\pi n}{512}16\right) + 0.2\sin\left(\frac{2\pi n}{512}53.5\right) + 0.15\cos\left(\frac{2\pi n}{512}211.25\right)$$

Eq. 8.2
Signal in Figure 7 (b)

The smear around the peaks is called spectral leakage. The two waveforms that don't fit within the frame can not be described by two distinct harmonics and the DFT-algorithm tries to estimate the two waveforms by adding several harmonics with different amplitudes.

frequency resolution

The longer the signal we transform, the greater the value of N and therefore the longer the base (see chapter 3) for the signal. The frequency resolution is defined as: $\Delta f = \frac{2\pi}{N}$ or as: $\frac{fs}{N}$.

8.1.1 Windowing

windowing
tapering

To reduce the effect of spectral leakage, a window is applied to the signal. The common function of such a window is to taper the edges of the signal, thus reducing the size of the discontinuities. See Figure 8.7 in the book for some examples.

A window changes the signal and thus its spectrum. To minimize the overal effect of a window for example only the first and last 15 % of the signal is tapered. This can only be done if the signal is lengthy.

Spectral analysis and the choice of window to be used on the signal is not an easy task. A wrong window can hamper your data and thus your conclusions but usually no proper conclusions can be made without the use of some sort of window.

8.2 Digital filtering by fast convolution

fast convolution

A convolution in the time domain is equivalent to multiplication in the frequency domain. Thus instead of filtering in the time domain using convolution a much faster result is archieved when filtering in the frequency domain, using multiplication. The complete process is called fast convolution and is depicted in Figure 8.

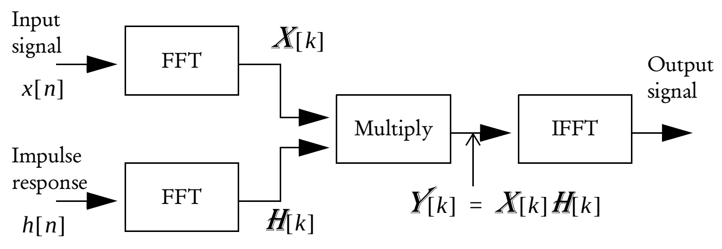


Figure 8
Fast convolution.

zero-filling

It is important to see that a linear convolution is needed here, and not a circular one. That is, the convolution of an aperiodic input signal and impulse response produces an aperiodic output. Fortunately, it turns out one period of the output signal derived by circular convolution gives the correct result, providing the lengths of the transforms are extended using an appropriate amount of zero-filling. See Figure 8.11 and 8.12 in the book for an example.

9 Random Digital Signals

In this and the next chapter we turn our attention to the description and processing of random signals. The signals thusfar are generally referred to as deterministic, e.g. they have defined, known values at each sampling instant. The individual values of random sigals are not defined, nor can we predict their future values with certainty.

9.1 Amplitude distribution

Usually a probability density function is used to descibe a continuous random variable. With DSP this is inappropriate, given the fact that a digital signal can only take a finite number of different amplitudes, limited by the wordlength of the processing unit. Instead a probability mass function p_{x_n} is defined:

$$p_{x_n} = probability\{x[n] = x_n\}$$

Eq. 9.1
Probability mass function

Thus p_{x_n} simply tells us the probability associated with each of the allowed values of the sequence. An associated function, the probability distribution function, is:

$$P_{x_n} = probability\{x[n] \le x_n\} = \sum_{-\infty}^{x_n} p_{x_n}$$

Eq. 9.2
Probability distribution function

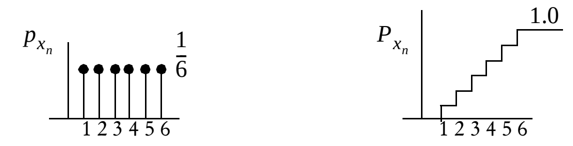


Figure 9
Probability functions for a die-tossing experiment

The probability function in Figure 9 is uniform or even. Another type of distribution which is very common in DSP is called the Gaussian distribution. Because of its large number of applications in different fields it is also known as the normal distribution.

uniform distribution
Gaussian distribution

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}$$

Eq. 9.3
Gaussian probability density

As far as DSP is concerned, any random sequence which is produced by superposition of many contributing processes is likely to have a gaussian form of amplitude distribution. For example, noise arising in digital communication and computer systems is often of this type.

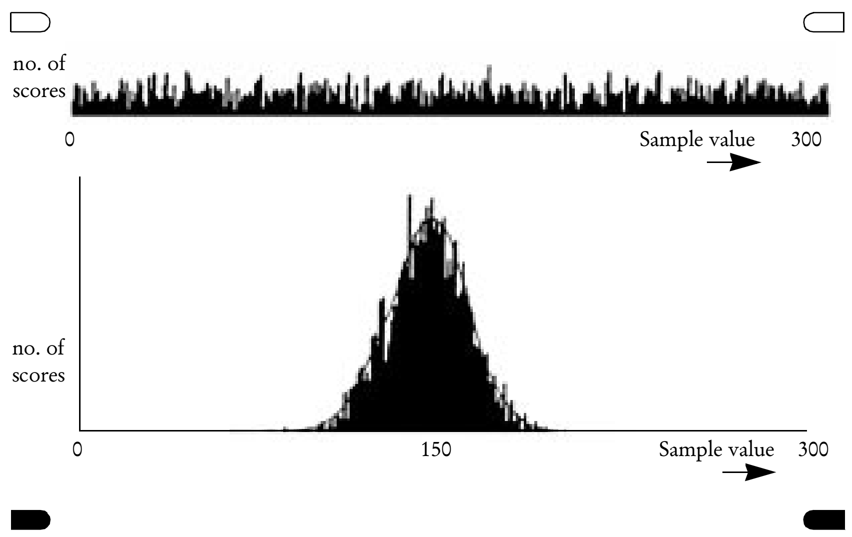


Figure 10
Estimates of the probability mass function for an uniform distribution and a gaussian distribution.

9.2 Mean, mean-square and variance

The use of a probability mass function is in most practical applications somewhat overdone. It is usually enough to get the mean value and the average size of fluctuations about the mean of the signal.

The mean of a random digital sequence is defined as:

mean

$$m = E\{x[n]\} = \sum_{-\infty}^{\infty} x_n p_{x_n}$$

Eq. 9.4
Mean

where E denotes mathematical expectation. It is the same as the average value of the sequence and in electronic engineering is also widely referred to as the d.c. value.

mathematical expectation
d.c. value

The mean of a signal has some useful properties:

- If every value of a signal is multiplied by a constant, the mean is also multiplied by the same constant
- If two or more random signals are added together, then the mean of their sum equals the sum of their means

The mean-square, or average power, is defined as follows:

mean-square
average power

$$\overline{m^2} = E\{x[n]^2\} = \sum_{-\infty}^{\infty} x_n^2 p_{x_n}$$

Eq. 9.5
Mean-square

The variance refers to fluctuations about the mean and is given the symbol σ^2:

variance

$$\sigma^2 = E\{(x[n]-m)^2\} = \sum_{-\infty}^{\infty} (x_n - m)^2 p_{x_n}$$

Eq. 9.6
Variance

Or: variance = mean-square - (mean)^2. The variance is also refered to as the fluctuation power or a.c. power.

fluctuation power
a.c. power

9.3 Ensemble averages, time averages

Formally, a random or stochastic digital signal is one giving rise to an infinite set of random variables, of which a particular sample sequence *x[n]* is one realisation.The set of all sequences which could be generated by the process is known as an infinite ensemble, and it is this to which the probability functions and expected values defined by Eq. 9.4, Eq. 9.5 and Eq. 9.6 strictly refer. Such measures are therefore known as ensemble averages.

infinite ensemble
ensemble average

time average

In practise, we generally deal with single sequences. Each sample represens a single value of one of the variables described by the underlying stochastic process. We must therefore make a connection between the properties of an ensemble and those of an indivudual sequence existing over a period of time. This connection can be found in terms of time averages (only if the signal is stationary):

$$m_x = \langle x[n] \rangle = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} x[n]$$

Eq. 9.7
Time-averaged mean

$$\sigma^2 = \langle (x[n] - m_x)^2 \rangle = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} (x[n] - m_x)^2$$

Eq. 9.8
Time variance

9.4 Autocorrelation

autocorrelation

Autocorrelation ('in relation to itself') is one of the most important techniques for exploring properties of a random digital signal. It is essentially a time-domain measure and we shall see later that it is related to the spectral properties of a sequence in the frequency domain.

autocovariance

Knowledge of the sequence's recent history could help us predict the next value. In other words the sequence may have a time-domain structure, and there are many practical situations in which such a structure must be characterized. The most widely used measures for this purpose are the autocorrelation function and the autocovariance function:

$$\phi_{xx}[m] = \langle x[n] \cdot x[n+m] \rangle$$
$$= \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} x[n] \cdot x[n+m]$$

Eq. 9.9
Autocorrelation

$$\gamma_{xx}[m] = (x[n] - m_x)(x[n+m] - m_x)$$
$$= \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} (x[n] - m_x)(x[n+m] - m_x)$$

Eq. 9.10
Autocovariance

And the relation between autocorrelation and autocovariance is given as:

$$\gamma_{xx}[m] = \phi_{xx}[m] - m_x^2$$

Eq. 9.11

Because the defintion of mean, variance, autocorrelation and autocovariance are based of infinitely long time sequences, and in practise we only have finite sequence length, these quantities have to be estimated. The estimates will always possess statistical variability, and will only tend towards the true functions as the sequence length tend toward infinity. Interpreting these results most therefore be done with a little caution.

Example 9.4 in the book illustrates a number of important points and Figure 11 gives five sequences and their estimated autocorrelation functions.

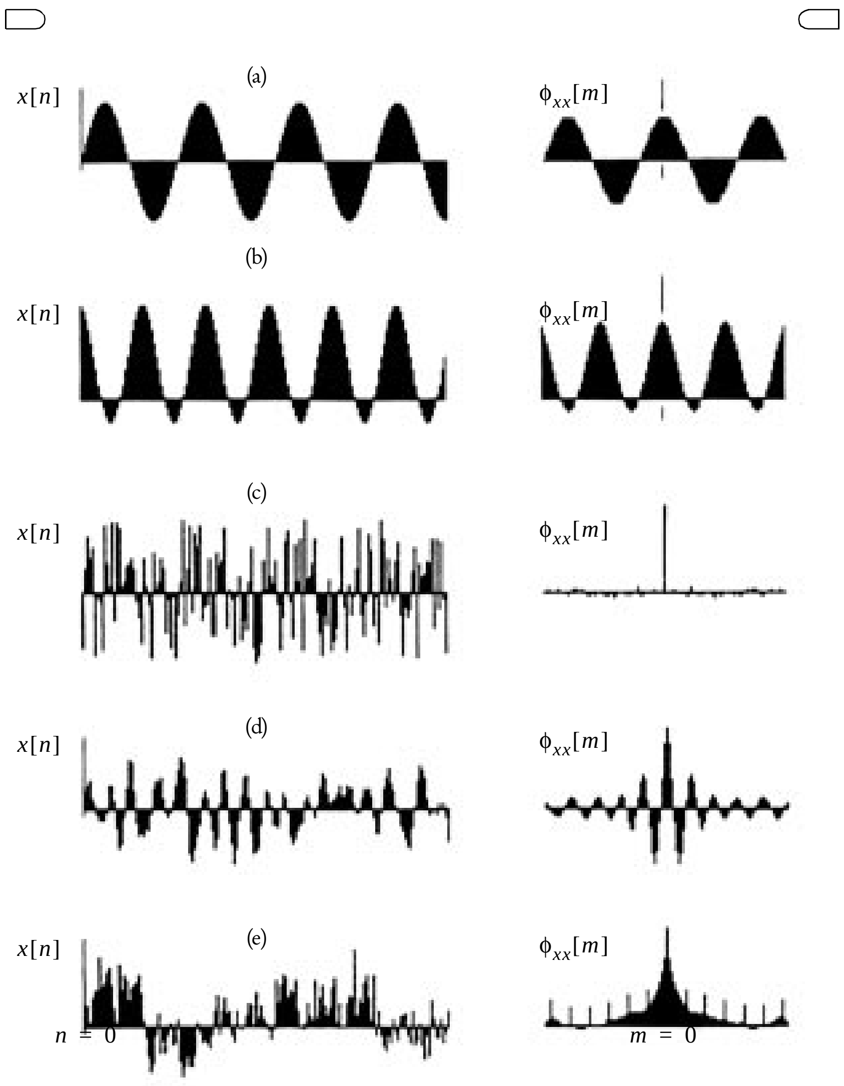


Figure 11
Five digital sequences and their estimated autocorrelation functions. The central value of the autocorrelation is always the maximum value, but parts (a) and (b) illustrate that it need not be the only one with this value. Part (c) is a uniformly-distributed completely random sequence (white noise). It has a maximum at zero and (nearly) zero values at either side. Part (d) is the sequence of part (c) after a bandpass filter. Notice the immense impact of the filter on the sequence in the autocorrelation function! Part (e) is also white noise, only now after a low-pass filter. After the filter a repetitive impuls train is added to the signal. This impulse train cannot be seen directly in the sequence, but comes out nicely in the autocorrelation function.

The number of cross-products decreases as the shifted sequence 'slides over' the unshifted one. This implies that estimates in the tails of the computed autocorrelation function are somewhat less reliable than those in the center. For this reason, it is usual to restrict shift values to no more than about 15-20% of the available sequence length.

9.5 Power spectrum

power spectral density
power spectrum

The frequency-domain counterpart of the autocorrelation function is called the power spectral density, or more simply the power spectrum:

$$P_{xx}(\Omega) = \sum_{m=-\infty}^{\infty} \phi_{xx}[m]e^{-j\Omega m}$$

Eq. 9.12
Power spectrum

The inverse will (of course) result in the autocorrelation function:

$$\phi_{xx}[m] = \frac{1}{2\pi}\int_{2\pi} P_{xx}(\Omega)e^{j\Omega m}d\Omega$$

Eq. 9.13
Inverse power spectrum

Note that by putting $m = 0$ we obtain the total power of the sequence:

$$\phi_{xx}[0] = \frac{1}{2\pi}\int_{2\pi} P_{xx}(\Omega)d\Omega$$

Eq. 9.14
Total power

9.6 Cross-correlation

We have seen how autocorrelation may be used to characterize a sequence's time-domain structure, revealing statistical relationships between successive sample values. Cross-correlation is an essentially similar process, but instead of comparing a sequence with a shifted version of itself, it compares two different sequences. It gives us statistical information about the time-domain relationships between them.

cross-correlation

The cross-correlation function and the cross-covariance function of two sequences *x[n]* and *y[n]* may be defined in terms of time averages:

cross-covariance

$$\phi_{xy}[m] = \langle x[n]y[n+m] \rangle = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} x[n]y[n+m]$$

Eq. 9.15 Cross-correlation

$$\gamma_{xy}[m] = \langle (x[n]-m_x)(y[n+m]-m_y) \rangle = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} (x[n]-m_x)(y[n+m]-m_y)$$

Eq. 9.16 Cross-covariance

Just as the form of an autocorrelation function reflects the various frequency components in the underlying sequence, so the form of a cross-correlation function reflects components held in common between *x[n]* and *y[n]*.

9.7 Cross-spectrum

The frequency-domain counterpart of a cross-correlation function is known as a cross-spectral density or cross-spectrum. It gives valuable information about frequencies held in common between *x[n]* and *y[n]*.

cross-spectral density
cross-spectrum

If two sequences *x[n]* and *y[n]* have no shared frequencies or frequency ranges, their cross-spectrum is zero. They are said to be linearly independent.

$$\boldsymbol{P}_{xy}(\Omega) = \sum_{m=-\infty}^{\infty} \phi_{xy}[m] e^{-j\Omega n}$$

Eq. 9.17 Cross-spectrum

$$\phi_{xy}[m] = \frac{1}{2\pi} \int_{2\pi} \boldsymbol{P}_{xy}(\Omega) e^{j\Omega m} d\Omega$$

Eq. 9.18 Inverse cross-spectrum

10 Random DSP

DSP is concerned primarily with processing signals, rather than merely describing them. With processing signals and system responses, questions may arise. How are the statistical measures of random signals and noise which we developed in the previous chapter affected by processing? If we know such measures at the input of a given filter or processor, can we predict what they will be at the output? How do we design an optimum processor for enhancing a signal contaminated by noise?

The effects of processing can be readily described and predicted, as will be shown in this chapter. You may be confident that a clear understanding of this chapter will stand you in good stead for further reading of the DSP literature and for tackling more advanced topics in random signal processing.

10.1 Response of linear processors

In chapter 2 it was shown that the output of a system is the convolution of the input and the impulse response of the system (Eq. 2.3). The difference is, that the input signal is now random.

The convolution (Eq. 2.3) deals with individual sample values, and is therefore not of direct use to us as it stands. However, the equation can be modified to handle time-averaged statistics. Thus the mean value of the output sequence may be derived as follows:

$$\begin{aligned} m_y &= E\{y[n]\} \\ &= \sum_{k=-\infty}^{\infty} E\{x[n-k]\}h[k] \\ &= m_x \sum_{k=-\infty}^{\infty} h[k] \end{aligned}$$

Eq. 10.1
Output mean

This result uses the fact that the expectation of a sum equals the sum of expectations. It shows that the mean of the output is equal to the mean of the input multiplied by the sum of all impulse response terms. In other words, the d.c. value of the output must equal that of the the input multiplied by the zero-frequency response of the processor:

$$m_y = m_x \boldsymbol{H}(0)$$

Eq. 10.2
Output mean in freq. dom.

Before the effects on variance are descibed, first the autocorrelation functions of the input and output are given (again using Eq. 2.3):

$$\begin{aligned} \phi_{yy[m]} &= E\{y[n]y[n+m]\} \\ &= E\left\{\sum_{k=-\infty}^{\infty} x[n-k]h[k] \sum_{r=-\infty}^{\infty} x[n+m-r]h[r]\right\} \\ &= \sum_{k=-\infty}^{\infty} h[k] \sum_{r=-\infty}^{\infty} E\{x[n-k]x[n+m-r]\}h[r] \end{aligned}$$

Eq. 10.3
Autocorrelation with time av.

The expectation of this last expression equals the value of the input autocorrelation relevant to the time shift $(m+k-r)$. Hence:

$$\phi_{yy[m]} = \sum_{k=-\infty}^{\infty} h[k] \sum_{r=-\infty}^{\infty} \phi_{xx}[m+k-r]h[r] \qquad \textbf{Eq. 10.4}$$

It is now helpful to substitute q for $(r-k)$, giving:

$$\phi_{yy[m]} = \sum_{k=-\infty}^{\infty} h[k] \sum_{q=-\infty}^{\infty} \phi_{xx}[m-q]h[r]$$
$$= \sum_{q=-\infty}^{\infty} \phi_{xx}[m-q] \sum_{k=-\infty}^{\infty} h[k]h[k+q] \qquad \textbf{Eq. 10.5}$$

If we now write:

$$j[q] = \sum_{k=-\infty}^{\infty} h[k]h[k+q] \qquad \textbf{Eq. 10.6}$$

Autocorrelation of impulse response

we obtain:

$$\phi_{yy}[m] = \sum_{q=-\infty}^{\infty} \phi_{xx}[m-q]j[q] \qquad \textbf{Eq. 10.7}$$

The result can be summarized as follows:

The autocorrelation function of the output sequence of a linear processor may be found by convolving the autocorrelation function of its input sequence with a function representing the autocorrelation function of the processors's impulse response.

Eq. 10.7 provides a key to the relationship between the output variance and input variance. If we put $m = 0$ we obtain the peak central value of the output autocorrelation function. The autocorrelation function is always even, thus creating:

$$\phi_{yy}[0] = \sum_{m=-\infty}^{\infty} \phi_{xx}[m]j[m] \qquad \textbf{Eq. 10.8}$$

Using Eq. 9.11, Eq. 10.1 and Eq. 10.8 gives:

$$\sigma_y^2 = \left(\sum_{m=-\infty}^{\infty} \phi_{xx}[m]j[m]\right) - \left(m_x \sum_{n=-\infty}^{\infty} h[n]\right)^2 \qquad \textbf{Eq. 10.9}$$

Output variance

The basic notion with the output power spectrum is that the frequency response $H(\Omega)$ of a processor defines its effect, in amplitude and phase, on any input frequency Ω. Since power is propertional to amplitude-squared and involves no phase information, the equivalent power response of the processor is $|H(\Omega)|^2$. Hence:

$$P_{yy}(\Omega) = |H(\Omega)|^2 P_{xx}(\Omega) \qquad \textbf{Eq. 10.10}$$

Output power spectrum

10.2 White noise through a filter

White noise creates from a communications theory point of view a paradox: because it is completely unpredictable in advance it contains the maximum amount of information. If it is decoded the maximum amount of information is retreived. Thus if a white sequences represents unwanted noise it is the most chaotic possible, but if it represents a signal if carries the greatest possible amount of information.

To make the distinction between noise and a signal we need to be able to predict the effects of linear prosessing upon the sequence. Measures as the mean, variance, autocorrelation function and power spectrum are likely to be of most interest.

A white input sequence has an autocorrelation function consisting of a single, isolated, unit impulse at $m = 0$. Thus the statistical properties of the input are:

$$\begin{aligned} m_x &= 0 \\ \sigma_x^2 &= 1 \\ \phi_{xx}[m] &= \delta[m] \\ \boldsymbol{P}_{xx}(\Omega) &= 1 \end{aligned}$$

Eq. 10.11

If we pass such a sequence through an LTI filter, the mean value m_y of the output will also be zero. The output autocorrelation function reduces to:

$$\phi_{yy[m]} = \sum_{q=-\infty}^{\infty} \phi_{xx}[m-q]j[q] = \sum_{q=-\infty}^{\infty} \delta[m-q]j[q] = j[m]$$

Eq. 10.12

The output variance is (using Eq. 9.11):

$$\sigma_y^2 = \phi_{yy}[0] - m_y^2 = \phi_{yy}[0] = j[0]$$

Eq. 10.13

When applying a white noise sequence to a LTI processor, any correlation in the time domain or frequency domain must be due to the processor, hence the statistics of the output can be summarized as follows:

$$\begin{aligned} m_y &= 0 \\ \sigma_y^2 &= j[0] \\ \phi_{yy}[m] &= j[m] \\ \boldsymbol{P}_{yy}(\Omega) &= |\boldsymbol{H}(\Omega)|^2 \end{aligned}$$

Eq. 10.14
White noise output statistics

10.3 System identification by cross-correlation

The cross-correlation function can be used to indicate the frequencies, or frequency ranges, held in common between two random sequences *x[n]* and *y[n]*. The cross-correlation function also contains information about a processor if the two sequences are an input-output-pair. Moreover, the statistical comparison of random input and output signals can yield useful information about a deterministic system or processor.

When feeding a random (usually white noise) signal to an unknown system, any non-whiteness in the output signal must be due to the frequency-selective properties of the system itself. Viewed in this way, we may anticipate that a white-noise input will give a particular simple and attractive relationship between the input-output cross-correlation function and the properties of the system under investigation.

We can develop this important idea quantitatively by recalling that the cross-correlation function of two sequences equals their average cross-product, thus we may write:

$$\phi_{xy}[m] = E\{x[n]y[n+m]\}$$

Eq. 10.15

Using system convolution:

$$\begin{aligned} \phi_{xy}[m] &= E\left\{x[n] \sum_{k=-\infty}^{\infty} x[n+m-k]h[k]\right\} \\ &= \sum_{k=-\infty}^{\infty} \phi_{xx}[m-k]h[k] \end{aligned}$$

Eq. 10.16

This important result shows that the input-output cross-correlation function equals the convolution of the input autocorrelation function with the impulse response of the system.

Assume a zero-mean, white noise input of unit variance. Then the input autocorrelation function consists of an unit impulse at $m = 0$, hence:

$$\phi_{xy}[m] = \sum_{k=-\infty}^{\infty} \delta[m-k]h[k] = h[m] \qquad \text{Eq. 10.17}$$

Given a white noise input, the cross-correlation function between input and output becomes identical to the system's impulse response. And the impulse response of any LTI system completely defines it in the time domain.

Because of the convolution relationship of the Fourier transformation, the cross-spectrum is defined as follows:

$$\boldsymbol{P}_{xy}(\Omega) = \boldsymbol{P}_{xx}(\Omega)\boldsymbol{H}(\Omega) \qquad \text{Eq. 10.18}$$

In the case of the white noise input with zero mean and unit variance, the input power spectrum is unity, therefore:

$$\boldsymbol{P}_{xy}(\Omega) = \boldsymbol{H}(\Omega) \qquad \text{Eq. 10.19}$$

In this case the cross-spectrum takes the same form as the frequency response of the system.

10.4 Signals in noise

10.4.1 Signal recovery

One of the most common requirements in signal recovery is to separate a comparative narrowband signal from wideband noise.

In such situations we are interested in predicting the improvement in signal-to-noise ratio which can be obtained by linear filtering. As far as the signal is concerned, it is convenient to specify an ideal bandpass filter covering the signals bandwidth.

As far as the noise is concerned, we need to know its spectral distribution and then to assess the reduction in noise level through the filter. This is quite readily done using results derived earlier in this chapter.

Assuming that the noise has zero mean value then its autocorrelation and autocovariance sequences are identical and the output noise power from the filter is given by an output version of Eq. 9.14:

$$\phi_{yy}[0] = \frac{1}{2\pi}\int_{2\pi} \boldsymbol{P}_{yy}(\Omega)d\Omega \qquad \text{Eq. 10.20}$$

Substitution from Eq. 10.10 gives:

$$\phi_{yy}[0] = \frac{1}{2\pi}\int_{2\pi} |\boldsymbol{H}(\Omega)|^2 \boldsymbol{P}_{xx}(\Omega)d\Omega \qquad \text{Eq. 10.21}$$

If the filter's squared-magnitude response has a bandwidth of $\Delta\Omega$ (see Figure 10.10 in the book), the output noise power becomes:

$$\phi_{yy}[0] = \frac{1}{2\pi}\int_{2\Delta\Omega} \boldsymbol{P}_{xx}(\Omega)d\Omega \qquad \text{Eq. 10.22}$$

A further simplification is possible if the input noise to the filter is assumed white, with unit variance. Its input power spectrum in then unity (see section 10.2), thus:

$$\phi_{yy}[0] = \frac{1}{2\pi}\int_{2\Delta\Omega} 1 d\Omega = \frac{\Delta\Omega}{\pi} \qquad \text{Eq. 10.23}$$

We see that the reduction in total noise power (or variance) through the filter is simply given by the ratio of the filter bandwidth to the frequency interval π. Since the signal is assumed to be transmitted unaltered, this (ratio)$^{-1}$ gives the improvement in signal-to-noise power ratio. The improvement in amplitude ratio is its square root.

10.4.2 Matched filter detection

matched filter

Detection of a signal in noise by use of a matched filter is one of the most important topics in linear DSP.

In the previous section we considered the problem of recovering a signal of unknown waveshape, using prior information about the frequency noise. A rather different problem arises when we know the signal's waveshape and we need to establish when, or indeed if, it occurs. The presents of such a signal can be detected by using a matched filter.

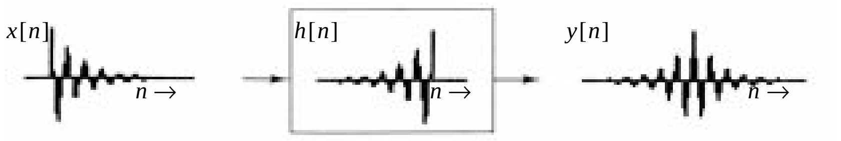


Figure 12
A matched filter.

Consider the time-limited input signal *x[n]* in Figure 12, processed by an LTI filter whose impulse response *h[n]* is a reversed version of *x[n]*. This filter is by definition the matched filter for this particular shape of input signal. The output is found by convolving the input and impulse response. This means the impulse response function is reversed hence the calculation of the output is simular to that of the autocorrelation function of *x[n]*. The autocorrelation function gives a maximum at the start of the sequence (see section 9.4).Thus the output signal has the same shape as the the autocorrelation function of the input signal. Finding the original signal in the sequence is now easy: the peaks could indicate the start of (a distorted version of) the original. If the sequence is only contaminated by noise, the matched filter is an ideal filter for signal detection.

The same effect occurs in the frequency domain. It turns out that the frequency response of a matched filter bears a simple relationship to the spectrum of the signal to which it is matched.

The frequency response in this case will be:

$$\begin{aligned} H(\Omega) &= \sum_{n=-\infty}^{\infty} x[-n]e^{-j\Omega n} \\ &= \sum_{k=-\infty}^{\infty} x[k]e^{j\Omega k} \\ &= X(-\Omega) \end{aligned}$$ Eq. 10.24

Now the spectrum $X(\Omega)$ of any real sequence can be expressed in the form:

$$X(\Omega) = A(\Omega) + jB(\Omega)$$ Eq. 10.25

Hence:

$$H(\Omega) = X(-\Omega) = A(-\Omega) + jB(-\Omega) = A(\Omega) - jB(\Omega) = X^*(\Omega)$$ Eq. 10.26

The output signal spectrum is now:

$$Y(\Omega) = X(\Omega)H(\Omega) = X(\Omega)X^*(\Omega) = |X(\Omega)|^2$$ Eq. 10.27

Index