KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Politechnika Wrocławska

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

ROZWÓJ POTENCJAŁU I OFERTY DYDAKTYCZNEJ POLITECHNIKI WROCŁAWSKIEJ

Wrocław University of Technology

Advanced Informatics and Control

D. P. Goodall, O. C. L. Haas

SIGNAL AND IMAGE PROCESSING

Wrocław 2011

Projekt współfinansowany ze środków Unii Europejskiej w ramach
Europejskiego Funduszu Społecznego

Wrocław University of Technology

Advanced Informatics and Control

D. P. Goodall, O. C. L. Haas

SIGNAL AND IMAGE PROCESSING

Wrocław 2011

Reviewers: Keith J. Burnham

ISBN 978-83-62098-89-7

Published by PRINTPAP Łódź, www.printpap.pl

Preface

This book is one of a series of Masters level monographs, which have been produced for taught modules within a common course designed for Advanced Informatics and Control. The new common course development forms a collaboration between Coventry University, United Kingdom, and Wroclaw University of Technology, Poland. The new course recognises the complexity of new and emerging advanced technologies in informatics and control, and each text is matched to the topics covered in an individual taught module. The source of much of the material contained in each text is derived from lecture notes, which have evolved over the years, combined with illustrative examples, which may well have been used by many other authors of similar texts. Whilst the sources of the material may be many, any errors that may be found are the sole responsibility of the authors.

Most forms of communication are usually performed using signals that are transmitted electronically. Such signals, which, for example, may be speech signals or image signals, often need to be processed to achieve some desired objective. Hence, the importance of methods for processing signals. In this text, we consider only certain aspects of signal processing, including speech processing, and discuss in detail some methods of image processing. Much of the work on image processing is based on using the computer software package MATLAB® and the Image Processing Toolbox.

Outline of the book

The material in this book has been developed for a set of eleven lecture/ tutorial (weekly) sessions. A possible teaching strategy, comprising the main topic areas, is listed in the following two tables; one table for speech processing and one table for image processing.

Lecture	Topic
	Signal Processing
	Signal processing concepts and discrete-time analysis
1	Classification of signals and systems; convolution; linear models; review of z-transform and properties.
	Discrete-time system responses
2	Inverse z-transforms; impulse and step responses; causality; system function; BIBO stability.
	Spectral analysis
3	Frequency and steady-state response for discrete-time systems; Fourier transform and properties; Fourier transform of periodic signals.
4	Sampling theorem and aliasing; discrete Fourier transform; discrete spectra; digital filters and windows.
	Random discrete-time processes
5	Random processes; autocorrelation and cross-correlation functions; white-noise process; response of linear systems with random inputs.
6a	Power spectral density; cross spectral density; application to response of linear, time-invariant filters; system identification; speech quantization and encoding.

Lecture	**Topic**
	Image Processing *Image processing concepts and fundamentals*
6b	Digital image processing fundamentals; sampling and quantization.
7	Image enhancement; histogram processing; types of noise.
8	Smoothing and sharpening filters; low pass, high pass filtering; region-oriented segmentation.
9	Mathematical morphology; relationship between pixels; point, line, edge and combined detection.
10	Case study 1: Automatic organ segmentation on CT images.
11	Case study 2: Target tracking using mathematical morphology.

The principal material of this book is composed of two parts. The first part considers important aspects of signal processing, whilst the second part introduces some specialised techniques for image processing.

In the first part, Chapter 1 introduces the main concepts of signals and linear systems. Also, since 'real-life' signals contain some random components and are not precisely known, in general, random signals (usually known as stochastic signals) are defined and some examples of well-known deterministic signals are presented. Such 'real-life' signals are often composed of more than one signal. Simplistically, the composition may be modelled in an additive fashion. However, in practice, some signals are composed through *convolution* and this process is explained, together with its properties, in Chapter 1.

Chapter 2 introduces discrete linear models and filters, through difference equations. An analysis tool for discrete-time analysis, namely the z-transform, is presented and some properties of the z-transform are studied. Using the z-transform, the discrete-time system response to a unit impulse sequence and the unit step sequence is investigated. The concept of a system function is introduced and, since stability of systems and filters is important, bounded-input bounded-output stability is discussed. Also, definitions of causality and finite impulse response/ infinite impulse response of systems and filters are provided. Finally, in this chapter, an application to speech production models for speech processing is presented.

As well as investigating systems/ filters in the time-domain, a frequency-domain approach can be used. Chapter 3 provides a study on frequency-domain techniques, utilising the frequency response function for a linear system, which can be applied to estimating the steady-state response of a system or filter. Using the Fourier transform, a sampling process is investigated and the so-called 'aliasing' problem analysed. Discrete Fourier transform and spectra are defined, as well as the power spectrum. Finally, Chapter 3 discusses the role of digital filters with respect to frequency components of the output signal, and the use of windows in the time-domain.

As signals are usually contaminated by noise, which may be a random signal, a study of the response of a linear system to random inputs is important. Chapter 4 introduces some special functions used for the analysis of systems with random inputs. Moreover, in many applications, autoregressive models are appropriate and, for such models, linear predictive filters can be used to identify the parameters of the model. These linear predictive filters are also studied in Chapter 4, and an application to speech processing is given. The power spectral density of a signal is defined and utilised for investigating the response of linear systems. Finally, in Chapter 4, speech quantization and encoding is studied.

In the second part of the book, a practical approach to digital image processing, with a minimal amount of mathematical derivations, is presented. Readers with a mathematical background and interested in the mathematical underpinning of specific techniques will be referred to one of the following books that were used to support the material presented in Part II: [Sonka et al., 1999, Nixon and Alberto, 2002, Forsyth and Ponce, 2003, Bernd, 2004] and, in addition, [Bovik, 2005]. In this second part of the book on image processing, [Gonzalez et al., 2004] is used as the main reference for some of the MATLAB®-based examples and explanations. The work, underpinning some the medical applications and research examples described in this book, can be found in the Part II of [Haas and Burnham, 2008].

Acknowledgements

With regard to Part I on Signal Processing, David Goodall is indebted to colleagues and students of Coventry University (U.K.) who helped him, directly or indirectly, in the preparation of this book; in particular, he would like to mention Mike Chapman (at Coventry University). In addition, David Goodall would like to give special thanks to Mike Herring, formerly of Cheltenham and Gloucester College of Higher Education (U.K.), during initial exploratory excursions into Speech Processing. Furthermore, David Goodall acknowledges the support given by his wife, Alison, and his three daughters, Tanja, Emma and Fiona, whilst preparing this book.

With regard to the Part II on Image Processing, Olivier Haas acknowledges the support given by his wife, who had to cope with their young children, whilst he was busy preparing the chapters for this book.

Contents

Glossary

Nomenclature

$\approx$	approximately equal to
$\in$	belongs to
$\stackrel{\text{def}}{=}$	defined as
$\square$	end of example or proof
!	factorial
$\exp$	exponential
iff	if and only if
$\lim$	limit
$\max$	maximum
$\mathbb{Z}$	set of integers
$\mathbb{N}$	set of natural numbers
$\mathbb{R}$	set of real numbers
$\mathbb{R}^+$	$\{r \in \mathbb{R} : r > 0\}$
$\mathbb{C}$	set of complex numbers
$\cap$	intersection of sets
$\|a\|$	absolute value of $a \in \mathbb{R}$
i	satisfies $i^2 = -1$
$\Re\{\cdot\}$	real part
$\Im\{\cdot\}$	imaginary part
$\|z\|$	modulus of $z \in \mathbb{C}$
$\arg[\cdot]$	argument of a complex number
$\mathrm{conj}(\cdot)$	complex conjugate
$f'(\cdot)$	1^{st} derivative of f
$f^{(n)}(\cdot)$	n^{th} derivative of f
$\oint_C$	integration around a closed contour C
$\delta(\cdot)$	unit impulse signal
$\zeta(\cdot)$	unit step signal
$p_\alpha(\cdot)$	unit pulse signal of width α
$r(\cdot)$	unit ramp signal
$a(\cdot)$	exponential signal

Nomenclature (cont.)

y_δ	impulse response
y_ζ	step response
$(x * y)(\cdot)$	convolution of x and y
$\{x(\cdot)\}$	denotes the sequence x
$\mathcal{Z}\left[\{\cdot\}\right]$	z-transform
$X(z)$	bilateral z-transform of $\{x(k)\}$
$X_+(z)$	unilateral z-transform of $\{x(k)\}$
$\mathcal{Z}^{-1}\left[\cdot\right]$	inverse z-transform
$H\left(e^{i\theta}\right)$	system frequency response
$\left\lvert H\left(e^{i\theta}\right)\right\rvert$	system gain spectrum
$\arg\left[H\left(e^{i\theta}\right)\right]$	system phase-shift spectrum
AR	autoregressive
ARMA	autoregressive moving average
MA	moving average
FIR	finite impulse response
IIR	infinite impulse response
BIBO	bounded-input bounded-output
$\mathcal{F}\{\cdot\}$	Fourier transform
$F(i\omega)$	Fourier transform of $f(t)$
$\mathcal{F}^{-1}\{\cdot\}$	inverse Fourier transform
$\lvert F(i\omega)\rvert$	amplitude spectrum of $f(t)$
$\theta(\omega)$	phase spectrum of signal $f(t)$
$E(\cdot)$	energy in a signal
$P_{av}(\cdot)$	average power in a signal
$G(\omega)$	energy spectral density
$\lvert F(i\omega)\rvert^2$	energy spectrum of signal $f(t)$
ω_0	fundamental frequency

Nomenclature (cont.)

T	(sampling) period
$\gamma_T(\cdot)$	unit comb function, with period T
$p(t)$	continuous-time sampling function
f_s	sampling frequency
x_s	sampled signal
DFT	discrete Fourier transform
IDFT	inverse discrete Fourier transform
DTFT	discrete-time Fourier transform
FFT	fast Fourier transform
$X(n)$	DFT of signal $\{x(k)\}$
$\lvert X(n)\rvert$	DFT amplitude spectrum
ψ_n	DFT phase spectrum
$P(n)$	power spectrum
$\{\text{rect}_N(k)\}$	rectangular window sequence
μ_x	mean of random variable x
σ_x or $\overline{x}$	standard deviation of x
σ_x^2 or $\text{Var}[x]$	variance of x
$\mathcal{E}\left[\cdot\right]$	expected/ average value
$R_{xx}(\cdot)$	autocorrelation function
$C_{xx}(\cdot)$	autocovariance function
$\langle\cdot\rangle$	time average
$R_{xy}(\cdot)$	cross-correlation function
$C_{xy}(\cdot)$	cross-covariance function
$(S/N)_{out}$	output signal-to-noise ratio
$S_{xx}(\theta)$	power spectral density
$S_{xy}(\theta)$	cross spectral density

Useful MATLAB$^{\circledR}$ commands for image processing

`adapthisteq(I)` Enhances the contrast of the intensity image `I` by transforming the values using contrast-limited adaptive histogram equalization (CLAHE).

`MOV=aviread(FILENAME)` Reads the AVI movie FILENAME into the movie structure MOV. If FILENAME does not include an extension, then '`.avi`' will be used.

`MOV=aviread(...,INDEX)` Reads only the frame(s) specified by INDEX. INDEX can be a single index or an array of indices into the video stream, where the first frame is number one.

`L=BWLABEL(BW,N)` Returns a matrix L, of the same size as BW, containing labels for the connected components in BW. N=4 specifies 4-connected objects and N=8 specifies 8-connected objects; if argument is omitted,then the default is 8.

`C=conv2(A,B)` Computes the two-dimensional convolution of `A` and `B`. If one of these matrices describes a two-dimensional finite impulse response (FIR) filter, the other matrix is filtered in two dimensions. The size of `C`, in each dimension, is equal to the sum of the corresponding dimensions of the input matrices, minus one. That is, if the size of `A` is $[ma, na]$ and the size of `B` is $[mb, nb]$, then the size of `C` is $[ma + mb - 1, na + nb - 1]$.

`C=conv2(A,B,'shape')` Returns a subsection of the two-dimensional convolution, as specified by the shape parameter:
`'full'` returns the full two-dimensional convolution (which is the default);
`'same'` returns the central part of the convolution of the same size as `A`;
`'valid'` returns only those parts of the convolution that are computed without the zero-padded edges. For this option, `C` has size
when $ma \geq mb$ and $na \geq nb$. Otherwise `conv2` returns [ ].

`corr2(A,B)` Computes the correlation coefficient between `A` and `B`, where `A` and `B` are matrices or vectors of the same size.

`double(X)` Returns the double-precision value of `X`. If `X` is already a double-precision array, `'double'` has no effect.

Useful MATLAB® commands for image processing (cont.)

`B=edge(I)` Takes an intensity image `I` as its input and returns a binary image `B` of the same size as `I`, with 1's where the function finds edges in `I` and 0's elsewhere. An example is: `B=edge(I,'canny',thresh,sigma)`.

`find(X)` Finds indices of non-zero elements by returning the linear indices corresponding to the non-zero entries of the array X. X may be a logical expression.

`filter2(B,X)` Filters the data in X with the 2-D FIR filter in thematrix B. This is computed using 2-D correlation and is the same size as X.

`H=fspecial('gaussian')` Creates a rotationally symmetric Gaussian lowpass filter.

`H=fspecial('gaussian',hsize,sigma)` Creates a rotationally symmetric Gaussian lowpass filter of size ‘hsize’, with standard deviation ‘sigma’. ‘hsize’ can be a vector specifying the number of rows and columns in H, or a scalar.

`im2bw(I,level)` Converts greyscale image I to a binary image. The output image replaces all pixels in the input image with luminance greater than the level with a value of 1 (white) and replaces all other pixels with a value of 0 (black).

`image(C)` Creates an image graphics object by interpreting each element in a matrix `C` as an index into the figure's colormap, or directly as RGB values, depending on the data specified.

`imfilter(A,H)` Filters the multi-dimensional array `A` with the multidimensional filter `H`.

`imhist(I,N)` Displays a histogram with N bins for intensity image I above a greyscale colour-bar of length N.
If I is a binary image, then N=2.

Useful MATLAB® commands for image processing (cont.)

`imread(A)`	Returns the image data in the array `A`.
`imread(FILENAME,FMT)`	Reads a greyscale or colour image from the file specified by the string FILENAME, where the string FMT specifies the format of the file.
`imshow(I)`	Displays the intensity image `I`.
`irerode(IM,SE)`	Erodes the greyscale, binary, or packed binary image IM, returning the eroded image, IM2. SE is a structuring element object, or array of structuring element objects, returned by the `strel` function.
`nlfilter(A,[M N],FUN)`	For general sliding-neighborhood operations, the command applies the function FUN to each M-by-N sliding block of A. FUN is a function that accepts an M-by-N matrix as input and returns a scalar: `C = FUN(X)`. C is the output value for the central pixel in the M-by-N block X. FUN must be a FUNCTION_HANDLE. Here NLFILTER calls FUN for each pixel in A. NLFILTER zero pads the M-by-N block at the edges, if necessary.
`medfilt2(A,[M N])`	Performs median filtering of matrix `A`, where each output pixel contains the median value in the M-by-N neighbourhood around the corresponding pixel in the input image.
`ordfilt2(A,ORDER,DOMAIN)`	Replaces each element in A by the ORDER-th element in the sorted set of neighbours specified by the nonzero elements in DOMAIN.
`regionprops(L,properties)`	Measures a set of properties for each labelled region in the label matrix `L`.
`I=rgb2gray(RGB)`	Converts the truecolor image RGB to the greyscale intensity image `I`.
`strel(shape,parameters)`	Creates a structuring element, SE, of the type specified by shape. Depending on shape, strel can take additional parameters.
`uint8(X)`	Converts the elements of array `X` into unsigned integers, where the unsigned 8-bit integer is in the range 0 to 255.

List of Figures

List of Tables

Part I

Signal Processing

Chapter 1

Signals and systems

1.1 Introduction

In order to develop the analytical tools for processing signals, basic signal and systems classifications are necessary. These can be found in many books on signal processing, such as [Balmer, 1997, Ziemer et al., 1998]. Currently in view of electronic communications and many other applications, analysis and processing of discrete-time signals and, in addition, analysis of discrete-time systems is very important. These topics have been studied by many authors, such as [Bellanger, 1990, Jackson, 1996, Jong, 1982, Oppenheim et al., 1999, Stearns and Hush, 1990] and [Proakis and Manolakis, 1989], to name but a few.

Often, in practice, many signals are complex in nature, in that the signals may consist of a combination of many signals. To analyse such signals some assumptions are made, such as, for example, assuming the signal of interest is composed of a linear combination of other signals. However, this is not the only possibility. For example, a signal may be composed of a convolution of signals. This topic is discussed in this chapter, as well as classification of signals and systems, both continuous-time and discrete-time.

1.2 Classification of signals and systems

A signal may be a function of one or more independent variables, e.g. a voice signal is a function of time, whereas an image signal is a function of two spatial variables.

Definition 1.1 *A* ***continuous-time*** *signal is a signal $x(t)$ that varies continuously with time t, where $t \in \mathbb{R}$ or an appropriate subset.*

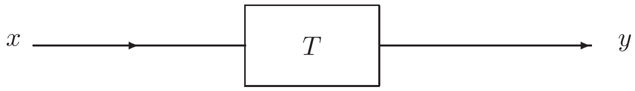
Remark 1.1 *A continuous-time signal is not necessarily a continuous function of time; for example,*

$$x(t) = \begin{cases} 1, & t \geq 0, \\ 0, & t < 0, \end{cases}$$

is a discontinuous continuous-time signal.

Definition 1.2 *A* ***discrete-time*** *signal is a signal $x(k)$ which is defined only at a sequence of discrete values k, i.e. $k \in I$ (a subset of the set of integers, viz. $I \subset \mathbb{Z}$).*

The amplitude of a signal may assume a continuous range of values or a set of discrete values. In the latter case, the signal is said to be *quantized*. A continuous-time signal with continuous range of amplitude is sometimes referred to as an *analogue* signal, whilst a quantized discrete-time signal is referred to as a *digital* signal. In most engineering applications, the signal is analogue in nature and is to be processed using a digital computer. This is accomplished by sampling the continuous-time signal at discrete values to produce a *sampled-data* signal. Some different types of signal are shown in Figure 1.1.
A system is defined in terms of a transformation (or operator) that maps input signal(s) into output signal(s). For the system illustrated in the diagram:



we may write $y = Tx$.

Systems can be classified in the same way as signals, i.e. *continuous-time* systems, *discrete-time* systems, *analogue* and *digital* systems. Furthermore, a system may be classified as either *linear* or *nonlinear*.

Definition 1.3 *If x_1 and x_2 are two inputs to a system with responses (i.e. outputs) y_1, y_2, respectively, then a system is* ***linear*** *iff*

$$T(ax_1 + bx_2) = aTx_1 + bTx_2 = ay_1 + by_2,$$

where T is the system operator and a, b are arbitrary constants (this is known as the ***principle of superposition****); otherwise, the system is said to be* ***nonlinear****.*

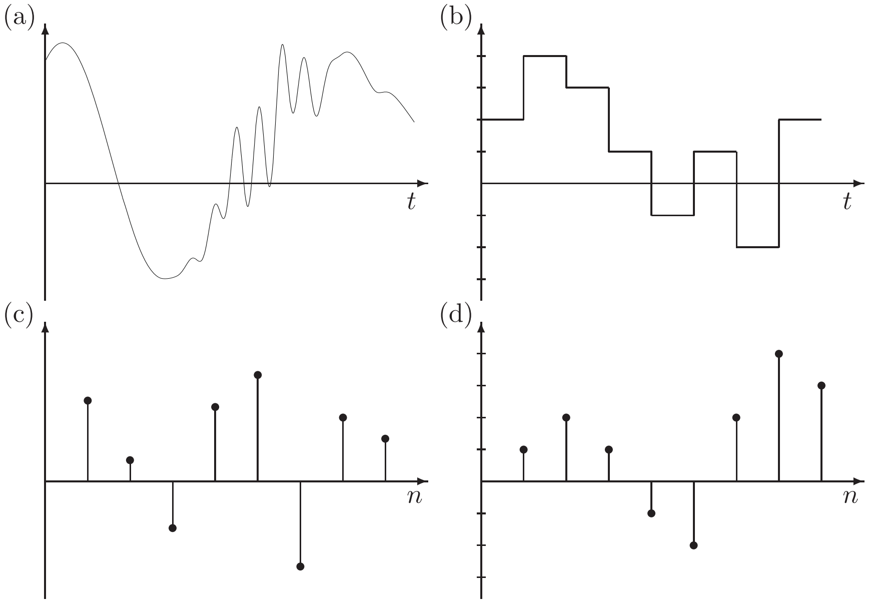


Figure 1.1: Various types of signals.

A *time-invariant* system is one in which none of the parameters of the system change with time, i.e., for a continuous-time system, if $Tx(t) = y(t)$ then $Tx(t-\tau) = y(t-\tau)$. In a *time-varying* system the input-output relationship changes with time.

Sometimes, the concept of 'memory' may be important for a system. In particular, a discrete-time, memoryless system is defined in Definition 1.4.

Definition 1.4 *A discrete-time system is* ***memoryless*** *iff the output $y(k)$ at every value k depends only on the input $x(k)$ at the same value of k.*

For example, the system in which $x(k)$ and $y(k)$ are related by $y(k) = [x(k)]^2$, for each value of k, is memoryless, whereas the *ideal delay system* for which $y(k) = x(k-\kappa)$, where κ is a fixed positive constant (known as the delay of the system), is not memoryless.

Another important concept is that of 'causality'.

Definition 1.5 *A discrete-time system is* ***causal*** *(or* ***nonanticipatory****) iff the output $y(k)$ depends on the input $x(k_0)$ for $k \geq k_0$ (i.e. the response to an input does not depend on future values of that input).*

Analogous to Definitions 1.4 and 1.5, memoryless and causal continuous-time systems can also be defined.

1.3 Deterministic/stochastic signals

Deterministic signals are signals which can be modelled by known functions of time. Some examples of deterministic signals are given in Table 1.1.

Non-deterministic signals are known as *stochastic*, which are *random* in nature. Signals, containing both random and deterministic elements, often occur when obtained from measurements of some process. Measurements are typically contaminated with *noise*, usually considered as a random signal, either from sensors, measurement instrumentation, the surrounding environment, or even the process itself.

1.4 Convolution of signals

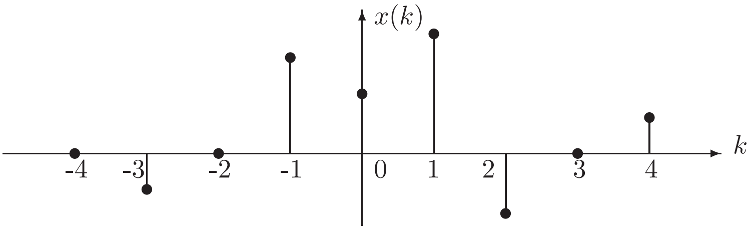
Convolution of signals is a very important topic in signal processing. A study of this can be found in [Gabel and Roberts, 1987, Ziemer et al., 1998] and Chap-

signal	continuous-time ($t \in \mathbb{R}$)	discrete-time ($k \in \mathbb{Z}$)
unit step	$\zeta(t) \overset{\text{def}}{=} \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases}$	$\zeta(k) \overset{\text{def}}{=} \begin{cases} 0, & k < 0 \\ 1, & k \geq 0 \end{cases}$
unit ramp	$r(t) \overset{\text{def}}{=} \begin{cases} 0, & t < 0 \\ t, & t \geq 0 \end{cases}$	$r(k) \overset{\text{def}}{=} \begin{cases} 0, & k < 0 \\ k, & k \geq 0 \end{cases}$
unit pulse	$p_\alpha(t) \overset{\text{def}}{=} \begin{cases} 1, & 0 \leq t < \alpha, \ \alpha \in \mathbb{R}^+ \\ 0, & \text{otherwise} \end{cases}$	$p_\alpha(k) \overset{\text{def}}{=} \begin{cases} 1, & 0 \leq k < \alpha, \ \alpha \in \mathbb{N} \\ 0, & \text{otherwise} \end{cases}$
unit impulse	$\delta(t) \overset{\text{def}}{=} \lim_{\alpha \to 0} \frac{1}{\alpha} p_\alpha(t)$	$\delta(k) \overset{\text{def}}{=} \begin{cases} 1, & k = 0 \\ 0, & k \neq 0 \end{cases}$
exponential	$a(t) \overset{\text{def}}{=} \alpha^t$	$a(k) \overset{\text{def}}{=} \alpha^k$

Table 1.1: Deterministic signals.

ter 2 of [Proakis and Manolakis, 1989]. A further study of this topic, involving extensive use of the computer software package MATLAB®, is given in [Ingle and Proakis, 1997].

Consider, initially a discrete deterministic signal $\{x(k)\}$. The general term of this (arbitrary) sequence can be expressed as a sum of scaled, delayed unit impulses.



For the example shown above,

$$x(k) = a_{-3}\delta(k+3) + a_{-1}\delta(k+1) + a_0\delta(k) + a_1\delta(k-1) + a_2\delta(k-2) + a_4\delta(k-4),$$

where a_i are the magnitudes of the appropriate impulses. More generally, the general term of an arbitrary sequence can be expressed as

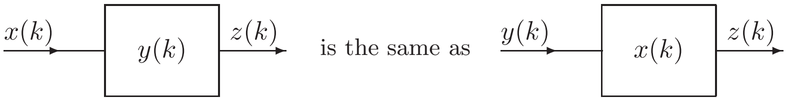
$$x(k) = \sum_{m=-\infty}^{\infty} x(m)\delta(k-m). \tag{1.1}$$

Definition 1.6 *The* ***convolution*** *of the two sequences* $\{x(k)\}$ *and* $\{y(k)\}$ *is defined by*

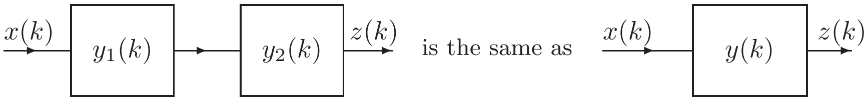
$$\boxed{(x * y)(k) \overset{\text{def}}{=} \sum_{m=-\infty}^{\infty} x(m)y(k-m).}$$

Important properties of convolution for discrete-time signals are:

(CD1) Commutativity: $(x * y)(k) = (y * x)(k)$

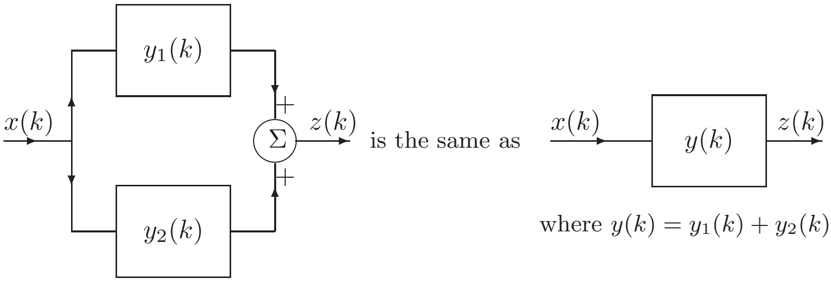


(CD2) Associativity: $((x * y_1) * y_2)(k) = (x * (y_1 * y_2))(k)$



where $y(k) = (y_1 * y_2)(k)$

(CD3) Distributivity: $(x * (y_1 + y_2))(k) = (x * y_1)(k) + (x * y_2)(k)$



where $y(k) = y_1(k) + y_2(k)$

(CD4) By definition of $\delta(k)$ and convolution, the impulse sequence has the property that if $y(k) = \delta(k - k_0)$, $k_0 \in \mathbb{Z}$, then $(x * y)(k) = x(k - k_0)$.

Example 1.1 Evaluate the convolution $(x * y)(k)$ when

a) $x(k) = a^k\zeta(k)$, with $a \in \mathbb{R}$, and $y(k) = p_3(k)$, where $\{p_3(k)\}$ is the unit pulse sequence defined in Table 1.1;

b) $x(k) = y(k) = \zeta(k)$.

Solution:

a) Now, by definition of $\{p_3(k)\}$, $y(k)$ can be expressed in the form $y(k) = \delta(k) + \delta(k-1) + \delta(k-2)$ and, hence, by the convolution impulse sequence property (CD4) it follows that

$$(x * y)(k) = a^k\zeta(k) + a^{k-1}\zeta(k-1) + a^{k-2}\zeta(k-2) = \sum_{j=0}^{2} a^{k-j}\zeta(k-j).$$

b) By Definition 1.6,

$$(x * y)(k) = \sum_{m=-\infty}^{\infty} \zeta(m)\zeta(k-m).$$

It can be verified that the following result holds for all m:

$$\zeta(m)\zeta(k-m) = \begin{cases} 0, & k < 0, \\ \zeta(m) - \zeta(m-k-1), & k \geq 0. \end{cases}$$

Now $\zeta(m) - \zeta(m-k-1) = p_{k+1}(m) = \begin{cases} 1, & 0 \leq m \leq k, \\ 0, & \text{otherwise}, \end{cases}$ and, therefore,

$$(x*y)(k) = \begin{cases} 0, & k < 0, \\ \displaystyle\sum_{m=0}^{k} 1, & k \geq 0, \end{cases} = \begin{cases} 0, & k < 0, \\ k+1, & k \geq 0, \end{cases} = (k+1)\zeta(k). \quad (1.2)$$

Alternatively, since $\zeta(k)$ may be represented by $\sum_{n=0}^{\infty} \delta(k-n)$, then

$$(x * y)(k) = \zeta(k) * \left(\sum_{n=0}^{\infty} \delta(k-n)\right) = \sum_{n=0}^{\infty} \zeta(k) * \delta(k-n).$$

Using the convolution property for impulse functions, see (CD4),

$$(x * y)(k) = \sum_{n=0}^{\infty} \zeta(k-n).$$

Noting that $\zeta(k-n) = 0$ when $k < 0$ and when $n > k$, this leads to (1.2).

□

Example 1.2 Using MATLAB®,

a) plot the sequence obtained from the convolution of the sequences $\{x\} = \{1,\ 1,\ 1,\ 2,\ 2,\ 2,\ 3,\ 3,\ 3\}$ and $\{y\} = \{1,\ -1,\ 1,\ -1,\ 1\}$;

b) confirm the result obtained in (a) by deconvolving the sequence $\{y\}$ from the sequence $\{x * y\}$.

Solution:

a) The following MATLAB® commands may be used:

```
z=[1 1 1]; x=[z 2*z 3*z];  % generation of sequence x
y=[1 -1 1 -1 1];           % generation of sequence y
c=conv(x,y);               % forms convolution sequence
stem(c)           % discrete plot of convolution sequence
```

which produce the graph shown in Figure 1.2(a).

b) The MATLAB® commands:

```
% deconvolving the sequence y from the sequence x*y
[q,r]=deconv(c,y);
stem(q)
```

produce the graph shown in Figure 1.2(b).

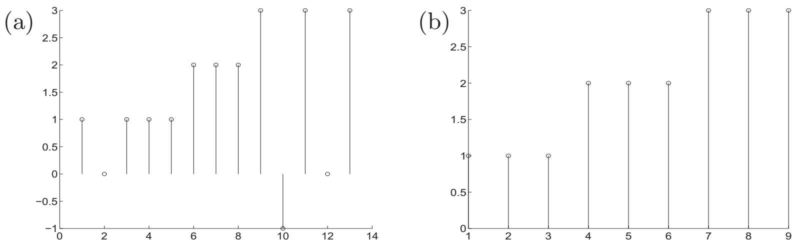


Figure 1.2: (a) Convolution of $\{x\}$ and $\{y\}$. (b) Deconvolving $\{y\}$ from $\{x*y\}$.

□

Example 1.3 Determine the convolution of the two sequences: $\{x(k); k = 2 : 1 : 5\} = \{-2, 4, 1, -1\}$ and $\{y(k); k = -3 : 1 : 1\} = \{3, 1, 0, -2, 5\}$, using MATLAB®, and, in particular, write down the value $(x * y)(0)$.

Solution: As in Example 1.2, MATLAB® can be used to determine the convolution of the two sequences. However, in this example, the time-indices for the two sequences do not begin at zero. Hence, the time-index information for the convolved sequence is required. The MATLAB® commands:

```
x=[-2 4 1 -1];
y=[3 1 0 -2 5];
tix=[2:5];                        % time-indices for x
tiy=[-3:1];                       % times-indices for y
cb=tix(1)+tiy(1);                 % beginning of time-index for x*y
ce=tix(length(x))+tiy(length(y)); % end of time-index for x*y
tixy=[cb:ce]                      % time-index interval
convxy=conv(x,y)
```

produce the results:

```
tixy =

    -1      0      1      2      3      4      5      6

convxy =

    -6     10      7      2    -19     18      7     -5
```

Thus, it is clear that $(x * y)(0) = 10$.

□

Consider, now, the case of continuous-time signals.

Definition 1.7 *For continuous-time signals, f and g defined on $(-\infty, \infty)$, the convolution operation is defined as*

$$(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, \mathrm{d}\tau.$$

Analogous properties, to (CD1-4), hold for convolution of continuous-time signals, namely:

(CC1) Commutativity: $(f * g)(t) = (g * f)(t)$;

(CC2) Associativity: $((f * g_1) * g_2)(t) = (f * (g_1 * g_2))(t)$;

(CC3) Distributivity: $(f * (g_1 + g_2))(t) = (f * g_1)(k) + (f * g_2)(t)$;

(CC4) if $g(t) = \delta(t - a)$, $a \in \mathbb{R}$, and f is continuous, then $(f * g)(t) = f(t - a)$ holds.

1.5 Exercises

Exercise 1.1 Consider the following discrete-time systems, modelled by $y = Tx$ (where x is the input and y is the corresponding output):

a) $Tx(k) = (k - 1)x(k - 1)$;

b) $Tx(k) = \sin(x(k))x(k)$;

c) $Tx(k - 1) = x(k - 2) - 3x(k - 1)$;

d) $Tx(k - 1) = \sin(k)x(k)$.

Determine whether the systems are (i) linear (ii) time-invariant (iii) memoryless (iv) causal.

Exercise 1.2 A discrete signal $\{x(k)\}$ is defined by

$$x(k) \overset{\text{def}}{=} \begin{cases} 0, & k < 0, \\ 1, & 0 \leq k \leq 10, \\ 0, & 11 \leq k \leq 15, \\ 1, & 16 \leq k \leq 20, \\ 0, & k \geq 21. \end{cases}$$

Express $x(k)$ in terms of the unit step function $\zeta(k)$.

Exercise 1.3 If the discrete-time signal $\{z(k)\}$ is such that $z(k) = (x * y)(k)$, where $x(k) = 2^{-k}\zeta(k)$ and $y(k) = 2\delta(k) + 4\delta(k - 1) + 4\delta(k - 2)$, calculate and plot $z(k)$ for $k = 0, 1, 2, 3, 4, 5$ and 6.

Exercise 1.4 Confirm that, for all m,

$$\zeta(m)\zeta(k-m) = \begin{cases} 0, & k < 0, \\ \zeta(m) - \zeta(m-k-1), & k \geq 0. \end{cases}$$

Hence, or otherwise, evaluate the convolution $(x * y)(k)$, where $x(k) = \zeta(k)$ and $y(k) = a^k \zeta(k)$ with $a \in \mathbb{R}^+$ satisfying $0 < a < 1$.

Exercise 1.5 Using MATLAB®, determine the convolution of the following two sequences:

a) $\{x(k)\} = \{1,\ -1,\ 0,\ 2,\ 4\}$ and $\{y(k)\} = \{5,\ -2,\ 5\}$;

b) $\{x(k); k = -2:1:2\} = \{3,\ 0,\ 0,\ -2,\ -1\}$ and
$\{y(k); k = 1:1:4\} = \{2,\ -2,\ 0,\ -2\}$.

Exercise 1.6 Compute the convolution of the pair of discrete-time signals:

a) $x_1(k) = \zeta(k) - \zeta(k-M)$, $\quad x_2(k) = \delta(k)$, $\quad$ where $M \geq 1$ is a fixed real constant;

b) $x_1(k) = \zeta(k) - \zeta(k-M)$, $\quad x_2(k) = \zeta(k)$.

Exercise 1.7 Evaluate the convolution of the continuous-time signals $f(t) = \zeta(t - T_1) - \zeta(t - T_2)$, with $T_2 > T_1 > 0$, and $g(t) = e^{-at}\zeta(t)$, with $a \in \mathbb{R}^+$.

Chapter 2

System models, responses and the system function

2.1 Introduction

In this chapter, discrete discrete linear models and filters are introduced through difference equations. To analyse such models and filters appropriate analysis tools are required. One of the most important for digital systems is the concept of a z-transform. Definitions and properties are given in this chapter.

In addition, discrete-time system properties, such as stability and responses of a system to impulse and step inputs are examined. Moreover, a particular application, namely speech processing, is introduced and an appropriate model is presented for voiced/unvoiced speech.

2.2 Discrete linear models and filters

Suppose that in a certain discrete-time system the increase in output $y(k)$ at time k is proportional to the increase in input $x(k)$ at time $k-1$. The input-output relationship of the discrete-time system can be modelled by

$$y(k) - y(k-1) = c[x(k-1) - x(k-2)],$$

where c is the constant of proportionality, which may be rewritten as

$$y(k) = y(k-1) + cx(k-1) - cx(k-2). \tag{2.1}$$

This equation is called a *difference equation*. It relates the output signal $y(k)$ to its previous value $y(k-1)$ and the two past values of the input signal, $x(k-1)$ and $x(k-2)$. Since $y(k)$ is a linear combination of $y(k-1)$, $x(k-1)$ and $x(k-2)$, (2.1) is termed a *linear* difference equation. A general linear difference equation has the form:

$$y(k) = \sum_{i=1}^{N} a_i(k)y(k-i) + \sum_{j=0}^{M} b_j(k)x(k-j), \tag{2.2}$$

with $a_N(k) \neq 0, \quad b_M(k) \neq 0$.

Remark 2.1 *If $a_i = 0$, for all i, then $y(k)$ is referred to as a* moving-average *(MA) process, whilst if $b_j = 0$, for all j except $j = 0$, and $a_N \neq 0$ then $y(k)$ is called an* autoregressive *(AR) process. In general, $y(k)$ (given by* (2.2)*) is known as an* autoregressive moving-average *(ARMA) process.*

The integer N is the *order* of the difference equation.

Suppose the elimination of some undesirable component of a discrete-time signal (for example noise), or the extraction of signal components within a certain range of frequencies, is required. The signal may be *filtered* using a linear difference equation. For example, in a particular process the difference equation

$$18y(k) = 3y(k-1) + 4y(k-3) + 8x(k) - 12x(k-2)$$

is used. In this context the third order difference equation is termed a *digital filter* and, since all the coefficients are constant, is a *time-invariant* filter.

2.3 z-transforms

The z-transform method is a transform method suitable for the analysis of discrete-time systems, whereas the familiar Laplace transform method is used for continuous-time systems when time is restricted to the interval $[0, \infty)$.

2.3.1 z-transform of a sequence

The z-transform of a sequence $\{x(k)\}$ is defined as

$$\mathcal{Z}\left[\{x(k)\}\right] \overset{\text{def}}{=} \sum_{k=-\infty}^{\infty} x(k)z^{-k} = X(z),$$

where the values of $z \in \mathbb{C}$ are such that the series converges. This is sometimes known as a *two-sided* (or *bilateral*) z-transform. Consider, for example, the z-transform of the finite sequence $\{x(k)\}$ given by

$$x(k) = \begin{cases} (k-1)(-2)^{k+1}, & |k| \leq 2 \\ 0, & \text{otherwise} \end{cases}$$

(i.e. $x(-2) = \frac{3}{2}, \quad x(-1) = -2, \quad x(0) = 2, \quad x(1) = 0, \quad x(2) = -8$, otherwise $x(k) = 0$, or, alternatively, $x(k) = \frac{3}{2}\delta(k+2) - 2\delta(k+1) + 2\delta(k) - 8\delta(k-2)$).

$$X(z) = \sum_{k=-2}^{2} x(k)z^{-k} = \frac{3}{2}z^2 - 2z + 2 - \frac{8}{z^2},$$

which converges for all z except at $z = 0$ and $z = \infty$, i.e. the region of convergence is $\{z \in \mathbb{C} : \ 0 < |z| < \infty\}$.
A *one-sided* (or *unilateral*) z-transform of a sequence $\{x(k); \ k \geq 0, \ k \in \mathbb{Z}\}$ is defined by

$$\mathscr{Z}\left[\{x(k)\}\right] \stackrel{\text{def}}{=} \sum_{k=0}^{\infty} x(k)z^{-k}.$$

Example 2.1 Find the z-transform and region of convergence for the sequence $\{x(k)\}$ given by

a) $x(k) = a^k\zeta(k)$ b) $x(k) = -b^k\zeta(-k-1)$,

where $a, \ b \in \mathbb{C}$.

Solution:

a) By definition,

$$X(z) = \mathscr{Z}\left[\left\{a^k\zeta(k)\right\}\right] = \sum_{k=-\infty}^{\infty} a^k\zeta(k)z^{-k} = \sum_{k=0}^{\infty}(az^{-1})^k,$$

which converges to $1/\left(1 - az^{-1}\right)$, provided $\left|az^{-1}\right| < 1$,
i.e. $X(z) = \dfrac{z}{z-a}$, with region of convergence $\{z \in \mathbb{C} : |z| > |a|\}$.

b) $X(z) = -\sum_{k=-\infty}^{\infty} b^k \zeta(-k-1) z^{-k}$, but $\zeta(-k-1) = \begin{cases} 0, & k \geq 0, \\ 1, & k < 0, \end{cases}$ and, therefore,

$$X(z) = -\sum_{k=-\infty}^{-1} b^k z^{-k} \overset{(k=-n)}{=} -\sum_{n=1}^{\infty} (b^{-1}z)^n.$$

Note that

$$\sum_{n=1}^{\infty} a^n = \left(\sum_{n=0}^{\infty} a^n\right) - 1 \text{ and so } X(z) = 1 - \sum_{n=0}^{\infty} (b^{-1}z)^n,$$

which converges to $1 - 1/(1 - b^{-1}z)$, provided $\left|b^{-1}z\right| < 1$,
i.e. $X(z) = \dfrac{z}{z-b}$, with region of convergence $\{z \in \mathbb{C} : |z| < |b|\}$.

□

Example 2.1 illustrates that different sequences can have the same z-transform, except that the regions of convergence are different. For ease of computation, a table of z-transforms is provided in Table B.1 (Appendix B).

We now list some of the important properties of the z-transform.

(Z1) **Linearity**: If $\mathcal{Z}[\{x(k)\}] = X(z)$ with region of convergence $R_1 = \{z \in \mathbb{C} : \alpha_1 < |z| < \alpha_2\}$ and $\mathcal{Z}[\{y(k)\}] = Y(z)$ with region of convergence $R_2 = \{z \in \mathbb{C} : \beta_1 < |z| < \beta_2\}$, then, for any constants a and b,

$$\mathcal{Z}[\{ax(k) + by(k)\}] = aX(z) + bY(z)$$

with region of convergence $R_1 \cap R_2$.

(Z2) **Delay/Advance property**: Let $\mathcal{Z}[\{x(k)\}] = X(z)$ with region of convergence $R = \{z \in \mathbb{C} : c_1 < |z| < c_2\}$ then

$$\mathcal{Z}[\{x(k \pm m)\}] = z^{\pm m} X(z)$$

with region of convergence R.
Consider the sequence $\{x(k)\zeta(k)\} = \{x(0),\ x(1),\ x(2), \ldots\}$.
If $\mathcal{Z}[\{x(k)\zeta(k)\}] = X_+(z)$, where $X_+(z)$ denotes the unilateral z-transform

of $\{x(k)\}$, then the sequence $\{x(m),\ x(m+1),\ldots\}$ has z-transform

$$\begin{aligned}\mathcal{Z}\left[\{x(k+m)\zeta(k)\}\right] &= \sum_{k=0}^{\infty} x(k+m)z^{-k}\\ &= \sum_{j=m}^{\infty} x(j)z^{-j+m} \quad (\text{with } j=k+m)\\ &= z^m\left\{\sum_{j=0}^{\infty} x(j)z^{-j} - \sum_{j=0}^{m-1} x(j)z^{-j}\right\}\\ &= z^m X_+(z) - z^m \sum_{j=0}^{m-1} x(j)z^{-j}.\end{aligned}$$

(Z3) **Multiplication by an exponential** a^k: Suppose $\mathcal{Z}\left[\{x(k)\}\right] = X(z)$ with region of convergence $R = \{z \in \mathbb{C} :\ c_1 < |z| < c_2\}$, then

$$\mathcal{Z}\left[\left\{a^k x(k)\right\}\right] = X\left(\frac{z}{a}\right),$$

with region of convergence $\{z \in \mathbb{C} :\ |a|c_1 < |z| < |a|c_2\}$.

(Z4) **Multiplication by** k: Let $\mathcal{Z}\left[\{x(k)\}\right] = X(z)$ with region of convergence $R = \{z \in \mathbb{C} :\ c_1 < |z| < c_2\}$, then

$$\mathcal{Z}\left[\{kx(k)\}\right] = -z\frac{\mathrm{d}X}{\mathrm{d}z}(z),$$

with region of convergence R.

(Z5) **Convolution**: If $\mathcal{Z}\left[\{x(k)\}\right] = X(z)$ with region of convergence R_x and $\mathcal{Z}\left[\{y(k)\}\right] = Y(z)$ with region of convergence R_y, then, for

$$\begin{aligned}(x*y)(k) &= \sum_{m=-\infty}^{\infty} x(m)y(k-m),\\ \mathcal{Z}\left[\{(x*y)(k)\}\right] &= X(z)Y(z),\end{aligned}$$

with region of convergence $R_x \cap R_y$.

(Z6) **Initial Value Theorem**: For the sequence $\{x(k)\}$ that is zero for $k < k_0$, then $x(k_0)$ can be evaluated using the result:

$$x(k_0) = \lim_{z\to\infty} z^{k_0}X(z).$$

(Z7) **Final Value Theorem**: If $\mathcal{Z}[\{x(k)\}] = X(z)$ and $\lim\limits_{k\to\infty} x(k)$ exists, then

$$\lim_{k\to\infty} x(k) = \lim_{z\to 1}[z^{-1}(z-1)X(z)].$$

If $\{x(k)\}$ is a causal sequence, then

$$\lim_{k\to\infty} x(k) = \lim_{z\to 1}[(z-1)X_+(z)].$$

Example 2.2 Determine the z-transform of the sequence $\{p(k)\}$, where $p(k) = \begin{cases} 1, & 0 \le k \le 8, \\ 0, & \text{otherwise.} \end{cases}$

Solution: Note, for $k_0 \in \mathbb{N}$,

$$\zeta(k) - \zeta(k-k_0) = \begin{cases} 1, & 0 \le k \le k_0 - 1 \\ 0, & \text{otherwise.} \end{cases}$$

Thus, $p(k) = \zeta(k) - \zeta(k-9)$ and so, using the linearity property (Z1) and the delay property (Z2),

$$P(z) = \mathcal{Z}[\{\zeta(k) - \zeta(k-9)\}] = \frac{z}{z-1} - z^{-9}\left(\frac{z}{z-1}\right) = \frac{z(1-z^{-9})}{z-1}.$$

□

Example 2.3 Find the z-transform of the sequences

a) $\{e^{ak}\sin(\omega k)\zeta(k)\}$ b) $\{e^{ak}\delta(k-k_0)\}$ c) $\left\{\sin\left(\frac{\pi}{2}k\right)\zeta(k) * \left(\frac{1}{4}\right)\zeta(k)\right\}$,

where $a \in \mathbb{R}$ and $k_0 \in \mathbb{N}$.

Solution:

a) From Table B.1 (Appendix B), $\mathcal{Z}[\{\sin(\omega k)\zeta(k)\}] = \dfrac{z\sin(\omega)}{z^2 - 2z\cos(\omega) + 1}$.
Using the multiplication by an exponential property of z-transforms (Z3),

$$\begin{aligned}\mathcal{Z}\left[\left\{e^{ak}\sin(\omega k)\zeta(k)\right\}\right] &= \mathcal{Z}\left[\left\{(e^a)^k\sin(\omega k)\zeta(k)\right\}\right] \\ &= \frac{\left(\frac{z}{e^a}\right)\sin(\omega)}{\left(\frac{z}{e^a}\right)^2 - 2\left(\frac{z}{e^a}\right)\cos(\omega) + 1} = \frac{ze^a\sin(\omega)}{z^2 - 2ze^a\cos(\omega) + e^{2a}}.\end{aligned}$$

b) Using Table B.1 (Appendix B), $\mathcal{Z}\left[\{\delta(k)\}\right] = 1$ and so, in view of the delay property (Z2), $\mathcal{Z}\left[\{\delta(k-k_0)\}\right] = z^{-k_0}$. Hence, using the multiplication by an exponential property (Z3),

$$\mathcal{Z}\left[\left\{e^{ak}\delta(k-k_0)\right\}\right] = \left(\frac{z}{e^a}\right)^{-k_0} = e^{ak_0}z^{-k_0}.$$

c) Using Table B.1 (Appendix B) and the convolution property (Z5),

$$\mathcal{Z}\left[\left\{\sin\left(\tfrac{\pi}{2}k\right)\zeta(k) * \left(\tfrac{1}{4}\right)\zeta(k)\right\}\right] = \frac{z}{z^2+1} \times \frac{z}{z-\frac{1}{4}} = \frac{z^2}{(z^2+1)(z-\frac{1}{4})}.$$

□

2.3.2 The inverse z-transform

If $X(z) = \sum_{k=-\infty}^{\infty} x(k)z^{-k}$ converges to an analytic function, with region of convergence R, then

$$\{x(k)\} = \mathcal{Z}^{-1}\left[X(z)\right] = \frac{1}{2\pi i}\oint_C X(z)z^{k-1}\,\mathrm{d}z,$$

where C is a closed contour which lies within the region of convergence R. Some inversion methods are considered and described in Example 2.4.

Example 2.4 Find the inverse transform of $X(z) = \dfrac{z}{z^2-3z+2}$ with region of convergence:

a) $\{z \in \mathbb{Z} : |z| > 2\}$ b) $\{z \in \mathbb{Z} : |z| < 1\}$ c) $\{z \in \mathbb{Z} : 1 < |z| < 2\}$.

Solution:

1. **Residue method**: If the singularities of $X(z)$ are poles of finite order, then $\{x(k)\} = \mathcal{Z}^{-1}\left[X(z)\right]$ can be found using:

$$\mathcal{Z}^{-1}\left[X(z)\right] = \begin{cases} \displaystyle\sum_{\substack{\text{all poles of } X(z)z^{k-1} \\ \text{inside } C}} [\text{residue of } X(z)z^{k-1}], & k \geq 0 \\ \displaystyle -\sum_{\substack{\text{all poles of } X(z)z^{k-1} \\ \text{outside } C}} [\text{residue of } X(z)z^{k-1}], & k < 0. \end{cases}$$

Remarks 2.2

(i) *If* $X(z)z^{k-1}$ *has a simple pole at* $z = a$ *then the residue is*

$$\left[(z-a)X(z)z^{k-1}\right]_{z=a}.$$

(ii) *More generally, if* $X(z)z^{k-1}$ *has a pole of order* m *at* $z = a$ *then the residue is*

$$\left[\frac{1}{(m-1)!}\frac{\mathrm{d}^{m-1}}{\mathrm{d}z^{m-1}}\left\{(z-a)^m X(z)z^{k-1}\right\}\right]_{z=a}.$$

Now $X(z)z^{k-1} = \dfrac{z \times z^{k-1}}{z^2 - 3z + 2} = \dfrac{z^k}{(z-1)(z-2)}$ which has simple poles at $z = 1$ and $z = 2$ when $k \geq 0$, whilst for $k < 0$ there are poles at $z = 1,\ z = 2$ and $z = 0$.

a) Consider $k \geq 0$.

$$\begin{aligned}\mathcal{Z}^{-1}\left[X(z)\right] &= \left\{\left[(z-1)\frac{z^k}{(z-1)(z-2)}\right]_{z=1}\right.\\ &\quad \left. + \left[(z-2)\frac{z^k}{(z-1)(z-2)}\right]_{z=2}\right\}\\ &= \{-1 + 2^k\}.\end{aligned}$$

For $k < 0$, there are no poles outside C and so $\mathcal{Z}^{-1}\left[X(z)\right] = 0$. Hence, $\{x(k)\}$ is the sequence $\{(-1 + 2^k)\zeta(k)\}$.

b) Consider $k \geq 0$.
There are no poles inside C and therefore $\mathcal{Z}^{-1}\left[X(z)\right] = 0$.
For $k < 0$, the pole at $z = 0$ lies inside C, whilst the poles at $z = 1$ and $z = 2$ lie outside C. Therefore,

$$\begin{aligned}x(k) &= -\left[(z-1)\frac{z^k}{(z-1)(z-2)}\right]_{z=1}\\ &\quad + \left[(z-2)\frac{z^k}{(z-1)(z-2)}\right]_{z=2}\\ &= 1 - 2^k.\end{aligned}$$

Thus,

$$x(k) = \begin{cases} 1 - 2^k, & k < 0 \\ 0, & k \geq 0 \end{cases} = (1 - 2^k)\zeta(-k-1).$$

c) If $k \geq 0$, only the pole at $z = 1$ lies inside C. Hence,

$$\mathcal{Z}^{-1}\left[X(z)\right] = \left\{ \left[(z-1)\frac{z^k}{(z-1)(z-2)}\right]_{z=1} \right\} = \{-1\}.$$

When $k < 0$ there is one pole at $z = 2$ lying outside C. Therefore,

$$\mathcal{Z}^{-1}\left[X(z)\right] = \left\{ - \left[(z-2)\frac{z^k}{(z-1)(z-2)}\right]_{z=2} \right\} = \{-2^k\} \ .$$

Hence,

$$x(k) = \begin{cases} -2^k, & k < 0 \\ -1, & k \geq 0, \end{cases}$$

that is $x(k) = -2^k\zeta(-k-1) - \zeta(k)$.

2. **Partial fractions** :

$$X(z) = \frac{z}{(z-1)(z-2)} = z\left\{\frac{1}{z-2} - \frac{1}{z-1}\right\} = \frac{z}{z-2} - \frac{z}{z-1}$$

a) $|z| > 2$. Using Table B.1 (Appendix B),
$x(k) = 2^k\zeta(k) - 1^k\zeta(k) = (2^k - 1)\zeta(k)$.

b) $|z| < 1$. Using Table B.1 (Appendix B),
$x(k) = -2^k\zeta(-k-1) - \left(-1^k\zeta(-k-1)\right) = (1-2^k)\zeta(-k-1)$.

c) $1 < |z| < 2$, i.e. $|z| > 1$ and $|z| < 2$. Since $z = 1$ lies outside the region $|z| > 1$, we include the term $1^k\zeta(k)$. Also, since $z = 2$ lies outside the region $|z| < 2$, we include the term $-2^k\zeta(-k-1)$.
Thus, $x(k) = -2^k\zeta(-k-1) - \zeta(k)$.

□

Remarks 2.3

(i) *An inversion method based on convolution can also be used for this example. However, it is not discussed here.*

(ii) *A division method can be used for signals whose z-transform is a quotient of polynomials and which exist only for $k \geq 0$ or $k < 0$, but not both.*

Example 2.5 With the aid of MATLAB®, find the inverse transform of $X(z) = \dfrac{3}{4z^2 - 4z - 3}$ with region of convergence:

a) $\left\{z \in \mathbb{Z} : |z| > \frac{3}{2}\right\}$ b) $\left\{z \in \mathbb{Z} : \frac{1}{2} < |z| < \frac{3}{2}\right\}$ c) $\left\{z \in \mathbb{Z} : |z| < \frac{1}{2}\right\}$.

Solution: Firstly, $X(z)$ is expressed in powers of z^{-1}, i.e.

$$X(z) = \frac{3z^{-2}}{4 - 4z^{-1} - 3z^{-2}} = \frac{b(z)}{a(z)}.$$

The MATLAB® command `[r,p,k]=residuez(b,a)` evaluates $\dfrac{b(z)}{a(z)}$ in the form

$$\frac{r(1)}{1 - p(1)z^{-1}} + \ldots + \frac{r(n)}{1 - p(n)z^{-1}} + k(1) + k(2)z^{-1} + \ldots$$

and results in

$$X(z) = \frac{0.25}{1 - 1.5z^{-1}} + \frac{0.75}{1 + 0.5z^{-1}} - 1 = \frac{0.25z}{z - 1.5} + \frac{0.75z}{z + 0.5} - 1.$$

Hence,

$$X(z)z^{k-1} = \frac{0.25z^k}{z - 1.5} + \frac{0.75z^k}{z + 0.5} - z^{k-1}.$$

a) $k < 0$: Poles are $z = 1.5, \ -0.5, \ 0$. Since no poles lie outside C, $x(k) = 0$.
$k = 0$: Poles are $z = 1.5, \ -0.5, \ 0$.
All poles lie inside C, and so $x(k) = 0.25(1.5)^k + 0.75(-0.5)^k - 1$.
$k > 0$: Poles are $z = 1.5, \ -0.5$.
All poles lie inside C, and so $x(k) = 0.25(1.5)^k + 0.75(-0.5)^k$.
Therefore,

$$\begin{aligned} x(k) &= \begin{cases} 0, & k < 0, \\ 0.25(1.5)^k + 0.75(-0.5)^k - 1, & k = 0, \\ 0.25(1.5)^k + 0.75(-0.5)^k, & k > 0 \end{cases} \\ &= [0.25(1.5)^k + 0.75(-0.5)^k]\zeta(k) - \delta(k). \end{aligned}$$

b) $k < 0$: Since only the pole at $z = 1.5$ lies outside C, $x(k) = -0.25(1.5)^k$.
$k = 0$: Since only the poles at $z = 0,\ -0.5$ lie inside C,
$x(k) = 0.75(-0.5)^k - 1$.
$k > 0$: Since only the pole at $z = -0.5$ lies inside C, $x(k) = 0.75(-0.5)^k$.
Therefore,

$$\begin{aligned} x(k) &= \begin{cases} -0.25(1.5)^k, & k < 0, \\ 0.75(-0.5)^k - 1, & k = 0, \\ 0.75(-0.5)^k, & k > 0, \end{cases} \\ &= -0.25(1.5)^k\zeta(-k-1) + 0.75(-0.5)^k\zeta(k) - \delta(k). \end{aligned}$$

c) $k < 0$: Since the poles at $z = 1.5,\ -0.5$ lie outside C,
$x(k) = -0.25(1.5)^k - 0.75(-0.5)^k$.
$k = 0$: Since only the poles at $z = 0$ lies inside C, $x(k) = -1$.
$k > 0$: No poles lie inside C.
Therefore,

$$\begin{aligned} x(k) &= \begin{cases} -0.25(1.5)^k - 0.75(-0.5)^k, & k < 0, \\ -1, & k = 0, \\ 0, & k > 0, \end{cases} \\ &= -[0.25(1.5)^k + 0.75(-0.5)^k]\zeta(-k-1) - \delta(k). \end{aligned}$$

□

2.4 Solution of linear difference equations with constant coefficients

The z-transform may be used to solve linear difference equation with constant coefficients. The method is illustrated in Example 2.6.

Example 2.6 Obtain a closed form solution to the linear, second order difference equation

$$x(k+2) - 2x(k+1) + x(k) = 1, \qquad k \geq 0, \quad k \in \mathbb{Z}\ ,$$

given $x(0) = 0$ and $x(1) = -\frac{1}{2}$.

Solution: The first step is to take the (unilateral) z-transform of the difference equation. The resulting equation is then solved for $X_+(z)$. Finally, by inversion, the sequence $\{x(k)\}$ can be found. By the linearity property (Z1),

$$\mathcal{Z}\left[\{x(k+2)\}\right] - 2\mathcal{Z}\left[\{x(k+1)\}\right] + \mathcal{Z}\left[\{x(k)\}\right] = \mathcal{Z}\left[\{1\}\right].$$

Hence, using the advance property (Z2) and Table B.1 (Appendix B),

$$\left[z^2X_+(z) - z^2(x(0) + x(1)z^{-1})\right] - 2\left[zX_+(z) - zx(0)\right] + X_+(z) = \frac{z}{z-1}$$

and so

$$(z^2 - 2z + 1)X_+(z) = \frac{z}{z-1} + zx(1) + (z^2 - 2z)x(0).$$

Using $x(0) = 0$ and $x(1) = -\frac{1}{2}$,

$$(z-1)^2X_+(z) = \frac{z}{z-1} - \frac{z}{2}$$

and therefore

$$X_+(z) = \frac{z}{(z-1)^2}\left(\frac{1}{z-1} - \frac{1}{2}\right) = \frac{z(3-z)}{2(z-1)^3}.$$

Now $z^{k-1}X_+(z) = \dfrac{z^k(3-z)}{2(z-1)^3}$ has a pole of order 3 at $z = 1$. The residue at $z = 1$ is

$$\begin{aligned}\left[\frac{1}{2!}\frac{\mathrm{d}^2}{\mathrm{d}z^2}\left(\frac{z^k(3-z)}{2}\right)\right]_{z=1} &= \frac{1}{4}\left[\frac{\mathrm{d}}{\mathrm{d}z}\left(kz^{k-1}[3-z] + z^k(-1)\right)\right]_{z=1}\\ &= \frac{1}{4}\left[k(k-1)z^{k-2}(3-z) - 2kz^{k-1}\right]_{z=1}\\ &= \left\{\frac{1}{2}k(k-2)\right\}.\end{aligned}$$

Thus $x(k) = \dfrac{1}{2}k(k-2),\ \ k \geq 0.$

□

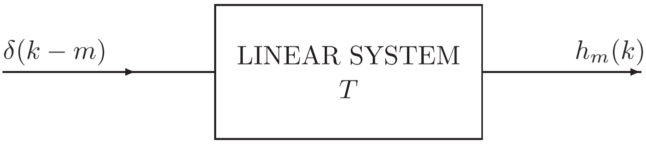
2.5 Discrete-time system responses

In this section, discrete-time system responses to inputs, such as a unit step function, an impulse function and sinusoids, are investigated.

2.5.1 The impulse response

The **impulse response** of a discrete-time system is defined to be the output that results when the input is the unit impulse sequence, $\{\delta(k)\}$, and the system is initially quiescent.

Any linear, discrete system can be completely characterized by its unit impulse response: suppose $h_m(k)$ be the response of a linear system to $\delta(k-m)$.



The response of the system to the signal $x(k)$ is

$$\begin{aligned} y(k) = Tx(k) = T\left[\sum_{m=-\infty}^{\infty} x(m)\delta(k-m)\right] & \quad \text{(from (1.1))} \\ = \sum_{m=-\infty}^{\infty} x(m)T[\delta(k-m)], & \quad \text{since the system is linear,} \\ = \sum_{m=-\infty}^{\infty} x(m)h_m(k) & \end{aligned} \tag{2.3}$$

Assume that the linear, discrete system is time-invariant, then, if $h(k)$ is the response to $\delta(k)$, $h(k-m)$ is the response to $\delta(k-m)$ and so (2.3) may be written as

$$y(k) = \sum_{m=-\infty}^{\infty} x(m)h(k-m). \tag{2.4}$$

Thus, any linear, time-invariant, discrete system is completely characterized by its unit impulse (sample) response, $h(k)$, in the sense that, given the input sequence $\{x(k)\}$, the output sequence $\{y(k)\}$ can be determined if $\{h(k)\}$ is known.

Remark 2.4 *Similarly, a continuous-time system, which is linear and time-invariant, is characterized by its* impulse response $h(t)$. *Using the definition of* *convolution, then, for input $x(t)$, the corresponding output is given by*

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\,\mathrm{d}\tau = (x*h)(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)\,\mathrm{d}\tau = (h*x)(t).$$

2.5.2 Causality, FIR/IIR systems/filters and the system function

Causality and FIR/IIR systems

It follows from Definition 1.5 that a discrete-time system, which is linear and time-invariant, is causal iff

$$\boxed{h(k) = 0 \quad \text{for } k < 0.} \tag{2.5}$$

Note that for a linear, time-invariant, causal system, $h(k-m) = 0$ for $k < m$ and, therefore, (2.4) reduces to

$$y(k) = \sum_{m=-\infty}^{k} x(m)h(k-m) \overset{(n=k-m)}{=} \sum_{n=0}^{\infty} h(n)x(k-n). \tag{2.6}$$

In view of (2.5), the concept of causality is often applied to signals (see Definition 2.1 for discrete-time signals).

Definition 2.1 *A discrete-time signal $\{x(k)\}$ is* ***causal*** *iff $x(k) = 0$ for $k < 0$.*

Thus, for example, $\{\zeta(k)\}$ is a causal sequence. The sequence $\{\zeta(-k-1)\}$ is sometimes known as an *anti-causal sequence*, since $\zeta(-k-1) = 0$ for $k \geq 0$.

Note that, if an input to a discrete-time system, say $\{x(k)\}$, is a causal signal, then (2.6) can be replaced by

$$y(k) = \sum_{m=0}^{k} x(m)h(k-m) = \sum_{m=0}^{k} h(m)x(k-m).$$

In this case, the impulse (or sample) response has only a finite number of non-zero values. In general, a linear, time-invariant system may have a impulse response that is of finite duration or infinite duration. A system with impulse response of finite duration is referred to as a *nonrecursive* or *finite impulse response* (FIR) system, and, if the impulse response is of infinite duration, the system is called a *recursive* or *infinite impulse response* (IIR) system. In the literature, FIR filters are sometimes known as *moving average* (MA) or *transversal* filters, whilst IIR filters are called *autoregressive moving average* (ARMA) filters.

Remark 2.5 *Strictly speaking, a difference equation realization of a MA filter has the form:*

$$y(k) = \frac{1}{M_1 + M_2 + 1} \sum_{m=-M_1}^{M_2} x(k-m).$$

Suppose $a_i = 0$ for all $i \geq 1$ in (2.2), then the linear system (2.2) is a FIR system, since

$$y(k) = \sum_{m=0}^{M} b_m x(k-m) \tag{2.7}$$

and, comparing (2.7) with (2.6),

$$h(m) = \begin{cases} b_m, & m = 0, 1, \ldots, M \\ 0, & \text{otherwise.} \end{cases}$$

In contrast, if $N \geq 1$ then (2.2) is either an IIR or FIR system.

System function

A discrete-time, linear, time-invariant system is completely characterized by its unit impulse response and can be represented by (2.3). Thus, assuming zero initial conditions and taking the z-transform of (2.3), one obtains, using z-transform convolution property (Z5),

$$Y(z) = \mathcal{Z}\left[\{(h * x)(k)\}\right] = H(z)X(z),$$

where

$$\boxed{H(z) = \sum_{k=-\infty}^{\infty} h(k)z^{-k} = \mathcal{Z}\left[\{h(k)\}\right].}$$

The transform

$$\boxed{H(z) = \frac{Y(z)}{X(z)}}$$

is called the *system* (or *transfer*) *function*. If $x(k) = \delta(k)$ then $X(z) = 1$ and, therefore, $Y(z) = H(z)X(z) = H(z)$. Hence, if the impulse response is denoted by $\{y_\delta(k)\}$, then

$$\boxed{\{y_\delta(k)\} = \mathcal{Z}^{-1}\left[H(z)\right].}$$

2.5.3 The step response

The **step response** for a discrete-time linear system is defined to be the output that results when the input is a unit step sequence, $\{\zeta(k)\}$, assuming the system is initially quiescent. Denoting this response by $\{y_\zeta(k)\}$, then

$$\boxed{\{y_\zeta(k)\} = \mathcal{Z}^{-1}\left[H(z)\,\frac{z}{z-1}\right].}$$

Since $\{\delta(k)\} = \{\zeta(k)\} - \{\zeta(k-1)\}$, the impulse response can be obtained from the step response using the result:

$$\boxed{\{y_\delta(k)\} = \{y_\zeta(k) - y_\zeta(k-1)\}.} \tag{2.8}$$

Example 2.7 A causal, linear, time-invariant system is modelled by the difference equation

$$y(k) = y(k-1) - 0.25y(k-2) + x(k) - x(k-1).$$

a) Determine the system function $H(z)$ and state the region of convergence.

b) Evaluate the unit step and impulse responses. In addition, generate and plot these responses using MATLAB®.

Solution:

a) Taking the z-transform of the difference equation,

$$Y(z) = z^{-1}Y(z) - 0.25z^{-2}Y(z) + X(z) - z^{-1}X(z),$$

or

$$\left(1 - z^{-1} + 0.25z^{-2}\right)Y(z) = \left(1 - z^{-1}\right)X(z).$$

Hence, the system function is

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 - z^{-1}}{1 - z^{-1} + 0.25z^{-2}} = \frac{z(z-1)}{z^2 - z + 0.25} = \frac{z(z-1)}{(z - \frac{1}{2})^2}.$$

Since the system is causal, the region of convergence is $\left\{z \in \mathbb{C} : |z| > \frac{1}{2}\right\}$.

b) Let $Y_\zeta(z)$ denote the z-transform of the unit step response, then

$$Y_\zeta(z) = H(z)\frac{z}{z-1} = \frac{z^2}{(z - \frac{1}{2})^2}.$$

Using partial fractions,

$$\frac{Y_\zeta(z)}{z} = \frac{z}{(z - \frac{1}{2})^2} = \frac{1}{z - \frac{1}{2}} + \frac{\frac{1}{2}}{(z - \frac{1}{2})^2}$$

and so

$$Y_\zeta(z) = \frac{z}{z - \frac{1}{2}} + \frac{\frac{1}{2}z}{(z - \frac{1}{2})^2}.$$

Taking inverse z-transforms,

$$y_\zeta(k) = \left(\tfrac{1}{2}\right)^k \zeta(k) + \tfrac{1}{2}k\left(\tfrac{1}{2}\right)^{k-1} \zeta(k) = (1+k)\left(\tfrac{1}{2}\right)^k \zeta(k).$$

The impulse response can be found using (2.8).

$$y_\delta(k) = y_\zeta(k) - y_\zeta(k-1) = (1+k)\left(\tfrac{1}{2}\right)^k \zeta(k) - k\left(\tfrac{1}{2}\right)^{k-1} \zeta(k-1).$$

Since $\delta(k) = \zeta(k) - \zeta(k-1)$, it follows that

$$y_\delta(k) = (1+k)\left(\tfrac{1}{2}\right)^k \zeta(k) - 2k\left(\tfrac{1}{2}\right)^k [\zeta(k) - \delta(k)],$$

but $2k\left(\frac{1}{2}\right)^k \delta(k) = 2k\left(\frac{1}{2}\right)^k\Big|_{k=0} \delta(k) = 0$ and so

$$y_\delta(k) = (1+k)\left(\tfrac{1}{2}\right)^k \zeta(k) - 2k\left(\tfrac{1}{2}\right)^k \zeta(k) = (1-k)\left(\tfrac{1}{2}\right)^k \zeta(k).$$

The unit step and impulse responses for the filter, over 25 samples, were generated using the MATLAB® commands:

```
b=[1 -1 0]; a=[1 -1 0.25];
subplot(2,1,1); impz(b,a,25);
subplot(2,1,2); stepz(b,a,25);
```

and are illustrated in Figure 2.1. The responses can also be displayed, using the Filter Visualization Tool (fvtool), with the MATLAB® command `fvtool(b,a)` and then clicking on the appropriate button.

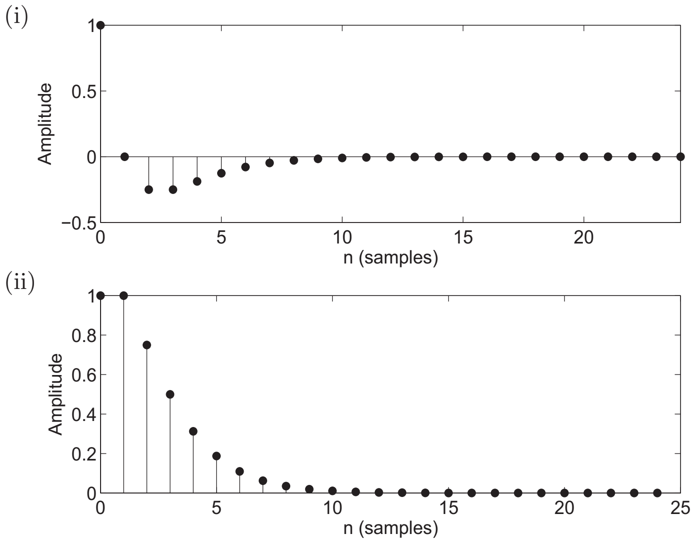


Figure 2.1: (i) Unit impulse response. (ii) Unit step response.

□

2.6 A voiced/unvoiced speech production model for speech processing

In this section, speech processing is considered and an appropriate system function is suggested for modelling the vocal and nasal tracts as shown in Figure 2.2 (for more details, see [Rabiner and Schafer, 1978, Deller et al., 2000]).

Other useful references on speech processing are the following: [Flanagan, 1972, Quatieri, 2001]

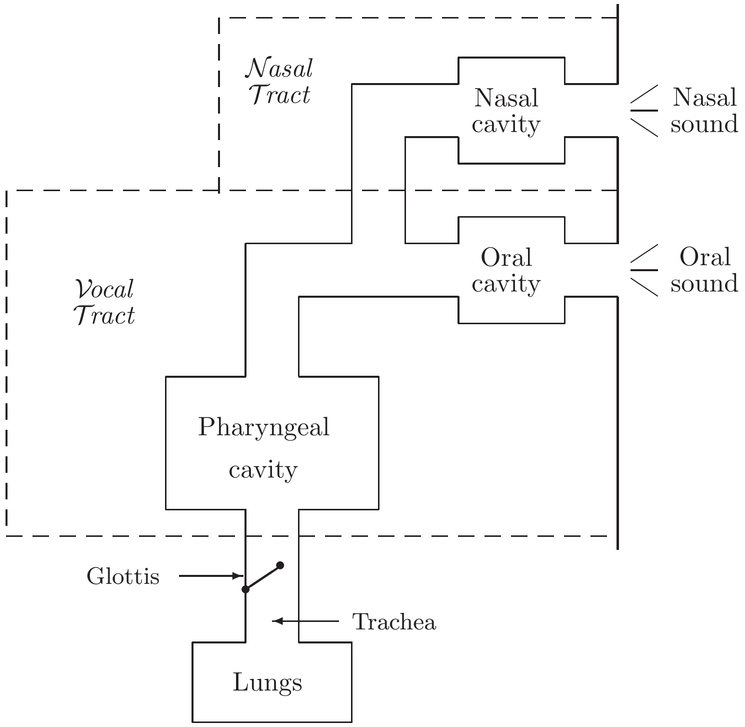


Figure 2.2: Schematic diagram of the vocal and nasal tracts.

2.6.1 Physiological aspects of speech production

A speech signal is the result of an acoustic pressure wave that is formed when air is forced through anatomical structures in the human speech production system. A crude physiological model of the human speech production system consists of the following elements: lungs, trachea (windpipe), glottis (valve), pharyngeal cavity (throat), oral (mouth) and nasal (nose) cavities, illustrated in Figure 2.2. As shown in the schematic illustration in Figure 2.2, the subsystem representing the nasal cavity is often known as the **nasal tract model**, whilst the pharyngeal and oral cavities are said to comprise the **vocal tract model**.

The vocal tract produces a sound when the lungs contract, forcing air up through an air-passage known as the *trachea*. The increase in air pressure causes the *glottis*, a valve, to burst open, momentarily, before quickly closing. When this is repeatedly performed, a very nearly periodic waveform is produced by this mechanism. The resulting flow of air is then perturbed by the vocal tract (or vocal and nasal tracts) to produce a sound. Some factors that can affect the waveform are : tension in the vocal chords, lung pressure, shape of the vocal cavities. The mouth, nose and throat are the primary resonating cavities. Their resonant characteristics can be changed (thereby altering the speech waveform) by moving the lips, tongue, jaw, and soft palate at the back of the mouth. Speech sounds can be classified into a number of distinct classes. Three important classes are:

a) *voiced sounds*: produced when quasi-periodic pulses of air excite the vocal tract (for example, ‘u’, ‘i’, ‘d’, ‘w’, etc.);

b) *unvoiced sounds* (or *fricatives*): generated by forming a constriction in the vocal tract and forcing air through the constriction. This creates a noise source which excites the vocal tract (for example, ‘sh’, ‘f’, ‘s’);

c) *plosive sounds*: obtained by making a complete closure of the vocal tract (with lips), building up pressure behind the closure, and abruptly releasing it (for example, ‘p’, ‘b’).

Resonances in the vocal tract are characterized by resonant frequencies in the vocal tract spectrum, which are called *formant* frequencies (or simply *formants*); the frequency bandwidth for human speech production being approximately 7 or 8 kHz. Tension of the vocal chords can affect *pitch*, namely an increase in tension causes the chords to vibrate at a higher frequency. The time between successive openings of the glottis valve, producing the quasi-periodic pulses of air, is called the ‘fundamental period’. By convention, see [Deller et al., 2000] for a fuller discussion, the fundamental period is often referred to as the ‘pitch period’. Pitch period and formant estimation are two of the most important problems in speech processing.

2.6.2 Lossless tube model [optional]

Due to soft tissue surrounding the vocal tract, sound waves in the vocal tract primarily propagate in one direction, namely along the vocal tract. Assuming there is no loss of sound through the boundary walls of the vocal tract, an acoustic model for speech production usually consists of a concatenation of lossless acoustic tubes with constant cross-sectional areas $\{A_k\}$, which are chosen to approximate the vocal tract 'area' function. A diagrammatic representation is shown in Figure 2.3. Considering two adjacent tubes, namely tube k and tube $k+1$, it is well known (see [Deller et al., 2000] and [Rabiner and Schafer, 1978]) that the 'volume' velocity, $v_k(t,x)$, in the k^{th} tube is given by

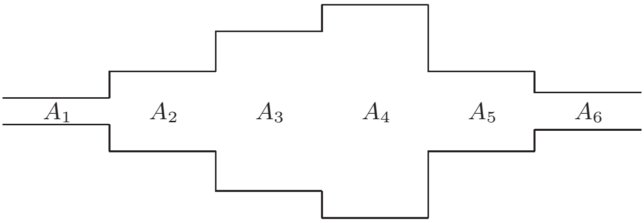


Figure 2.3: Acoustic lossless tube model.

$$v_k(t,x) = v_k^+\left(t - \frac{x}{c}\right) - v_k^-\left(t + \frac{x}{c}\right) ,$$

where t denotes time, x is the distance measured from the left-hand end of the k^{th} tube, c (assumed to be constant) is the speed of sound in air, and $v_k^+(\cdot)$ and $v_k^-(\cdot)$ denote the respective velocities of the travelling waves in the positive and negative directions in the k^{th} tube; moreover, by applying boundary conditions at the junction of tube k and tube $k+1$, the following equations hold:

$$\begin{aligned} v_{k+1}^+(t) &= (1+r_k)v_k^+(t-\tau_k) + r_k v_{k+1}^-(t) \\ v_k^-(t+\tau_k) &= -r_k v_k^+(t-\tau_k) + (1-r_k)v_{k+1}^-(t) , \end{aligned}$$

where τ_k denotes the delay incurred in travelling the length of the k^{th} tube and r_k (known as the **reflection coefficient**) is the proportion of the forward travelling wave in tube k, $v_k^+(t-\tau_k)$, that is reflected into tube $k+1$. At the boundary of the junction of the tubes the value of the reflection coefficient, r_k, is determined by

$$r_k = \frac{A_{k+1} - A_k}{A_{k+1} + A_k}$$

and it can be shown (see Exercise 2.20) that

$$-1 \leq r_k \leq 1 \ .$$

A signal flow graph representation is given in Figure 2.4. It is noted that the signal flow graph contains only additions, multiplications and delays and, therefore, a discrete-time model, utilizing these operations, can easily be implemented.

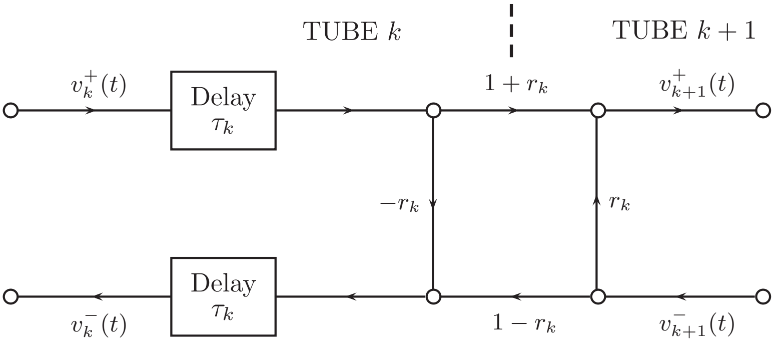


Figure 2.4: Signal flow diagram for the junction of tube k and tube $k+1$.

Assuming N tubes, in the model, are equal in length and each delay is half the sampling delay, i.e. $\tau_k = T/2$, then the z-transform system function can be shown (see, for example, Chapter 3, §3.2, in [Deller et al., 2000] and Chapter 3, §3.3.4, in [Rabiner and Schafer, 1978] for more details) to have the form

$$H(z) = \frac{Gz^{-\frac{N}{2}}(1+r_1)(1+r_2)\ldots(1+r_N)}{1-\sum_{k=1}^{N} a_k z^{-k}} = \frac{Gz^{-\frac{N}{2}}\prod_{k=1}^{N}(1+r_k)}{1-\sum_{k=1}^{N} a_k z^{-k}} \ ,$$

where the gain term, G, is constant. The poles of $H(z)$ define the formants (resonances) of the lossless tube model.

2.6.3 Digital speech production model

In order to analyse a speech (continuous-time) signal, the signal is sampled to obtain a discrete-time signal, $\{s(m)\}$. It is assumed that speech production can be modelled by a linear time-varying filter that is excited by an input, $\{x(m)\}$, consisting of a quasi-periodic train of impulses, in the case of voiced sounds, or a random noise source, in the case of unvoiced sounds. Usually during excitation, vocal and nasal tract properties of a speech signal change relatively slowly with time and, for most speech signals, we may assume that these properties remain fixed over short time periods, such as 10-20 msec. Thus, in these short time periods, speech production is assumed to be modelled by *time-invariant* filters. A linear time-invariant model for speech production is usually taken to be of the form:

$$S(z) = G(z)V(z)L(z)X(z),$$

where $S(z) = \mathscr{Z}\left[\{s(m)\}\right]$, $X(z) = \mathscr{Z}\left[\{x(m)\}\right]$, $G(z)$ is the glottal shaping model, $V(z)$ a , and $L(z)$ a lip radiation model. Experimentally, it is found that a reasonably good model for the vocal tract system function has the form:

$$H(z) = \frac{S(z)}{X(z)} = G(z)V(z)L(z) \approx \frac{G_s\left[1 - \displaystyle\sum_{j=1}^{q} b_j z^{-j}\right]}{\left[1 - \displaystyle\sum_{k=1}^{p} a_k z^{-k}\right]},$$

where the constant $G_s \in \mathbb{R}^+$ is known as the *speech gain parameter*. For non-nasal, voiced sounds an *all-pole* filter (namely an AR filter) is a good approximation for the system function. Thus, a further simplification of the speech production model (for non-nasal, voiced sounds) is:

$$S(z) = G_s \frac{X(z)}{A(z)},$$

where

$$A(z) \stackrel{\text{def}}{=} 1 - \sum_{k=1}^{p} a_k z^{-k}.$$

The parameters of the speech production model, namely $\{a_k;\ k = 1,\ 2, \ldots p\}$ and G_s may be estimated for a short segment of speech, for example 20 msec or 260 samples when sampled at 13 kHz (using the analysis model illustrated in Figure 2.5). Moreover, an analysis can be performed to identify the type of excitation, i.e. quasi-periodic train of impulses or a random noise source, and, in the case of voiced speech, compute an approximate value of the pitch period. This information can be encoded and used for the synthesis of speech (see Figure 2.6).

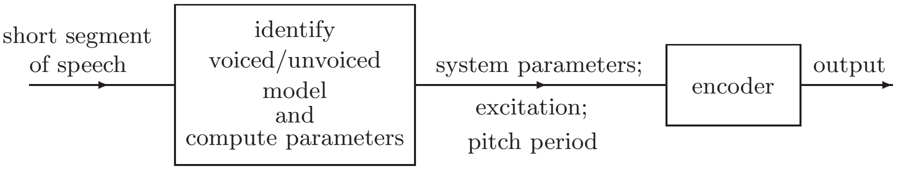


Figure 2.5: Analysis model.

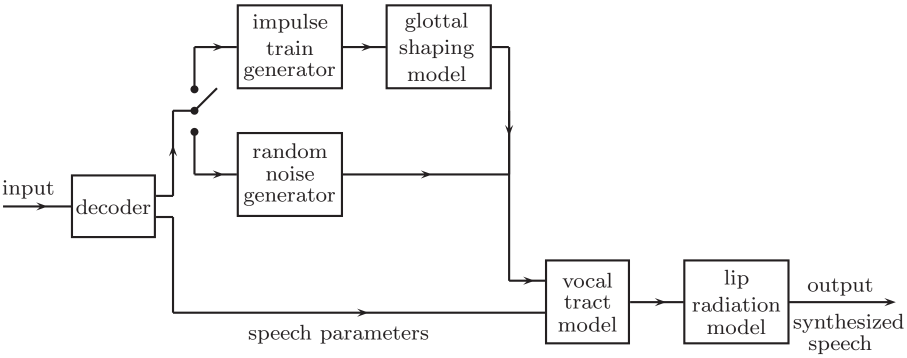


Figure 2.6: Synthesis model.

2.7 System function and BIBO stability

Another important property of systems is that of *stability*.

Definition 2.2 *A system is* ***stable*** (*or* ***bounded-input / bounded-output (BIBO) stable***) *iff, for any bounded input, the output is bounded at all times given zero initial conditions.*

Theorem 2.1 *Discrete-time systems, which are linear and time-invariant, are stable iff*

$$\sum_{k=-\infty}^{\infty} |h(k)| < \infty;$$

whereas linear, time-invariant, continuous-time systems are stable iff

$$\int_{-\infty}^{\infty} |h(t)|\, \mathrm{d}t < \infty.$$

Clearly, in view of Theorem 2.1, all linear, time-invariant, FIR systems and MA processes are BIBO stable.

A useful stability result can be obtained in terms of the system function $H(z)$, introduced in §2.5.1.

Theorem 2.2 *A linear, time-invariant, discrete-time system, with system function $H(z)$ and zero initial conditions, is BIBO stable iff the region of convergence for $H(z)$ contains the unit circle centered at the origin in the z plane.*

Therefore, this theorem can be used to determine stability for a given $H(z)$ without obtaining the impulse response or checking outputs for all bounded input signals.

Remark 2.6 *FIR filters are always stable.*

If a linear, time-invariant, discrete-time system is causal, $H(z)$ converges everywhere in the region $|z| > r$, where r is the largest radius of the circle containing the poles of $H(z)$. Hence, if the system is both stable and causal, all the poles of $H(z)$ must lie inside the unit circle.

Corollary 2.1 *A causal, linear, time-invariant, discrete-time system, with system function $H(z)$ and initially quiescent, is BIBO stable iff all the poles of $H(z)$ lie inside the unit circle centered at the origin in the z plane.*

Definition 2.3 *A linear, time-invariant, discrete-time system is said to be* ***minimum-phase*** *iff all the poles and all the zeros of the system function lie inside the unit circle $\{z \in \mathbb{Z} : |z| = 1\}$.*

2.8 Exercises

Exercise 2.1

Consider filters, modelled by the following input-output relationships:

a) $y(k) - y(k-2) + x(k) = 0$;

b) $y(k) = (-1)^k y(k-1) + 2^{-k} x(k-1)$;

c) $y(k+2) = x(k+2) - 5x(k)$;

where $\{x(k)\}$ denotes the input and $\{y(k)\}$ the corresponding output to the filter. Classify the filters as (i) moving average (MA), (ii) autoregressive (AR), (iii) autoregressive moving average (ARMA). Also, determine which of the filters are time-invariant.

Exercise 2.2 Determine the z-transform and its region of convergence for each of the following discrete signals $\{x(k)\}$:

a) $x(k) = \begin{cases} 0, & k < 0, \\ a^k, & 0 \leq k \leq 5, \\ 0, & k \geq 6; \end{cases}$

b) $x(k) = 3(-\frac{1}{2})^k \zeta(k) - 2(3)^k \zeta(-k-1)$;

c) $x(k) = \frac{1}{2}\delta(k) + \delta(k-1) - \frac{1}{3}\delta(k-2)$;

d) $x(k) = k\zeta(k-1)$.

Exercise 2.3 Given the signal $\{x(k)\}$, where

$$x(k) = \begin{cases} 0, & k < 0, \\ a^k, & 0 \leq k \leq 10, \\ 1 + a^k, & k \geq 11, \end{cases}$$

express $x(k)$ in terms of $\zeta(k)$ and, hence, use the delay property (Chapter 2, Z2) to find the z-transform of the signal $\{x(k)\}$, together with its region of convergence.

Exercise 2.4 Find $\mathcal{Z}\left[\{(x * y)(k)\}\right]$, when $x(k) = \delta(k) + \frac{1}{2}\zeta(k)$ and $y(k) = \delta(k) - 2\delta(k-2)$, using

(i) the convolution property (Chapter 1, CD4): if $y(k) = \delta(k-k_0)$, $k_0 \in \mathbb{Z}$, then $(x * y)(k) = x(k-k_0)$, and then determine the z-transform of the resulting sequence, using the delay property (Chapter 2, Z2);

(ii) the convolution z-transform property (Chapter 2, Z5).

Exercise 2.5 Find the z-transform of the signal $\{x(k)\}$, defined by $x(k) \overset{\text{def}}{=} (\frac{1}{2})^{k+2}\zeta(k)$, using the advance property (Chapter 2, Z2). Also, determine the region of convergence.

Exercise 2.6 Invert the following z-transforms of causal sequences.

a) $\dfrac{2z^2 - 3z}{z^2 - 3z + 2}$ b) $\dfrac{z^2 - z}{(z-4)(z-2)^2}$ c) $\dfrac{z-3}{z^2 - 3z + 2}$

Exercise 2.7 Solve the difference equation defined, for $k \geq 0$ ($k \in \mathbb{Z}$), by

$$6y(k+2) + 5y(k+1) - y(k) = 10 \ , \qquad y(0) = 0, \ y(1) = 1.$$

Exercise 2.8 Using the partial fraction expansion method, with the help of MATLAB® (if required), find the inverse z-transform of

$$X(z) = \frac{z(z^2 - 4z + 5)}{(z-1)(z-2)(z-3)}$$

for the following regions of convergence:
a) $\{z \in \mathbb{C} : 2 < |z| < 3\}$; b) $\{z \in \mathbb{C} : |z| > 3\}$; c) $\{z \in \mathbb{C} : |z| < 1\}$.

Exercise 2.9 The system function of a causal, linear, time-invariant AR process is

$$H(z) = z^2 / \left(z^2 - \tfrac{1}{4}\right).$$

a) Obtain a difference equation realisation for this filter.

b) Determine $\{y_\zeta(k)\}$ and $\{y_\delta(k)\}$ and comment whether the process is FIR or IIR, stating your reasons.

Exercise 2.10 If $X(z) = \mathscr{Z}[\{x(k)\}]$ and $X(z) = \dfrac{z(4-3z)}{z^2+4}$, $|z| > 2$, find $x(k)$.

Exercise 2.11 A linear, time-invariant, discrete system with input $x(k)$ and output $y(k)$ is characterized by its sample response: $h(k) = a^k\zeta(k)$ for $0 < a < 1$. Find the response $y(k)$ of the system to the input signal $x(k) = \zeta(k)$. Comment on the stability of the system.

Exercise 2.12 Consider a discrete system with unit sample response $h(k)$ given by $h(k) = 2^{-k}\zeta(k)$.

a) Find a difference equation realization of the system.

b) If the input to the system is $x(k) = 2\delta(k) + 4\delta(k-1) + 4\delta(k-2)$, calculate, using the MATLAB® command `filter`, $y(k)$ for $k = 0, 1, 2, 3, 4, 5$ and 6, assuming the system is initially quiescent.

c) Is the system (i) stable (ii) causal?

Exercise 2.13 A linear, time-invariant, discrete filter is characterized by its unit impulse response $h(k)$ given by $h(k) = 3(-\frac{1}{4})^k\zeta(k-1)$. Is the filter a) causal; b) stable; c) FIR or IIR? Give your reasons.

Exercise 2.14 For the system modelled by

$$50y(k+2) - 35y(k+1) + 6y(k) = 25[x(k+1) + x(k)], \quad k \in \mathbb{Z},$$

a) find the impulse and step responses to the system;

b) plot the impulse and step responses using MATLAB® for $k = 0, \ldots 20$.

Exercise 2.15 Find, using z-transforms, the unit step response of a discrete system with the unit impulse response function

$$h(k) = \begin{cases} 3^k, & k < 0, \\ 0.4^k, & k \geq 0. \end{cases}$$

Exercise 2.16 Given a causal system specified by its system function:

$$H(z) = \frac{z(z-1)}{(z-\frac{1}{2})(z+\frac{1}{4})},$$

a) find a difference equation realization of the system;

b) determine the impulse response of the system;

c) is the system minimum-phase?

Exercise 2.17 Suppose the unit step response of a linear, time-invariant, causal filter is

$$y_\zeta(k) = \left[\left(-\tfrac{1}{2}\right)^k + 6\left(\tfrac{3}{4}\right)^k\right]\zeta(k).$$

a) Find the system function, $H(z)$, for this filter.
Plot the poles and zeros of $H(z)$, using MATLAB®, and state the region of convergence for $H(z)$.

b) Explain why the filter is stable.

c) Is the filter minimum-phase?

d) Find the unit impulse response of the filter.

e) Obtain a difference equation realization for the filter.

Exercise 2.18 A stable filter has a zero at -2 and two simple poles at $\frac{1}{2}$ and $-\frac{3}{4}$. In addition, it is known that $y_\delta(1) = \frac{7}{4}$, where $\{y_\delta(k)\}$ denotes the response of the filter to the unit impulse sequence $\{\delta(k)\}$ with zero initial conditions.

a) Determine the system function and its region of convergence.

b) Is the filter causal?

c) Determine the step response of the filter and plot the step response, using MATLAB® , for $k = 0 \dots 24$ $(k \in \mathbb{Z})$.

Exercise 2.19 Investigate BIBO stability for a linear causal system with system function

$$H(z) = \frac{1}{z^2 + \alpha z - \alpha},$$

where α is a real parameter.

Exercise 2.20 [Optional] For two adjacent, lossless, acoustic tubes, tube k and tube $k+1$, with respective constant cross-sectional areas $A_k > 0$ and $A_{k+1} > 0$, the reflection coefficient r_k, for the junction of the two tubes, can be expressed as

$$r_k = \frac{\frac{A_{k+1}}{A_k} - 1}{\frac{A_{k+1}}{A_k} + 1} \qquad \text{or} \qquad r_k = \frac{1 - \frac{A_k}{A_{k+1}}}{1 + \frac{A_k}{A_{k+1}}} \; .$$

Using these results, show that $-1 \le r_k \le 1$.

Chapter 3

Frequency response, Fourier spectra and sampling

3.1 Introduction

In Chapter 2, the impulse and step responses were studied as solutions to a difference equation. This was, essentially, a study in the time-domain. In this chapter a response of a system is viewed in the frequency-domain, which leads to a function, of frequency, that allows one to estimate a steady state response of a system to sinusoids.

Since sampling is an important aspect of signal processing, this is investigated, via the Fourier transform and a study of a signal's spectra, and the well-known problem of aliasing is discussed. The spectrum of a signal is important and many books have been written on spectral analysis; for more details see [Kay, 1987] and [Marple, 1987], to name but a few. In addition, the discrete Fourier transform is introduced, for analysing digital signals, and some appropriate spectra are described. Since window functions are often used to filter digital signals in the time-domain, a number of common windows are described in this chapter and spectral properties of various windows are illustrated.

3.2 Frequency and steady-state response

Consider the response of a linear, time-invariant, discrete-time system to a complex exponential input of the form $x(k) = \exp(i\theta k)$, where θ represents angular frequency and i denotes $\sqrt{-1}$. Suppose the impulse response of the system is $\{h(k)\}$, then the output response is

$$y(k) = (h * x)(k) = \sum_{m=-\infty}^{\infty} h(m)x(k-m) = \sum_{m=-\infty}^{\infty} h(m)e^{i\theta(k-m)}.$$

Thus,

$$y(k) = e^{i\theta k} \sum_{m=-\infty}^{\infty} h(m)e^{-i\theta m};$$

that is the output signal is the original input signal, with $x(k) = e^{i\theta k}$, multiplied by a function which only depends on frequency θ.

Definition 3.1 *Let $\{h(k)\}$ denote the impulse response of a linear, BIBO stable, time-invariant, discrete-time system. Then*

$$\boxed{H(e^{i\theta}) \overset{\text{def}}{=} \sum_{m=-\infty}^{\infty} h(m)e^{-i\theta m}} \tag{3.1}$$

is called the ***frequency response*** *of the linear, time-invariant system.*

Therefore, the output can be written as

$$y(k) = e^{i\theta k}H(e^{i\theta}) = x(k)H(e^{i\theta})$$

and, hence, the magnitude of the output is the magnitude of the input multiplied by $|H(e^{i\theta})|$. Now $|H(e^{i\theta})|$ is known as the **gain spectrum** (or sometimes **amplitude response** or **magnitude response**) whilst $\arg[H(e^{i\theta})]$ is called the **phase-shift spectrum** (or **phase response**).
Some of the main properties of the frequency response are:

(FR1) $H(e^{i\theta})$ is periodic in θ with period 2π;

(FR2) $|H(e^{i\theta})|$ is an even function of θ and symmetrical about π;

(FR3) $\arg[H(e^{i\theta})]$ is an odd function of θ and antisymmetrical about π.

Remark 3.1 *$h(k)$ can be recovered from $H(e^{i\theta})$ by using*

$$\boxed{h(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{i\theta}) e^{i\theta k} \, \mathrm{d}\theta.} \tag{3.2}$$

Consider the response of a linear, time-invariant, causal, system with system function

$$H(z) = \frac{b(z)}{a(z)}$$

to a causal input sequence obtained by sampling $e^{i\omega t}$ with sampling period T, i.e. the input sequence is $\{x(k)\} = \{e^{i\omega kT}\zeta(k)\} = \{\left(e^{i\omega T}\right)^k \zeta(k)\}$. Assuming the system is initially quiescent,

$$Y(z) = H(z)X(z) = H(z)\,\frac{z}{z - e^{i\theta}},$$

where $\theta = \omega T$ is referred to as a **normalized frequency** variable. Thus,

$$\frac{Y(z)}{z} = \frac{b(z)}{a(z)}\,\frac{1}{z - e^{i\theta}}.$$

Suppose the system is BIBO stable and, for the sake of simplicity, that $H(z)$ has only simple poles at $z = \lambda_i,\ i = 1,\ 2, \ldots,\ n$. Performing a partial fraction expansion gives

$$\frac{Y(z)}{z} = \sum_{j=1}^{n} \frac{\alpha_j}{z - \lambda_j} + \frac{\beta}{z - e^{i\theta}}.$$

Therefore,

$$\begin{aligned}\{y(k)\} = \mathcal{Z}^{-1}\left[Y(z)\right] &= \mathcal{Z}^{-1}\left[\sum_{j=1}^{n} \frac{\alpha_j z}{z - \lambda_j} + \frac{\beta z}{z - e^{i\theta}}\right] \\ &= \left\{\left(\sum_{j=1}^{n} \alpha_j (\lambda_j)^k + \beta e^{ik\theta}\right)\zeta(k)\right\}.\end{aligned}$$

Since the system is assumed to be stable, then $|\lambda_j| < 1$ for all j and all the terms in the summation decay to zero as $k \to \infty$. This means that the long-term or **steady-state response** of a stable system with input sequence $\{x(k)\} =$

$\{e^{ik\theta}\zeta(k)\}$ is simply $\{\beta e^{ik\theta}\zeta(k)\} = \beta\{x(k)\}$. Using the cover-up rule, the factor β is found to be

$$\beta = H(e^{i\theta}),$$

which is the system function $H(z)$ evaluated on the unit circle $z = e^{i\theta}$. Thus, the frequency response can be determined from

$$H(e^{i\theta}) = H(z)\big|_{z=e^{i\theta}}.$$

For a stable system $H(z)$, the input signal $\{x(k)\} = \{e^{ik\theta}\}$ gives rise to the steady-state response

$$\{y(k)\} = \left\{H(e^{i\theta})e^{ik\theta}\right\}.$$

By taking real and imaginary parts, the respective steady-state responses to the inputs $\{\cos(k\theta)\}$ and $\{\sin(k\theta)\}$ are given by

$$\boxed{\Re\left\{H(e^{i\theta})e^{ik\theta}\right\} = \left\{|H(e^{i\theta})|\cos(k\theta + \phi_0)\right\}} \tag{3.3a}$$

and

$$\boxed{\Im\left\{H(e^{i\theta})e^{ik\theta}\right\} = \left\{|H(e^{i\theta})|\sin(k\theta + \phi_0)\right\},} \tag{3.3b}$$

where $|H(e^{i\theta})|$ is the gain spectrum and $\phi_0 = \arg H(e^{i\theta})$ is the phase-shift spectrum. In particular, if the input is the d.c. signal $\{A\zeta(k)\}$, where A is a constant, then the steady-state output is $\left\{AH(e^{i0})\zeta(k)\right\}$, where $H(e^{i0})$ is known as the *d.c. gain*.

Example 3.1 A causal, linear, time-invariant system is modelled by

$$y(k) + \tfrac{1}{2}y(k-2) = x(k-1), \quad k \in \mathbb{Z}.$$

a) Show that the system is BIBO stable.

b) Determine the gain and phase-shift spectra.

c) Find the steady-state output of the system when the input signal, for $k \geq 0$, is

$$x(k) = 3 - 6\sin\left(\tfrac{\pi}{3}k\right) + 5\cos\left(\tfrac{\pi}{2}k\right).$$

Solution:

a) Taking the z-transform of the difference equation,

$$Y(z) + \tfrac{1}{2}z^{-2}Y(z) = z^{-1}X(z)$$

and so

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^{-1}}{1+\frac{1}{2}z^{-2}}.$$

The poles of $H(z)$ satisfy $z^2 + \frac{1}{2} = 0$, i.e. $z = \pm\frac{1}{\sqrt{2}}i$. Since $|z| = \frac{1}{\sqrt{2}} < 1$, the system is BIBO stable.

b) The frequency response is

$$\begin{aligned} H(e^{i\theta}) = H(z)\big|_{z=e^{i\theta}} &= \frac{e^{-i\theta}}{1+\frac{1}{2}e^{-i2\theta}} = \frac{1}{e^{i\theta}+\frac{1}{2}e^{-i\theta}} \\ &= \frac{1}{\frac{3}{2}\cos(\theta) + i\frac{1}{2}\sin(\theta)}. \end{aligned}$$

Hence, the gain spectrum is

$$|H(e^{i\theta})| = \frac{1}{\sqrt{\frac{9}{4}\cos^2(\theta)+\frac{1}{4}\sin^2(\theta)}} = \frac{2}{\sqrt{9\cos^2(\theta)+\sin^2(\theta)}}$$

or

$$\frac{2}{\sqrt{1+8\cos^2(\theta)}} \quad \text{or} \quad \frac{2}{\sqrt{5+4\cos(2\theta)}}.$$

The phase-shift spectrum is

$$\begin{aligned} \arg H(e^{i\theta}) = \arg 1 - \arg\left(\tfrac{3}{2}\cos(\theta) + i\tfrac{1}{2}\sin(\theta)\right) &= 0 - \tan^{-1}\left(\frac{\frac{1}{2}\sin(\theta)}{\frac{3}{2}\cos(\theta)}\right) \\ &= -\tan^{-1}\left(\tfrac{1}{3}\tan(\theta)\right). \end{aligned}$$

c)

θ	$\left\|H\left(e^{i\theta}\right)\right\|$	$\arg H\left(e^{i\theta}\right)$
0	$\frac{2}{3}$	0
$\frac{\pi}{3}$	$\frac{2}{\sqrt{3}}$	$-\frac{\pi}{6}$
$\frac{\pi}{2}$	2	$-\frac{\pi}{2}$

Therefore, since $x(k) = 3 - 6\sin\left(\frac{\pi}{3}k\right) + 5\cos\left(\frac{\pi}{2}k\right) = 3e^{i0} - 6\Im\{e^{i\frac{\pi}{3}k}\} + 5\Re\{e^{i\frac{\pi}{2}k}\}$, then, using (3.3a-b), the steady-state output is

$$\begin{aligned} y(k) &= 3H\left(e^{i0}\right) - 6\left|H\left(e^{i\frac{\pi}{3}}\right)\right|\sin\left(\tfrac{\pi}{3}k + \arg H\left(e^{i\frac{\pi}{3}}\right)\right) \\ &\quad + 5\left|H\left(e^{i\frac{\pi}{2}}\right)\right|\cos\left(\tfrac{\pi}{2}k + \arg H\left(e^{i\frac{\pi}{2}}\right)\right) \\ &= 3 \times \tfrac{2}{3} - 6 \times \tfrac{2}{\sqrt{3}}\sin\left(\tfrac{\pi}{3}k - \tfrac{\pi}{6}\right) + 5 \times 2\cos\left(\tfrac{\pi}{2}k - \tfrac{\pi}{2}\right) \\ &= 2 - 4\sqrt{3}\sin\left((2k-1)\tfrac{\pi}{6}\right) + 10\sin\left(\tfrac{\pi}{2}k\right). \end{aligned}$$

□

Example 3.2 A causal digital filter has system function

$$H(z) = \frac{5(2 - z^{-1})}{(5 - 4z^{-1})(4 + 3z^{-1})}.$$

a) Find the difference equation realization of the filter.

b) Using MATLAB®, determine the poles and zeros of $H(z)$ and illustrate diagrammatically. Comment on the stability of the filter.

c) Plot, using MATLAB®, the gain and phase-shift spectra of $H(z)$.

d) Using MATLAB®, plot the response of the filter $\{y(k)\}$, for $k \in \mathbb{Z}$ satisfying $-10 \leq k \leq 40$, when the input is the (i) unit impulse $\{\delta(k-5)\}$, (ii) unit step $\{\zeta(k-5)\}$, (iii) unit pulse $\{p_{10}(k)\}$.

e) Find, using the z-transform final value theorem, the steady-state for the unit step response.

Solution:

a)

$$
\begin{aligned}
H(z) = \frac{Y(z)}{X(z)} &= \frac{5(2 - z^{-1})}{(5 - 4z^{-1})(4 + 3z^{-1})} \\
\Longrightarrow \quad (20 - z^{-1} - 12z^{-2})Y(z) &= (10 - 5z^{-1})X(z)
\end{aligned}
$$

Taking inverse z-transforms,

$$20y(k) - y(k-1) - 12y(k-2) = 10x(k) - 5x(k-1).$$

b) With $H(z)$ in ascending powers of z^{-1}, $H(z) = \dfrac{10 - 5z^{-1}}{20 - z^{-1} - 12z^{-2}} = \dfrac{b(z)}{a(z)}$. Thus, the following MATLAB® commands can be used.

```
b=[10 -5];
a=[20 -1 -12];
zplane(b,a);      % Plots the poles and zeros of H(z)
magz=abs(roots(a))' % Magnitudes of the poles of H(z)
```

The poles and zeros of $H(z)$ are illustrated in Figure 3.1.

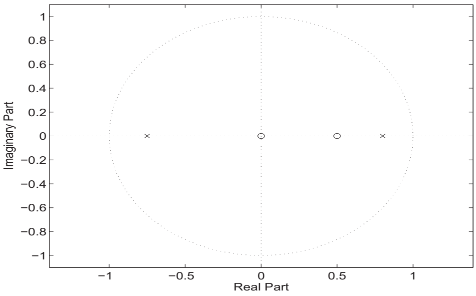


Figure 3.1: Pole-zero plot.

The commands produce the result:

```
magz= 0.8000 0.7500.
```

Since the filter is causal and the magnitudes of both poles are less than unity, the system is (BIBO) stable.

c) Plots of the gain and phase-shift spectra of $H(z)$ may be obtained using the MATLAB® commands:

```
% Evaluate the 256-point complex frequency response
[h,w]=freqz(b,a,256);
% Determine the gain and phase-shift spectra
mag=abs(h); phase=angle(h);
subplot(2,1,1); plot(w,mag); grid;
xlabel('frequency'); ylabel('magnitude');
title('Gain spectrum')
subplot(2,1,2); plot(w,phase); grid;
xlabel('frequency'); ylabel('phase');
title('Phase-shift spectrum')
```

and are illustrated in Figure 3.2.

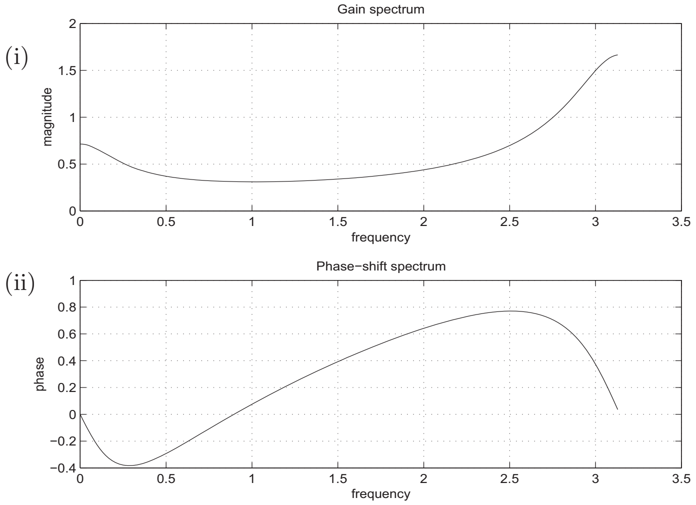


Figure 3.2: (i) Gain spectrum, and (ii) phase-shift spectrum for $H(z)$.

Alternatively, the command `fvtool(b,a)` can be used to display the gain spectrum and phase-shift spectrum.

d) The response to the unit impulse sequence $\{\delta(k-5)\}$ and the unit step sequence $\{\zeta(k-5)\}$ are shown in Figure 3.3 (i) and (ii), respectively, and the response to the unit pulse sequence $\{p_{10}(k)\}$ is shown in Figure 3.3 (iii).

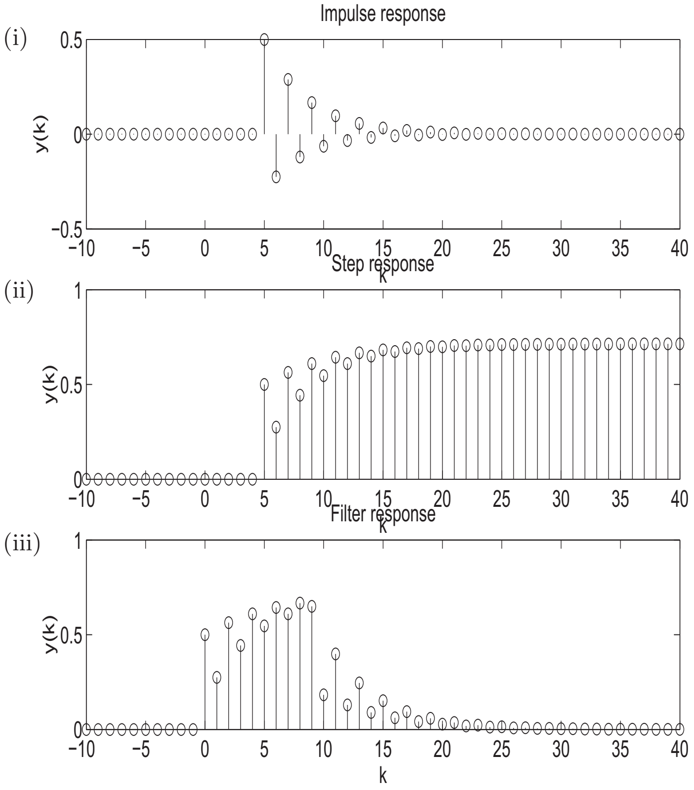


Figure 3.3: Filter response to (i) $\{\delta(k-5)\}$, (ii) $\{\zeta(k-5)\}$, and (iii) $\{p_{10}(k)\}$.

These graphs were obtained using the following MATLAB® commands.

```
k=[-10:40];
% Unit impulse:  delta(k-5) for -10<=k<=40
x1=[k-5 == 0];
% Unit step:  zeta(k-5) for -10<=k<=40
x2=[k-5 >= 0];
% Unit pulse:  p_10(k) for -10<=k<=40
x3=[k >= 0]-[k-10 >= 0];
y1=filter(b,a,x1); % Filter response to unit impulse
y2=filter(b,a,x2); % Filter response to unit step
y3=filter(b,a,x3); % Filter response to unit pulse
figure(3);
subplot(3,1,1);stem(k,y1); title('Impulse response');
xlabel('k'); ylabel('y(k)')
subplot(3,1,2);stem(k,y2); title('Step response');
xlabel('k'); ylabel('y(k)')
subplot(3,1,3);stem(k,y3); title('Filter response');
xlabel('k'); ylabel('y(k)')
```

e) Using the Final Value Theorem for z-transforms (see §2.3.1, property (Z7)),

$$\begin{aligned}\lim_{k\to\infty} y_\zeta(k) &= \lim_{z\to 1}(z-1)Y_\zeta(z)\\ &= \lim_{z\to 1} zH(z)\\ &= \lim_{z\to 1}\frac{5z(2-z^{-1})}{(5-4z^{-1})(4+3z^{-1})}\\ &= \tfrac{5}{7} \approx 0.7143\,.\end{aligned}$$

This is confirmed graphically in Figure 3.3 (ii).

□

MATLAB® is also useful for designing digital filters such as, for example, Butterworth and Chebyshev (try `butter` and `cheby1`, using the 'help' command in MATLAB®).

3.3 The Fourier transform

The *Fourier transform* of $t \mapsto f(t)$ is defined as

$$\boxed{\mathcal{F}\{f(t)\} \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\,\mathrm{d}t = F(i\omega),} \tag{3.4}$$

where ω is real, and the inverse transform is defined as

$$\boxed{f(t) = \mathcal{F}^{-1}\{F(i\omega)\} \overset{\text{def}}{=} \frac{1}{2\pi}\int_{-\infty}^{\infty} F(i\omega)e^{i\omega t}\,\mathrm{d}\omega.} \tag{3.5}$$

$F(i\omega)$ is called the continuous *frequency spectrum* of $f(t)$. We may write

$$F(i\omega) = |F(i\omega)|\,\exp(i\theta(\omega)).$$

Definition 3.2 *The continuous* ***amplitude spectrum*** *of $f(t)$ is defined as $|F(i\omega)|$ and $\theta(\omega)$ is the continuous* ***phase spectrum*** *of $f(t)$.*

A set of sufficient conditions for the existence of $F(i\omega)$ are the *Dirichlet* conditions:

a) $f(t)$ is absolutely integrable, that is

$$\int_{-\infty}^{\infty} |f(t)| \, \mathrm{d}t < \infty;$$

b) $f(t)$ must have a finite number of maxima and minima and finite discontinuities in any finite interval.

These sufficient conditions are satisfied by many useful signals; in particular, those signals for which

$$\int_{-\infty}^{\infty} |f(t)|^2 \, \mathrm{d}t < \infty. \tag{3.6}$$

The *energy* in a signal $f(t)$, $E(f)$, is defined by

$$E(f) \overset{\text{def}}{=} \int_{-\infty}^{\infty} |f(t)|^2 \, \mathrm{d}t.$$

Thus, the above conditions include *finite energy* signals, i.e. those that satisfy (3.6). For those signals that have infinite energy, the concept of *average power*, defined by

$$P_{av}(f) \overset{\text{def}}{=} \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} |f(t)|^2 \, \mathrm{d}t,$$

is more useful. For finite energy signals, the average power, is zero. Average power is more useful for aperiodic or random signals.

Remark 3.2 *It is possible to obtain Fourier representations for certain classes of infinite energy signals. In particular, Fourier representations can be obtained for signals whose average power is finite, that is $P_{av}(f) < \infty$.*

For finite energy signals, $E(f)$ can be expressed in the frequency domain.

$$\begin{aligned} E(f) &= \int_{-\infty}^{\infty} |f(t)|^2 \, \mathrm{d}t \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} f^*(t) \int_{-\infty}^{\infty} F(i\omega) e^{i\omega t} \, \mathrm{d}\omega \, \mathrm{d}t \quad \text{(using (3.5))} \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(i\omega) \int_{-\infty}^{\infty} f^*(t) e^{i\omega t} \, \mathrm{d}t \, \mathrm{d}\omega \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(i\omega) \mathrm{conj}\left(\int_{-\infty}^{\infty} f(t) e^{-i\omega t} \, \mathrm{d}t \right) \, \mathrm{d}\omega, \end{aligned}$$

where $\mathrm{conj}(\cdot)$ denotes the operation of complex conjugation. Hence, using (3.4),

$$E(f) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(i\omega) \mathrm{conj}(F(i\omega)) \, \mathrm{d}\omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} |F(i\omega)|^2 \, \mathrm{d}\omega.$$

This result is known as *Parseval's theorem* for Fourier transforms.

Definition 3.3

$$G(\omega) \overset{\text{def}}{=} \frac{1}{2\pi} |F(i\omega)|^2$$

is the ***energy spectral density*** *of the signal $f(t)$.*

Remark 3.3 *Analogous to $|F(i\omega)|$ being known as the amplitude spectrum of $f(t)$, $|F(i\omega)|^2$ is sometimes known as the* energy spectrum *of $f(t)$.*

3.3.1 Properties of the Fourier transform

(FT1) **Linearity:** If $\mathcal{F}\{f_1(t)\} = F_1(i\omega)$ and $\mathcal{F}\{f_2(t)\} = F_2(i\omega)$ then

$$\mathcal{F}\{af_1(t) + bf_2(t)\} = aF_1(i\omega) + bF_2(i\omega), \quad \text{where } a,\ b \text{ are constants.}$$

(FT2) **Symmetry:** If $\mathcal{F}\{f(t)\} = F(i\omega)$ then $\mathcal{F}\{F(it)\} = 2\pi f(-\omega)$.
If $t \mapsto f(t)$ is an even function, then $\mathcal{F}\{F(it)\} = 2\pi f(\omega)$.

(FT3) **Change of scale:** If $\mathcal{F}\{f(t)\} = F(i\omega)$ then, for a real constant $\alpha \neq 0$,

$$\mathcal{F}\{f(\alpha t)\} = \frac{1}{|\alpha|} F\left(i\frac{\omega}{\alpha}\right).$$

(FT4) **Time shift (Delay):** If $\mathcal{F}\{f(t)\} = F(i\omega)$ then

$$\mathcal{F}\{f(t-t_0)\} = F(i\omega)\exp(-i\omega t_0).$$

(FT5) **Frequency shift (Modulation):** If $\mathcal{F}\{f(t)\} = F(i\omega)$ then

$$\mathcal{F}\{f(t)\exp(i\omega_0 t)\} = F(i(\omega-\omega_0)).$$

(FT6) **Frequency differentiation and integration:** If $\mathcal{F}\{f(t)\} = F(i\omega)$ then

(i) $\mathcal{F}\{-itf(t)\} = \dfrac{\mathrm{d}F}{\mathrm{d}\omega}(i\omega)$

(ii) $\mathcal{F}\left\{\dfrac{f(t)}{-it}\right\} = \displaystyle\int F(i\omega)\,\mathrm{d}\omega.$

(FT7) **Time differentiation and integration:** If $\mathcal{F}\{f(t)\} = F(i\omega)$ then

(i) $\mathcal{F}\{f'(t)\} = i\omega F(i\omega)$, assuming $f \to 0$ as $t \to \pm\infty$.
In general, assuming $f,\ f', \dots f^{(n-1)} \to 0$ as $t \to \pm\infty$,

$$\mathcal{F}\left\{f^{(n)}(t)\right\} = (i\omega)^n F(i\omega).$$

(ii) $\mathcal{F}\left\{\displaystyle\int_{-\infty}^{t} f(u)\,\mathrm{d}u\right\} = \dfrac{1}{i\omega}F(i\omega) + \pi F(0)\delta(\omega),$
where $\delta(t)$, the *impulse* function, has the property:

$$\int_{-\infty}^{\infty} g(t)\delta(t-t_0)\,\mathrm{d}t = g(t_0), \tag{3.7}$$

assuming $g(t)$ is continuous at $t = t_0$.
This integration property can be applied to functions whose energy is infinite but average power is finite.

(FT8) **Convolution theorems:**

Theorem 3.1 *If $\mathcal{F}\{f(t)\} = F(i\omega)$ and $\mathcal{F}\{g(t)\} = G(i\omega)$, then*

$$\mathcal{F}\{f(t)g(t)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(iu)G(i(\omega-u))\,\mathrm{d}u \overset{\text{def}}{=} \frac{1}{2\pi}(F*G)(i\omega).$$

Remark 3.4 *The above result is sometimes known as* frequency convolution*.*

Alternatively, the convolution theorem may be written in the form:

Theorem 3.2 *If $\mathcal{F}\{f(t)\} = F(i\omega)$ and $\mathcal{F}\{g(t)\} = G(i\omega)$, then*

$$\mathcal{F}^{-1}\{F(i\omega)G(i\omega)\} = \int_{-\infty}^{\infty} f(u)g(t-u)\,\mathrm{d}u = (f * g)(t).$$

Here, $(f * g)(t)$ is referred to as *time convolution*.

Remark 3.5 *The convolution operation has both the commutative and associative properties.*

3.3.2 Fourier transform of power signals

Some functions are not absolutely integrable, for example the unit impulse function, the unit step function and trigonometric functions. For each of the above cases, the signals have infinite energy, but their power is finite.

(i) **Impulse function:** An application of (3.7) with $g(t) = e^{-i\omega t}$ gives

$$\int_{-\infty}^{\infty} e^{-i\omega t}\delta(t-t_0)\,\mathrm{d}t = e^{-i\omega t_0},$$

that is

$$\mathcal{F}\{\delta(t-t_0)\} = \exp(-i\omega t_0).$$

In particular,

$$\mathcal{F}\{\delta(t)\} = 1.$$

Using the symmetry property (FT2) and the change of scale property (FT3),

$$\mathcal{F}\{\exp(i\omega_0 t)\} = 2\pi\delta(\omega-\omega_0) \quad \text{and, hence,} \quad \mathcal{F}\{1\} = 2\pi\delta(\omega). \tag{3.8}$$

(ii) **Unit step function:** Using the impulse function property (3.7), for $t \neq 0$,

$$\zeta(t) = \int_{-\infty}^{\infty} \zeta(\tau)\delta(\tau - t)\,\mathrm{d}\tau.$$

Let $\tau = t - v$ then, as a result of the definition of $\zeta(t)$,

$$\zeta(t) = \int_{-\infty}^{\infty} \zeta(t-v)\delta(-v)\ \mathrm{d}v = \int_{-\infty}^{t} \delta(-v)\ \mathrm{d}v\ , \quad \text{for } t \neq 0.$$

Using the Fourier transform property (FT7)(*ii*),

$$\mathcal{F}\left\{\zeta(t)\right\} = \mathcal{F}\left\{\int_{-\infty}^{t} \delta(-v)\ \mathrm{d}v\right\} = \frac{1}{i\omega}\mathcal{F}\left\{\delta(-t)\right\} + \pi\big[\mathcal{F}\left\{\delta(-t)\right\}\big]_{\omega=0}\delta(\omega).$$

Since $\mathcal{F}\left\{\delta(t)\right\} = 1$, it follows from property (FT3) that $\mathcal{F}\left\{\delta(-t)\right\} = 1$. Therefore,

$$\mathcal{F}\left\{\zeta(t)\right\} = \frac{1}{i\omega} + \pi\delta(\omega).$$

3.3.3 Fourier transforms of periodic signals

Suppose $T \in \mathbb{R}^+$ and $f(t)$ is periodic with period T. Then $f(t)$ can be expressed in terms of the exponential Fourier series as

$$f(t) = \sum_{n=-\infty}^{\infty} F_n e^{in\omega_0 t},$$

where $\omega_0 = \dfrac{2\pi}{T}$ (rad.sec^{-1}) is the *fundamental frequency* and

$$F_n = \frac{1}{T}\int_{period} f(t)e^{-in\omega_0 t}\ \mathrm{d}t.$$

Now, formally,

$$F(i\omega) = \mathcal{F}\left\{\sum_{n=-\infty}^{\infty} F_n e^{in\omega_0 t}\right\} = \sum_{n=-\infty}^{\infty} F_n \mathcal{F}\left\{e^{in\omega_0 t}\right\},$$

using the linear property (FT1). Here, it is assumed that the function f is such that the change in order of summation and integration is allowable. It follows from (3.8) that

$$\boxed{F(i\omega) = 2\pi \sum_{n=-\infty}^{\infty} F_n \delta(\omega - n\omega_0).} \qquad (3.9)$$

The result (3.9) shows that the Fourier amplitude spectrum of the periodic signal $f(t)$ is given by a series of impulses of magnitude $2\pi|F_n|$ at intervals of ω_0.

The next example illustrates how to identify, in a straightforward manner, the particular frequencies of periodic components in a signal using the Fourier transform.

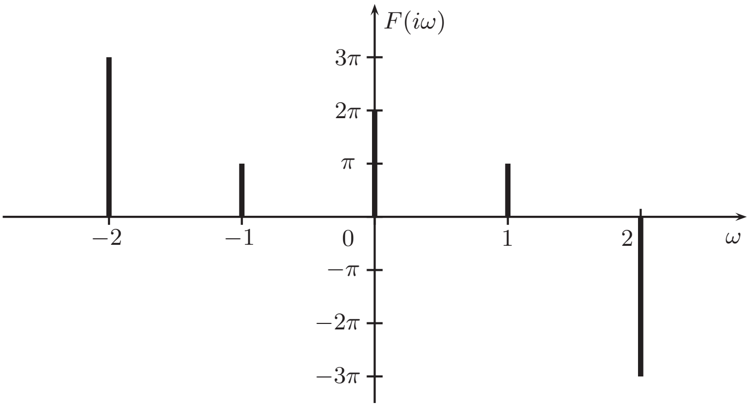
Example 3.3 Determine the Fourier transform of $f(t) = 1 + \cos(t) - 3\sin(2t)$.

Solution: Since $1 = e^{0it}$, $\cos(t) = \frac{1}{2}\left(e^{it} + e^{-it}\right)$ and $\sin(2t) = \frac{1}{2}\left(e^{2it} - e^{-2it}\right)$, then f can be expressed as: $f(t) = e^{0it} + \frac{1}{2}\left(e^{it} + e^{-it} - 3e^{2it} + 3e^{-2it}\right)$. Since f is periodic with period $T = 2\pi$,

$$f(t) = \sum_{n=-\infty}^{\infty} F_n e^{in\omega_0 t}$$

with $\omega_0 = 2\pi/T = 1$, $F_0 = 1$, $F_1 = F_{-1} = \frac{1}{2}$, $F_2 = -\frac{3}{2}$, $F_{-2} = \frac{3}{2}$, and $F_n = 0$, otherwise. Thus, from (3.9) with $\omega_0 = 1$, it follows that

$$\begin{aligned}\mathcal{F}\{f(t)\} = F(i\omega) &= 2\pi \sum_{n=-\infty}^{\infty} F_n \delta(\omega - n) \\ &= \pi[2\delta(\omega) + \delta(\omega - 1) + \delta(\omega + 1) - 3\delta(\omega - 2) + 3\delta(\omega + 2)].\end{aligned}$$



Thus, the impulses in the Fourier transform at $\omega = \pm 1$ and ± 2 indicate that the signal, $f(t)$, has periodic components with (angular) frequencies of 1 and 2 rads./sec., respectively.

□

Fourier transform of a unit comb function

Consider the *unit comb function*, which is defined by

$$t \mapsto \gamma_{_T}(t) \overset{\text{def}}{=} \sum_{k=-\infty}^{\infty} \delta(t - kT).$$

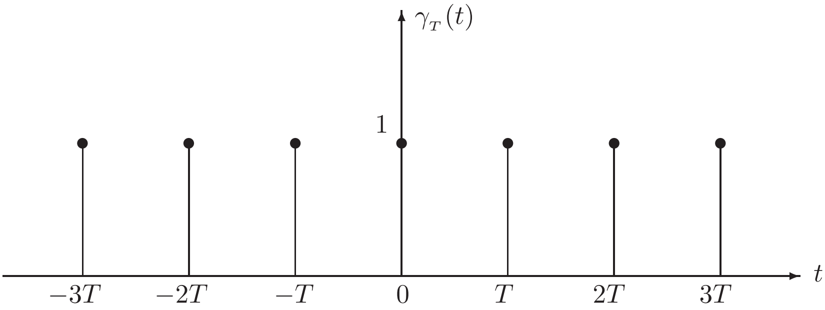


Figure 3.4: A graphical representation of a unit comb function

Since $\gamma_{_T}(t)$ is periodic, with period T, it can be represented by its Fourier series:

$$\gamma_{_T}(t) = \sum_{k=-\infty}^{\infty} \Gamma_n e^{ik\omega_0 t}, \quad \text{where } \omega_0 = \frac{2\pi}{T} \text{ and } \Gamma_n = \frac{1}{T}\int_{-\frac{1}{2}T}^{\frac{1}{2}T} \gamma_{_T}(t) e^{-ik\omega_0 t}\ \mathrm{d}t.$$

By definition of γ_T,

$$\begin{aligned}
\Gamma_n &= \frac{1}{T}\int_{-\frac{1}{2}T}^{\frac{1}{2}T} \delta(t) e^{-ik\omega_0 t}\ \mathrm{d}t \\
&= \frac{1}{T}\int_{-\infty}^{\infty} g(t)\delta(t)\ \mathrm{d}t, \quad \text{where } g(t) = \begin{cases} e^{ik\omega_0 t}, & |t| < \frac{1}{2}T, \\ 0, & \text{otherwise,} \end{cases} \\
&= \frac{1}{T} g(0), \quad \text{by definition of } \delta(t), \\
&= \frac{1}{T}.
\end{aligned}$$

Hence

$$\gamma_{_T}(t) = \frac{1}{T}\sum_{n=-\infty}^{\infty} e^{ik\omega_0 t}.$$

Taking the Fourier transform of both sides and using (3.9),

$$\mathcal{F}\left\{\gamma_{_T}(t)\right\} = \frac{2\pi}{T} \sum_{n=-\infty}^{\infty} \delta(\omega - n\omega_0)$$

and, since $\omega_0 = 2\pi/T$,

$$\boxed{\mathcal{F}\left\{\gamma_{_T}(t)\right\} = \omega_0 \sum_{n=-\infty}^{\infty} \delta(\omega - n\omega_0) = \omega_0 \gamma_{\omega_0}(\omega).} \tag{3.10}$$

Thus, the transform of a unit comb function is composed of the product of ω_0 with the unit comb function in the frequency domain, where the impulses are located at the harmonic frequencies $n\omega_0 = n2\pi/T$.

3.4 The sampling theorem and aliasing

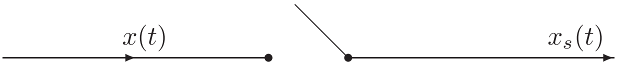


Figure 3.5: Sampling switch.

An analogue signal $x(t)$ can be sampled at the points $t = nT$, where T is the *sampling period*, to produce sampled values. This can be achieved using a *sampling switch* which closes briefly every T seconds. The switch is such that a value of $x(t)$ is obtained when the switch is closed and a zero value when the switch is open. The sampled signal $x_s(t)$ can be modelled by $x_s(t) = x(t)p(t)$, where $p(t)$ is called the *sampling function*. In practice, the time during which $p(t)$ is non-zero is small compared to the period T and ideally $p(t)$ can be modelled by an infinite train of impulse functions of period T. Thus, an appropriate representation for $p(t)$ is

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) = \gamma_{_T}(t).$$

Hence,

$$x_s(t) = x(t)\gamma_{_T}(t).$$

Using the frequency convolution theorem (Theorem 3.1) and (3.10),

$$\begin{aligned}\mathcal{F}\left\{x_s(t)\right\} &= \frac{\omega_0}{2\pi} X(i\omega) * \gamma_{\omega_0}(\omega), \quad \text{where } \omega_0 = \frac{2\pi}{T}, \\ &= \frac{1}{T} \sum_{n=-\infty}^{\infty} X(i\omega) * \delta(\omega - n\omega_0).\end{aligned}$$

Formally, by interchanging the order of integration and using the definition of $\delta(t)$,

$$\begin{aligned}X(i\omega) * \delta(\omega - n\omega_0) &= \int_{-\infty}^{\infty} X(iu)\delta(\omega - n\omega_0 - u)\,\mathrm{d}u \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t)e^{-iut}\delta(\omega - n\omega_0 - u)\,\mathrm{d}u\,\mathrm{d}t \\ &= X(i(\omega - n\omega_0)).\end{aligned}$$

Therefore,

$$\boxed{\mathcal{F}\left\{x_s(t)\right\} = \frac{1}{T} \sum_{n=-\infty}^{\infty} X(i(\omega - n\omega_0)) = f_s \sum_{n=-\infty}^{\infty} X(i(\omega - n\omega_0)),} \tag{3.11}$$

where $\quad f_s = T^{-1} \quad$ is the *sampling frequency.*

In all practical signals, there is some frequency beyond which the energy is negligible, for example, in musical signals frequencies beyond the range of the human ear are of no interest. Let $\omega_{\max} \in \mathbb{R}^+$ be the maximum frequency of interest. Now suppose all the frequencies above the maximum value are filtered out. This new signal is still denoted by $x(t)$ since it contains all the useful information in the original signal. Suppose $x(t)$ has an amplitude spectrum of the form illustrated in Figure 3.6. Here, $X(i\omega)$ is said to be *bandlimited* in the sense that it is zero beyond the frequency $\omega_{\max}$. In this case, $\omega_{\max}$ is called the *bandwidth* of the continuous-time signal $x(t)$. Since $\displaystyle\sum_{n=-\infty}^{\infty} X(i(\omega - n\omega_0))$ is a periodic function with period ω_0, if $\frac{1}{2}\omega_0 > \omega_{\max}$ then $\displaystyle\left|\sum_{n=-\infty}^{\infty} X(i(\omega - n\omega_0))\right|$ has graph as shown in Figure 3.7. It is clear from Figure 3.7 that all the relevant information of $|X(i\omega)|$ is retained. However, if $\frac{1}{2}\omega_0 < \omega_{\max}$ the shifted graphs overlap with each other and this causes loss of information in the band

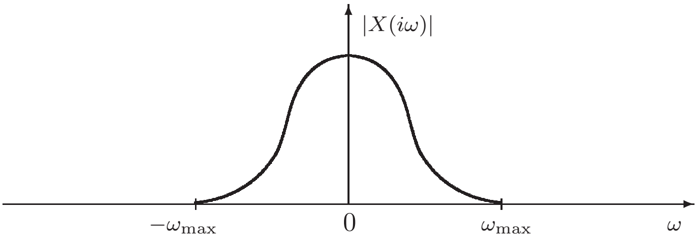


Figure 3.6: Amplitude spectrum of a bandlimited signal.

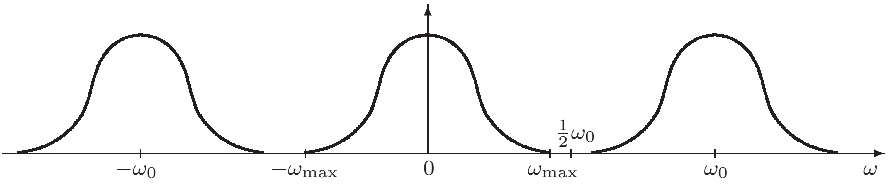


Figure 3.7: Amplitude spectrum of a sampled signal with no aliasing.

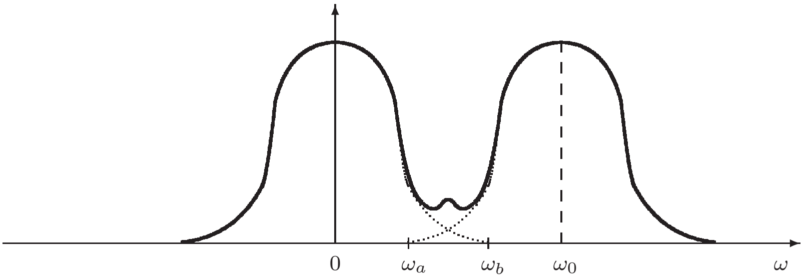


Figure 3.8: Amplitude spectrum with aliasing, when $\omega_a = \omega_0 - \omega_{\max} < \omega_b = \omega_{\max}$.

$[\omega_0 - \omega_{\max},\ \omega_{\max}]$ (see Figure 3.8). This is known as *aliasing*. To remove aliasing, we must ensure $\frac{1}{2}\omega_0 \geq \omega_{\max}$, that is $f_s \geq \omega_{\max}/\pi$. Defining

$$f_{\max} \overset{\text{def}}{=} \omega_{\max}/(2\pi),$$

which is the maximum frequency in Hz (Hertz), then $f_s \geq 2f_{\max}$ and so, in order to avoid aliasing, the minimum sampling frequency must be $2f_{\max}$ Hz. Alternatively, in terms of the sampling period, $T \leq (2f_{\max})^{-1}$.

Theorem 3.3 *A bandlimited signal $x(t)$, having no frequency components above $f_{\max}$ Hz, is completely specified by samples that are taken at a uniform rate greater than $2f_{\max}$ Hz, that is the sampling interval is less than $(2f_{\max})^{-1}$ seconds, or, alternatively, the signal bandwidth must be less than π/T, where T is the sampling period.*

Remarks 3.6

(i) *The above theorem is usually attributed to either Nyquist or Shannon (or both). The frequency $2f_{\max}$ is known as the* Nyquist rate *and π/T is called the* Nyquist frequency.

(ii) *In order to account for other effects than simply aliasing, for example, sampling over a finite interval, in practice, $T < (10f_{\max})^{-1}$ is recommended as a guideline.*

(iii) *Further information on sampling can be found in [Bellanger, 1990, Haddad and Parsons, 1991, Oppenheim et al., 1999] and, in addition, see [Ziemer et al., 1998].*

The Fourier transform of the analogue signal can be recovered from the Fourier transform of the sampled data signal by filtering out all frequency components above $\omega_{\max}$. This can be accomplished by using a *low pass* filter which transmits all frequencies below $\omega_{\max}$ and attenuates all frequencies above $\omega_{\max}$. For example, the filter with

$$H(i\omega) \overset{\text{def}}{=} \begin{cases} 1, & |\omega| < \omega_{\max}, \\ 0, & \text{otherwise}, \end{cases}$$

where $H(i\omega)$ is the Fourier transform of the impulse response of the filter, is an *ideal* low pass filter. From (3.11),

$$X_s(i\omega)H(i\omega) = \frac{1}{T}X(i\omega),$$

or

$$X(i\omega) = TX_s(i\omega)H(i\omega).$$

This result can then be used to show how $x(t)$ can be recovered from $x_s(t)$. From tables of the Fourier transform,

$$\mathcal{F}\left\{h(t)\right\} = 2\omega_{\text{max}}\text{sinc}(\omega_{\text{max}}\omega),$$

where

$$h(t) = \begin{cases} 1, & |t| < \omega_{\text{max}}, \\ 0, & \text{otherwise}, \end{cases} \quad \text{and} \quad x \mapsto \text{sinc}(x) \overset{\text{def}}{=} \begin{cases} \dfrac{\sin(x)}{x}, & \text{for } x \neq 0, \\ 1, & \text{for } x = 0. \end{cases}$$

Hence, using the symmetry property (FT2),

$$\mathcal{F}\left\{2\omega_{\text{max}}\text{sinc}(\omega_{\text{max}}t)\right\} = 2\pi H(i\omega),$$

where $H(i\omega)$ is defined above, and so

$$\mathcal{F}^{-1}\left\{H(i\omega)\right\} = \pi^{-1}\omega_{\text{max}}\text{sinc}(\omega_{\text{max}}t).$$

Therefore,

$$\begin{aligned} x(t) &= \mathcal{F}^{-1}\left\{X(i\omega)\right\} \\ &= \mathcal{F}^{-1}\left\{TX_s(i\omega)H(i\omega)\right\} \\ &= Tx_s(t) * [\pi^{-1}\omega_{\text{max}}\text{sinc}(\omega_{\text{max}}t)], \quad \text{using Theorem 3.2}, \\ &= x_s(t) * \text{sinc}(\omega_{\text{max}}t), \quad \text{since } T = (2f_{\text{max}})^{-1} = \pi(\omega_{\text{max}})^{-1}, \\ &= \left\{\sum_{n=-\infty}^{\infty} x(nT)\delta(t-nT)\right\} * \text{sinc}(\omega_{\text{max}}t). \end{aligned}$$

Now, by definition of $\delta(t)$, it follows from the continuous-time convolution property (CC4) that

$$\delta(t-nT)*\text{sinc}(\omega_{\text{max}}t) = \int_{-\infty}^{\infty}\delta(u-nT)\text{sinc}(\omega_{\text{max}}(t-u))\,du = \text{sinc}(\omega_{\text{max}}(t-nT)).$$

Thus

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT)\text{sinc}(\omega_{\text{max}}(t-nT)).$$

3.5 The discrete Fourier transform (DFT)

discrete Fourier transform (DFT) The z-transform of a sequence $\{x(k)\}$ is

$$X(z) = \sum_{k=-\infty}^{\infty} x(k) z^{-k}$$

and so the frequency response can be expressed as

$$X\left(e^{i\omega}\right) = \sum_{k=-\infty}^{\infty} x(k) e^{-i\omega k}. \tag{3.12}$$

Here, $\sum_{k=-\infty}^{\infty} x(k)e^{-i\omega k}$ is known as the *discrete-time Fourier transform* (DTFT) of the sequence $\{x(k)\}$. The inverse transform is defined by

$$x(k) \overset{\text{def}}{=} \frac{1}{2\pi} \int_{-\pi}^{\pi} X\left(e^{i\omega}\right) e^{i\omega k}\, \mathrm{d}\omega.$$

The formula (3.12) is not directly applicable to data analysis using a computer since

a) the formula assumes $x(k)$ is known for all k;

b) ω is continuous.

Suppose an analogue signal, $x(t)$, is sampled to produce a discrete signal $x(kT)$, where it is assumed that the sample values are equally spaced with sampling period T. In practice, it is more likely that a finite set of N sampled values $\{x(k);\ k = 0, 1, \ldots, N-1\}$, where

$$x(k) = x(kT),$$

have been generated. The spectrum of this sequence is

$$X\left(e^{i\theta}\right) = \sum_{k=0}^{N-1} x(k) e^{-i\theta k},$$

where θ is the normalised frequency $\theta = \omega T$.

Suppose $X\left(e^{i\theta}\right)$ is sampled uniformly at intervals of $\theta = \dfrac{2\pi}{N}$, that is N distinct values of $X\left(e^{i\theta}\right)$, say $X(n)$, $n = 0, 1, \ldots, N-1$, then

$$X(n) = \sum_{k=0}^{N-1} x(k) e^{-i\frac{2\pi kn}{N}}.$$

By convention, $X(n)$, a sampled version of the periodic discrete-time Fourier transform, is known as the *discrete Fourier transform* (DFT) of the sequence $\{x(k)\}$. The *inverse discrete Fourier transform* (IDFT) of the sequence $\{X(n)\}$ is

$$x(k) = \frac{1}{N} \sum_{n=0}^{N-1} X(n) e^{i\frac{2\pi kn}{N}}.$$

It can be shown, see Discrete-time Signal Processing by A.V. Oppenheim and R.W. Schafer (Prentice-Hall, 1989), that an estimate of the Fourier transform of the analogue signal $x(t)$, sampled at small intervals, is the DFT of $\{x(k)\}$ multiplied by the factor TN^{-1}. Special algorithms have been developed to determine the discrete Fourier transform efficiently (usually N is a power of 2). These are referred to as *fast Fourier transforms* (FFTs).

There are three possible transform domains for the DFT, namely n or $\theta_n \overset{\text{def}}{=} n(2\pi/N)$ or $\Omega_n \overset{\text{def}}{=} n(2\pi/(NT))$.

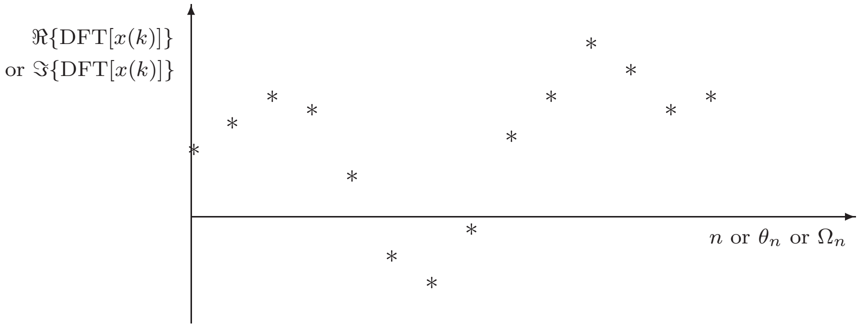


Figure 3.9: DFT transform domains.

1. n is called the *digital frequency index*.

2. θ_n is the *digital frequency* (rads.).

3. Ω_n is the *analogue frequency* (rads./sec.).

Also, $\delta\theta \overset{\text{def}}{=} \theta_1 = 2\pi/N$ rads. is known as the *digital frequency resolution*, whilst $\delta\Omega \overset{\text{def}}{=} \Omega_1 = 2\pi/(NT)$ rads./sec. is known as the *analogue frequency resolution.*

Remark 3.7 *Since frequency indices greater than $N/2$ are redundant for real signals, the highest observable unaliased frequency is related to the $N/2$ frequency index. In the case of the digital frequency $\theta_{\frac{1}{2}N}$, this is π rads., whilst the analogue frequency $\Omega_{\frac{1}{2}N}$ is π/T rads./sec.*

Note that the duration of the time signal being sampled is

$$\boxed{NT = \frac{2\pi}{\delta\Omega}.} \tag{3.13}$$

By the sampling theorem (namely Theorem 3.3), for a bandlimited signal to be recoverable from the samples of the signal, $f_s \geq 2f_h$, that is

$$T = \frac{1}{f_s} \leq \frac{1}{2f_h},$$

where f_s denotes sampling frequency and f_h represents the highest frequency content of the signal (in Hz). Using (3.13), this provides a lower limit on the number of signal samples that must be taken in order to meet a specified frequency resolution. Thus, if the analogue frequency resolution, $\delta\Omega$, is specified then, to avoid aliasing, the number of samples must satisfy

$$N \geq \frac{4\pi f_h}{\delta\Omega},$$

whilst if the frequency resolution is specified in Hertz, say δf, then

$$N \geq \frac{2f_h}{\delta f},$$

since $\delta f = \delta\Omega/(2\pi)$. This is then the lower limit on the number of samples required to compute the DFT.

3.6 Discrete spectra

For a discrete signal $x(k)$ of length N, its *energy* is defined as

$$E(x) \overset{\text{def}}{=} \sum_{k=0}^{N-1} |x(k)|^2.$$

Parseval's relation for discrete signals takes the form:

$$\boxed{\sum_{k=0}^{N-1} |x(k)|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |X(n)|^2.} \tag{3.14}$$

This relation shows that the energy computation can be carried out in the frequency domain as well as the time domain.
As in the case of a continuous time signal, discrete spectra can be defined.

Definition 3.4 *Since $X(n) = |X(n)| \exp(i\psi_n)$, the* DFT ***amplitude spectrum*** *is defined to be $|X(n)|$ and ψ_n is the* DFT ***phase spectrum****.*

For applications, the discrete spectra can be computed using MATLAB®. The DFT and IDFT are obtained using the commands `fft` and `ifft` (note that, in MATLAB®, the i notation is replaced by j).

Example 3.4 Using MATLAB®, plot the amplitude and phase spectra for the sequence $\{x(k)\}$, where

$$x(k) = \sin(2\pi f_1 k) + \cos(2\pi f_2 k),$$

$f_1 = 10$, $f_2 = 25$ and $k = 0 : 0.01 : n - 0.01$, with $n = 30$.

Solution: The following MATLAB® commands:

```
n=30; k=[0:0.01:n-0.01];
x=sin(2*pi*10*k)+cos(2*pi*25*k); % Generates given sequence
y=fft(x);                        % Computes the DFT of x
% Determine the amplitude and phase spectra
as=abs(y); ps=unwrap(angle(y));
f=[0:length(y)-1]*99/length(y); % computes the frequency vector
subplot(121); plot(f,as); title('Amplitude');
set(gca,'XTick',[10 25 75 90]);
subplot(122); plot(f,ps*180/pi); title('Phase');
set(gca,'XTick',[10 25 75 90]);
```

produce the plots in Figure 3.10.

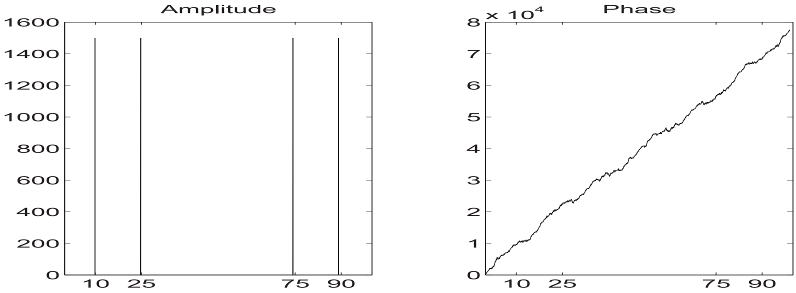


Figure 3.10: Gain and phase spectra.

□

When processing a signal, sometimes the signal is padded with additional zeros. This has the effect of producing closely spaced samples in the spectrum, that is it produces a high-density spectrum. However, it does not give a high-resolution spectrum, because no new information is added to the signal. To obtain a high-resolution spectrum more signal data is required.

Example 3.5 Consider the sequence $\{x(k)\}$, where

$$x(k) = \cos(0.4\pi k) - \sin(0.6\pi k).$$

a) Plot the amplitude spectrum of $\{x(k)\}$ for $0 \le k \le 10$, $k \in \mathbb{Z}$.

b) Append 490 zeros to $\{x(k)\}$, $0 \le k \le 10$, $k \in \mathbb{Z}$, and plot the new amplitude spectrum and compare the result with that obtained in a).

c) Using 490 additional sampling points, determine and plot the amplitude spectrum of $\{x(k)\}$ for $0 \le k \le 500$, $k \in \mathbb{Z}$.

Solution:

a) Using MATLAB®, an estimate of the amplitude spectrum is obtained by the following commands:

```
n=10; k1=[0:1:n];
x1=cos(0.4*pi*k1)-sin(0.6*pi*k1); y1=fft(x1);
w=2*pi/length(y1)*k1; plot(w/pi,abs(y1)), grid,
title('Amplitude spectrum'),
xlabel('Frequency in multiples of pi');
```

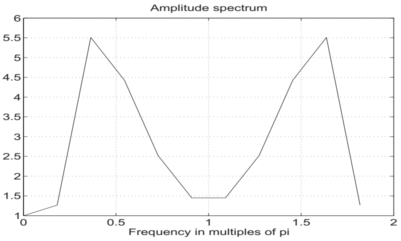


Figure 3.11: Amplitude spectrum for $x(k) = \cos(0.4\pi k) - \sin(0.6\pi k)$, $0 \leq k \leq 10$.

It is clear from Figure 3.11 that it is difficult to draw any conclusions due to the lack of samples in the spectrum.

b) The required MATLAB® commands are:

```
k2=[0:1:50*n];
x2=[x1 zeros(1,490)]; y2=fft(x2);
w=2*pi/length(y2)*k2; plot(w/pi,abs(y2)), grid,
title('Amplitude spectrum'),
xlabel('Frequency in multiples of pi');
```

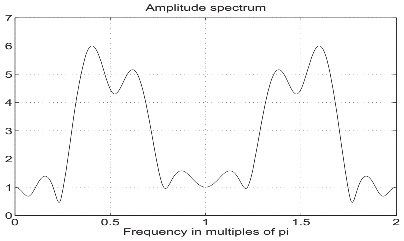


Figure 3.12: Amplitude spectrum for zero-padded data.

Using the data padded with additional zeros, Figure 3.12 indicates that, possibly, the sequence has two dominant frequencies at $\omega = 0.4\pi$ and $\omega = 0.6\pi$, approximately.

c) The MATLAB® commands:

```
x3=cos(0.4*pi*k2)-sin(0.6*pi*k2); y3=fft(x3);
w=2*pi/length(y3)*k2; plot(w/pi,abs(y3)), grid,
title('Amplitude spectrum'),
xlabel('Frequency in multiples of pi');
```

produce the graph shown in Figure 3.13. With additional sequence data,

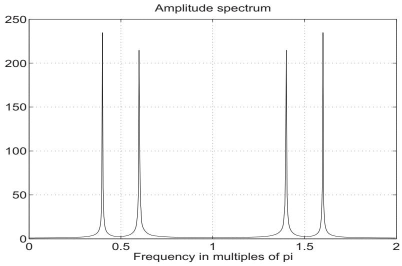


Figure 3.13: Amplitude spectrum for additional sampling points.

Figure 3.13 confirms that there are two dominant frequencies at $\omega = 0.4\pi$ and $\omega = 0.6\pi$.

□

The two-dimensional FFT and its inverse can be computed in MATLAB® with the commands: `fft2` and `ifft2`. These functions are useful for 2-dimensional signal processing and image processing.

3.6.1 Power spectrum

The total *average power* of a sequence $\{x(k),\ k \in \mathbb{Z}\}$ is

$$P_{av}(x) \stackrel{\text{def}}{=} \lim_{N\to\infty} \frac{1}{2N+1} \sum_{k=-N}^{N} |x(k)|^2.$$

For a finite sequence $\{x(k),\ k = 0,\ 1, \dots N-1\}$,

$$P_{av}(x) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{k=0}^{N-1} |x(k)|^2 = \frac{1}{N^2} \sum_{n=0}^{N-1} |X(n)|^2, \ \text{(using (3.14))}.$$

The *power spectrum* of the finite sequence $\{x(k),\ k = 0,\ 1, \ldots N-1\}$ is defined by

$$\boxed{P(n) \overset{\text{def}}{=} \frac{|X(n)|^2}{N^2} \quad \text{for } 0 \le n \le N-1.}$$

3.7 Digital filters

To process signals, appropriate filters, such as MA, AR, ARMA, for example, are designed to achieve some desired objectives, such as passing or rejecting signals of different frequencies. A **low-pass filter** is designed to pass all signals with frequencies less than a **cut-off**, or **critical frequency**, and reject all signals greater than the cut-off frequency, whilst a **high-pass filter**has the opposite effect. Thus, a high-pass filter rejects all signals of frequency less than the cut-off frequency and pass signals whose frequency content is greater than the cut-off frequency. Other types of filters are the **band-reject filter** and **band-pass filter**; for more details, see Signal Processing In Electronic Communication by Chapman *et al* (Horwood Publishing, 1997).

Example 3.6 A third order filter is described by the difference equation

$$\begin{aligned} y(k) = 0.5772y(k-1) - 0.4218y(k-2) + 0.0563y(k-3) + 0.0985x(k) \\ + 0.2956x(k-1) + 0.2956x(k-2) + 0.0985x(k-3) \end{aligned}$$

and the input to the filter is the signal described by, for $0 \le k \le 100$, $k \in \mathbb{Z}$,

$$x(k) = \sin(0.1\pi k) - \cos(0.3\pi k) - 3\cos(0.6\pi k) + \cos(0.7\pi k) + 2\sin(0.8\pi k).$$

Using MATLAB®, plot the amplitude and phase spectra of the filter and confirm that it is a low-pass filter by plotting the amplitude spectrum of the filtered signal.

Solution: The following MATLAB® commands:

```
a=[1.0000 -0.5772 0.4218 -0.0563];
b=[0.0985 0.2956 0.2956 0.0985];
[h,w]=freqz(b,a,256);
mag=abs(h); phase=angle(h);
figure(1), plot(mag), title('Amplitude spectrum'),
grid, xlabel('Frequency in multiples of pi');
figure(2), plot(phase), title('Phase spectrum'),
grid, xlabel('Frequency in multiples of pi');
k=[0:100];
x=sin(0.1*pi*k)-cos(0.3*pi*k)-3*cos(0.6*pi*k)+cos(0.7*pi*k)...
  ...+2*sin(0.8*pi*k);
y=filter(b,a,x);
w=2*pi/length(y)*k; as=abs(fft(y));
figure(3),plot(w/pi,as), title('Amplitude spectrum'),
grid, xlabel('Frequency in multiples of pi');
```

produce the graphs illustrated in Figures 3.14 and 3.15. Figure 3.15 indicates that the components of $\{x(k)\}$ with frequencies $\omega = 0.6\pi,\ 0.7\pi$ and $\omega = 0.8\pi$ have been attenuated by the low-pass filter.

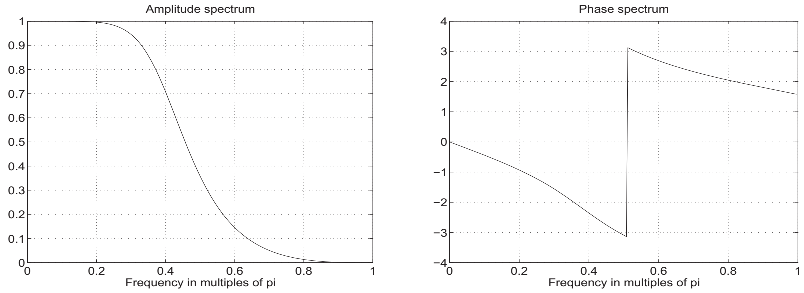


Figure 3.14: Amplitude and phase spectra for the filter.

□

The *bandwidth* of a filter is a measure of the extent of the significant band of frequencies in the spectrum. There are many ways in which this can be defined. One particular definition is the interval of frequencies for which the amplitude spectrum remains within $1/\sqrt{2}$ of its maximum value. For example, consider the

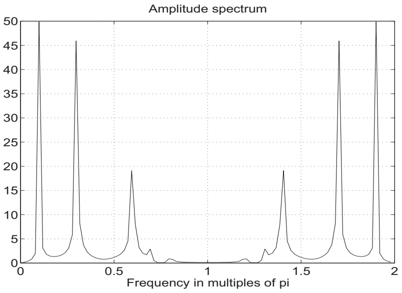


Figure 3.15: Amplitude spectrum for the filtered signal.

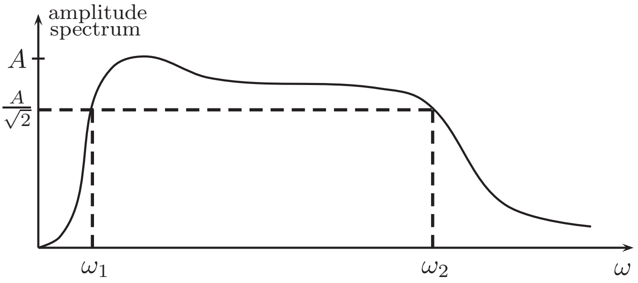


Figure 3.16: Amplitude spectrum with bandwidth $(\omega_2 - \omega_1)$.

amplitude spectrum illustrated in Figure 3.16. For this example the bandwith is $(\omega_2 - \omega_1)$.

An ideal frequency-selective filter has, in general, a noncausal, infinite-duration, impulse response. Often when filtering signals, it is desired to implement a causal filter with *finite impulse response* (FIR). This can be achieved by selecting an appropriate window function, as described in Section 3.8, and windowing the signal data.

3.8 Windows

Often in spectral analysis, there is a need to apply a time-weighting function to a signal, known as a *window*, before applying a Fourier transformation in order to achieve some desired objective. Here, in this section, we consider some well-known windows.

Rectangular window

The rectangular window sequence, illustrated in Figure 3.17, is defined by

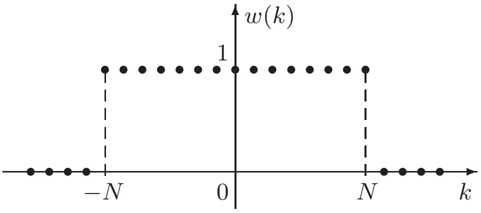


Figure 3.17: Rectangular window.

$$w(k) = rect_N(k) \overset{\text{def}}{=} \zeta(k+N) - \zeta(k-(N+1)) = \begin{cases} 1, & |k| \le N \\ 0, & \text{otherwise.} \end{cases}$$

Since

$$W(z) = \left(z^N - z^{-(N+1)}\right)\left(\frac{z}{z-1}\right) = \frac{z^{N+\frac{1}{2}} - z^{-(N+\frac{1}{2})}}{z^{\frac{1}{2}} - z^{-\frac{1}{2}}},$$

the DTFT of the sequence $\{w(k)\}$ is

$$W\left(e^{i\theta}\right) = \frac{\sin\left(\frac{1}{2}(2N+1)\theta\right)}{\sin\left(\frac{1}{2}\theta\right)} = \frac{(2N+1)\mathrm{sinc}\left(\frac{1}{2}(2N+1)\theta\right)}{\mathrm{sinc}\left(\frac{1}{2}\theta\right)}.$$

The graph of this function is illustrated in Figure 3.18. The discontinuities in the sampled signal produced by the ideal window function generate additional spectral components. This is referred to as *spectral leakage.* Windows which approach zero smoothly at either end of the sampled signal are used to minimize spectral leakage.

The first positive (negative) zero in its spectrum is the smallest positive (largest negative) value of θ such that $W(e^{i\theta}) = 0$. The *main lobe* of the window function is that part of the graph of $W(e^{i\theta})$ that lies between the first positive and first negative zero in $W(e^{i\theta})$. The *main lobe width* is the distance between the first positive and negative zeros in $W(e^{i\theta})$. As the length of the window increases, the main lobe narrows and its peak value rises, that is as N

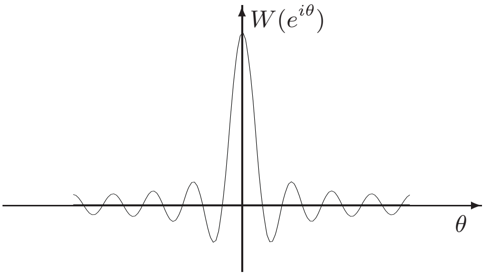


Figure 3.18: DTFT of the rectangular window sequence.

becomes large, $W(e^{i\theta})$ approaches an impulse, which is desirable. However, the main disadvantage is that the amplitudes of the side lobes also increase.

The use of any window leads to distortion of the spectrum of the original signal caused by the size of the side lobes in the window spectrum (producing oscillations in the filter response, known as *Gibb's phenomenon*) and the width of the window's main spectral lobe. The window function can be selected so that the amplitudes of the sides lobes are relatively small, with the result that the size of the oscillations are reduced; however, in general, the main lobe width does not decrease. Thus, in choosing a window, it is important to know the trade-off between having narrow main lobe and low side lobes in the window spectrum.

Some well-known window functions that taper to zero in a smooth fashion are defined and illustrated in Figures 3.19-3.22.

Bartlett window

$$w(k) = \left(1 - \frac{|k|}{N}\right) rect_N(k)$$

This is sometimes known as the *triangular* window.

Hann window

$$w(k) = \frac{1}{2}\left(1 + \cos\left(\frac{\pi k}{N}\right)\right) rect_N(k)$$

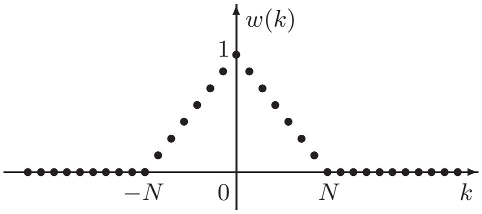


Figure 3.19: Bartlett window

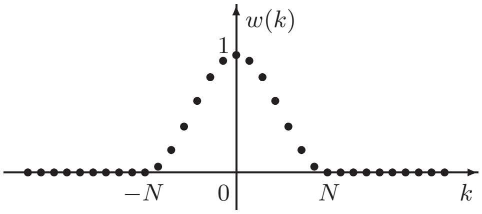


Figure 3.20: Hann window

Hamming window

$$w(k) = \left[0.54 + 0.46 \cos\left(\frac{\pi k}{N}\right)\right] rect_N(k)$$

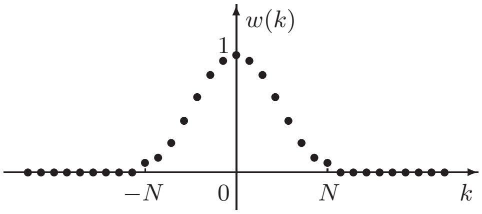


Figure 3.21: Hamming window

Blackman window

$$w(k) = \left[0.42 + 0.5 \cos\left(\frac{\pi k}{N}\right) + 0.08 \cos\left(\frac{2\pi k}{N}\right)\right] rect_N(k)$$

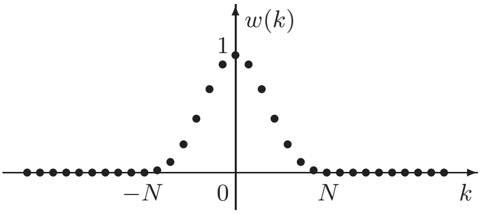


Figure 3.22: Blackman window

The normalised amplitude spectra for the Rectangular window, Bartlett window and Hann window are illustrated in Figure 3.23.

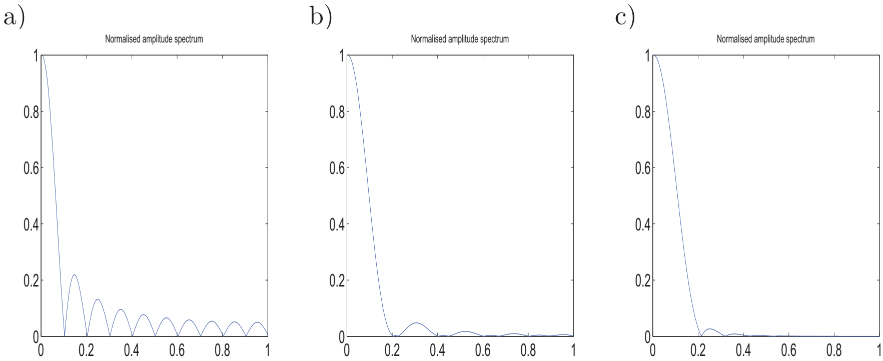


Figure 3.23: Amplitude spectra for a) Rectangular b) Bartlett c) Hann windows.

In particular, for further information, including more explanation on Gibb's phenomenon, see [Ziemer et al., 1998], and for more details on the spectral properties of the above windows see [Haddad and Parsons, 1991]. In MATLAB®, these windows can be generated using `rectwin`, `bartlett`, `hann`, `hamming`, and `blackman`.

With respect to speech processing, a segment of speech is obtained by windowing the speech signal. The number of samples in the speech segment should be chosen to be as 'small' as possible, due to the time-varying nature of the signal. On the other hand, to give a clear indication of periodicity when analysing the speech segment, the window must have a duration of at least two pitch periods of the voiced speech signal. Also, a rectangular window gives a much stronger indication of periodicity than, for example, the Hamming window, since the result of applying the Hamming window is to taper the speech segment.

3.9 Exercises

Exercise 3.1

A discrete-time system has a unit impulse response $h(k)$ given by

$$h(k) = \tfrac{1}{2}\delta(k) + \delta(k-1) + \tfrac{1}{2}\delta(k-2).$$

a) Find the system frequency response $H(e^{i\theta})$ and, using MATLAB® , plot the corresponding amplitude and phase spectra.

b) Determine the step response of the system.

Exercise 3.2 A discrete-time system with input $x(k)$ and output $y(k)$ is characterized by the difference equation: $y(k) = x(k+1) + x(k-1)$.

(a) Find the unit impulse response of this system. (b) Is the system causal? (c) Is the system linear? (d) Does this difference equation realization represent a FIR or IIR filter? (e) Find the frequency response of the system. (f) Is the system stable?

Exercise 3.3 A linear, time-invariant system is described $y(k) + \frac{1}{4}y(k-1) = 3x(k)$. Determine

a) the gain and phase-shift spectra;

b) the gain and phase-shift spectra when $\theta = 0, \frac{1}{3}\pi,\ \pi$ and, hence, find the steady-state output of the system for the input signal

$$x(k) = 5 + \tfrac{1}{3}\cos\left(\tfrac{1}{3}\pi k\right) - \sin\left(\pi k + \tfrac{1}{4}\pi\right).$$

Exercise 3.4 Determine the magnitude of the frequency response for the MA filter

$$y(k) = \frac{1}{M}\sum_{m=0}^{M-1} x(k-m),$$

where $M \in \mathbb{N}$, and, using MATLAB® , obtain its graph in the case when $M = 4$.

Exercise 3.5 A discrete-time, linear, time-invariant, causal filter is described by the difference equation:

$$y(k) = \tfrac{1}{2}y(k-1) + ax(k), \qquad a \in \mathbb{R}, \quad a > 0.$$

a) Find the system function, $H(z)$, and show that the filter is bounded-input bounded-output (BIBO) stable.

b) Select the parameter a such that $|H(e^{i0})| = 1$ and, for this case,

(i) show that the gain and phase-shift spectra are

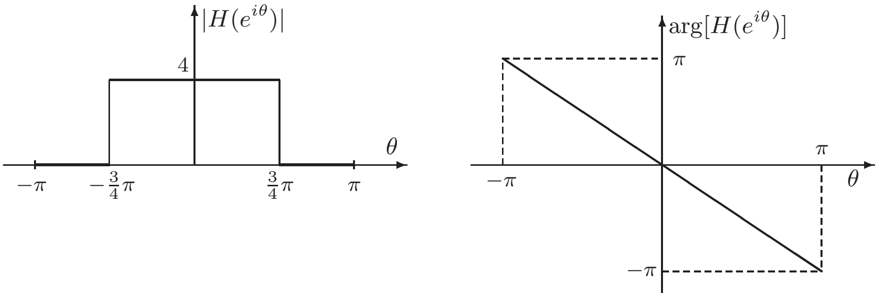
$$|H(e^{i\theta})| = \frac{1}{\sqrt{5 - 4\cos(\theta)}} \text{ and } \arg H(e^{i\theta}) = -\tan^{-1}\left(\frac{\sin(\theta)}{2-\cos(\theta)}\right);$$

(ii) find the steady-state output of the filter when the input signal is

$$2 - 3\cos\left(\tfrac{\pi}{3}k\right) + 6\sin\left(\pi k\right).$$

$\left[\text{You may use the result: } \tan^{-1}\left(1/\sqrt{3}\right) = \tfrac{\pi}{6}.\right]$

Exercise 3.6 Suppose the characteristics of $H(e^{i\theta})$, the frequency response for a digital filter, are shown below:



a) Determine $h(k)$, the unit impulse response.

b) Does $h(k)$ represent a FIR or IIR filter?

c) Is the filter causal?

Exercise 3.7 A system has the unit sample response $h(k)$ given by

$$h(k) = -\frac{1}{4}\delta(k+1) + \frac{1}{2}\delta(k) - \frac{1}{4}\delta(k-1).$$

a) Is the system stable?

b) Is the system causal?

c) Find the frequency response and plot the corresponding amplitude and phase spectra using MATLAB® .

Exercise 3.8 A glottal pulse filter is defined by $G(z) = (z+1)^2/z^3$. Determine, for this filter,

a) a difference equation realization;

b) the impulse and step responses;

c) the gain spectrum (amplitude response).

Exercise 3.9 The radiation of sound waves by the lips is often modelled, through sampling of the 'volume' velocity, by a discrete time-invariant causal system with system function

$$L(z) = \frac{\ell_1(1 - z^{-1})}{z^{-1} + \ell_2},$$

where $\ell_1 > 0$, $\ell_2 > 1$ and $\ell_2 \approx 1$. Explain why this system BIBO stable and find the gain spectrum for $L(z)$.

Exercise 3.10 A glottal shaping filter is modelled by a two-pole system with impulse response sequence $\{g(k)\}$, where $g(k) = \alpha^k \zeta(k) - \left(\frac{4}{5}\right)^k \zeta(k)$, α is a fixed parameter which satisfies $\frac{4}{5} < \alpha < 1$, and $\{\zeta(k)\}$ denotes the unit step sequence.

a) Find its frequency response $G(e^{i\theta})$ and show that

$$G(e^{i0}) = \frac{5\alpha - 4}{1 - \alpha} \qquad\qquad G(e^{i\pi}) = \frac{-(5\alpha - 4)}{9(1 + \alpha)}.$$

b) Determine a value of α such that

$$20 \log_{10} |G(e^{i0})| - 20 \log_{10} |G(e^{i\pi})| = 40 \text{ dB}.$$

Exercise 3.11 Show that a causal system modelled by

$$2y(k+1) + \alpha y(k) = x(k), \qquad k \in \mathbb{Z},\ \alpha \in \mathbb{R},$$

is BIBO stable if and only if $|\alpha| < 2$. For the case $\alpha = 1$, calculate the frequency response and, using MATLAB® , plot the amplitude response over the range $-2\pi \leq \theta \leq 2\pi$.

Exercise 3.12 Find the frequency response of the autoregressive causal filter

$$y(k) = ay(k-1) + x(k)\ , \quad |a| < 1.$$

Hence, or otherwise, determine the steady-state response to the input $x(k) = \cos(k\theta)\zeta(k)$.

Exercise 3.13 Find the **z**-transfer function for the system

$$8y(k+2) + 2y(k+1) - y(k) = x(k) + 2x(k+1), \qquad k \in \mathbb{Z},$$

which is causal, and verify that it is a stable system. Using MATLAB®, plot the amplitude response over the range $-2\pi \leq \theta \leq 2\pi$.

Exercise 3.14 Determine the amplitude and phase spectra for

$$f(t) = A[e^{-\alpha t}\zeta(t) - e^{\alpha t}\zeta(-t)], \quad \text{real } A,\ \alpha > 0.$$

Exercise 3.15 Determine the energy spectral density for the triangular pulse

$$f(t) = \begin{cases} 1 - |t|/T, & |t| < T \\ 0, & \text{otherwise.} \end{cases}$$

Exercise 3.16

Use Parseval's relation and the table of Fourier transforms to find the energy in the signals:

(a) $x(t) = \dfrac{1}{1+t^2}$ (b) $x(t) = \text{sinc}(t)$ (use the symmetry property).

Exercise 3.17

Using the symmetry property for the Fourier transform, show that the Fourier transform of
$f(t) = \frac{a}{\pi}\text{sinc}(at),\ a > 0$, is

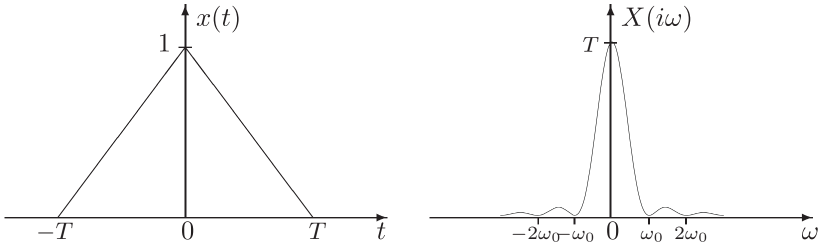
$$F(i\omega) = \begin{cases} 1, & |\omega| < a, \\ 0, & \text{otherwise.} \end{cases}$$

Hence, using the time-shift property and the identity $e^{i\theta} \equiv \cos(\theta) + i\sin(\theta)$, determine

$$g(t) = \mathcal{F}^{-1}\left\{G(i\omega)\right\} \overset{\text{def}}{=} \mathcal{F}^{-1}\left\{\cos(\omega)F(i\omega)\right\}.$$

How fast should $g(t)$ be sampled for perfect reconstruction, if $a = \frac{1}{2}\pi$?

Exercise 3.18 A signal $x(t)$ and its transform $X(i\omega)$ are shown below:



Suppose $x(t)$ is sampled every $0.01T$ seconds to produce a signal $x(n)$.

a) Sketch a graph of $\mathcal{F}\left\{x(nT)\right\}$.

b) Can $x(t)$ be reconstructed from $x(n)$?

c) If possible, determine the Nyquist sampling rate for $x(t)$.

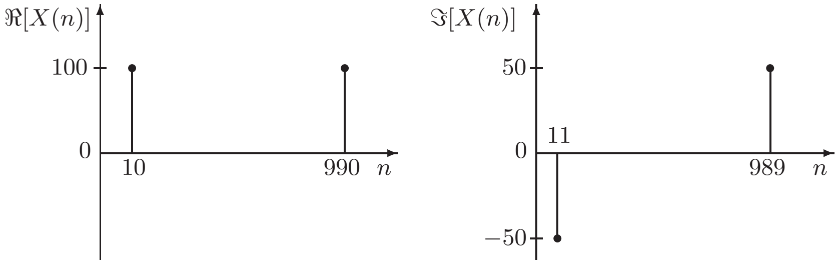
Exercise 3.19 Consider the signal $x(t) = 6\cos(10\pi t)$ sampled at 7Hz and 14Hz.

a) Find (i) $X(i\omega)$ (ii) $X_s(i\omega)$ when $f_s = 7$ and $f_s = 14$, and sketch the spectrum.

b) Suppose the reconstruction filter is an ideal low pass filter $H(i\omega)$, with bandwidth πf_s and amplitude response $(f_s)^{-1}$.
Sketch the graph of $X_s(i\omega)H(i\omega)$ when $f_s = 7$ and $f_s = 14$. In each case, reconstruct the continuous time signal. Explain your results.

Exercise 3.20 Determine the minimum number of signal samples that must be taken to evaluate the frequency spectrum of the signal, which has no frequency component higher than 2.5 kHz, whilst ensuring an analogue frequency resolution of at least 45 Hz. What is the minimum sampling period for the signal?

A signal $x(t)$ has been sampled at a rate of 50 samples/sec. to give a total of 1000 samples. A 1000-point DFT is taken and the resulting $\Re[X(n)]$, $\Im[X(n)]$ are shown in the diagrams below.



a) What is the time duration of the sampled segment of $x(t)$?

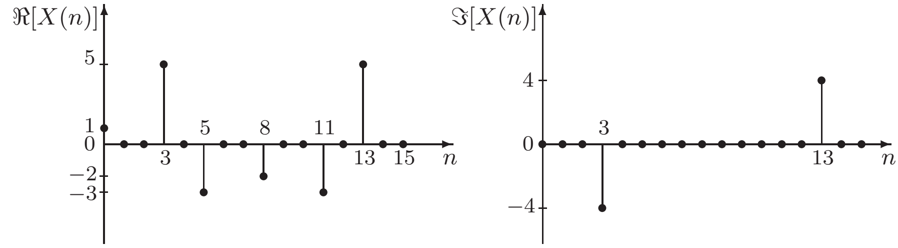
b) What is the digital frequency resolution?

c) What is the corresponding analogue frequency resolution in Hz?

d) What is the highest unaliased analogue frequency that one could see?

e) Determine the power spectrum for the signal $\{x(k) : k = 0,\ 1, \ldots,\ 999\}$.

Exercise 3.21 Consider a 16-point DFT shown in the following diagrams.



a) If the time samples are 0.03 sec. apart, what are the corresponding analogue radian frequencies and the associated analogue frequency resolution?

b) Find the power spectrum for the sequence $\{x(k) : k = 0,\ 1, \ldots,\ 15\}$.

Exercise 3.22 Using the `fft` MATLAB® command, confirm Parseval's relation for the sequence

$$x = \begin{bmatrix} i & 1+i & 2 & 1-i & -i \end{bmatrix}.$$

Exercise 3.23 Determine the power spectrum for the sequence $\{x(k)\}$, given:
$X(0) = 0 \qquad X(1) = \frac{3}{2} + i\frac{3}{2} \qquad X(2) = 1 \qquad X(3) = \frac{3}{2} - i\frac{3}{2}.$

Exercise 3.24 A third order filter is described by the difference equation

$$\begin{aligned} y(k) = &-1.1619y(k-1) - 0.6959y(k-2) - 0.1378y(k-3) + 0.0495x(k) \\ &- 0.1486x(k-1) + 0.1486x(k-2) - 0.0495x(k-3) \end{aligned}$$

and the input to the filter is the signal described by, for $0 \leq k \leq 100$, $k \in \mathbb{Z}$,

$$x(k) = \cos(0.3\pi k) - 2\cos(0.6\pi k) + \cos(0.7\pi k) + 4\sin(0.8\pi k) - 3\sin(0.9\pi k).$$

Using MATLAB®, plot the amplitude and phase spectra of the filter and determine whether the filter is low-pass or high-pass and confirm your analysis by plotting the amplitude spectrum of the filtered signal.

Chapter 4

Response of linear systems to random inputs

4.1 Introduction

In practice, many deterministic signals are contaminated by noise, giving rise to stochastic/random signals. Therefore, in this chapter, random processes are discussed. In order to analyse such signals, functions with special properties are utilised. In this chapter, the functions: autocorrelation, autocovariance, cross-correlation and cross-covariance are studied. These functions enable one to investigate the response of linear systems to random inputs.

As an application, autoregressive models and linear predictive filters are used to estimate certain speech parameters and formants, in non-nasal voiced speech, in speech processing.

Finally, speech quantization and encoding is investigated, using linear predictive filters, and the response of linear filters to quantization noise is examined.

4.2 Random processes

Only real random processes will be considered here.

Definition 4.1 *A* ***random*** *(or* ***stochastic****)* ***process*** *is a set of functions of some parameter (usually time) that has certain statistical properties.*

For a random signal, $x(t)$, its time behaviour could follow any of an infinite number of different paths. The set of all possible time behaviours of a signal is called an *ensemble*. An ensemble member is referred to as a *realization* of the random process. For any fixed t the value of $x(t)$ cannot be identified, although typically certain ranges of values will be more probable than others. In this manner, for every fixed t, $x(t)$ can be regarded as a *random variable.*

Definition 4.2 *If Ω is any set and x assigns a real number to every element $\omega \in \Omega$, then x is said to be a* ***random variable*** *and Ω is called the* ***sample space****.*

Information concerning random variables can be obtained by examining various statistics.
The *mean* (value) and *variance* of a random variable x are defined by

$$\begin{aligned} \mu_x = \overline{x} &\overset{\text{def}}{=} \mathcal{E}\left[x\right] \\ \sigma_x^2 = Var[x] &\overset{\text{def}}{=} \mathcal{E}\left[(x-\overline{x})^2\right], \end{aligned}$$

respectively, where $\mathcal{E}\left[\cdot\right]$ denotes *expected* or *average* value. The expected value operator satisfies the following rules: suppose $c_1,\ c_2$ are constants and x is a random variable, then

(a) $\mathcal{E}\left[c_1\right] = c_1$ (b) $\mathcal{E}\left[c_1 g_1(x) + c_2 g_2(x)\right] = c_1\mathcal{E}\left[g_1(x)\right] + c_2\mathcal{E}\left[g_2(x)\right].$

As a consequence of (a) and (b), it is easily shown that

$$\sigma_x^2 = \mathcal{E}\left[x^2\right] - \mu_x^2 = \overline{x^2} - \mu_x^2. \tag{4.1}$$

Another commonly used statistic is *standard deviation* which is defined by

$$\sigma_x \overset{\text{def}}{=} \{Var[x]\}^{\frac{1}{2}}.$$

If $\mu_x = 0$, then $Var[x] = \overline{x^2}$ is known as the *mean square* value and σ_x is known as the *root mean square* (r.m.s.) value. The mean is known as a *first order* statistic, whilst the variance is a *second order* statistic. Some other important averages of interest of a random process $x(t)$, with reference to second order statistics, are the *autocorrelation* and *autocovariance* functions.

4.3 Autocorrelation/autocovariance functions

The *autocorrelation function*, $R_{xx}(t_1, t_2)$, is defined by

$$R_{xx}(t_1, t_2) \overset{\text{def}}{=} \mathcal{E}\left[x(t_1)x(t_2)\right] = \overline{x(t_1)x(t_2)},$$

whilst the *autocovariance function*, $C_{xx}(t_1, t_2)$, is defined by

$$C_{xx}(t_1, t_2) \overset{\text{def}}{=} R_{xx}(t_1, t_2) - \mu_x(t_1)\mu_x(t_2).$$

Definition 4.3 *A random process, $x(t)$, is said to be* ***stationary in the wide-sense*** *(or* ***weakly stationary****) if its mean is a constant and the autocorrelation function, $R_{xx}(t_1, t_2)$, depends only on the time difference $\tau \overset{\text{def}}{=} t_2 - t_1$, that is*

$$\begin{aligned} \mathcal{E}\left[x(t)\right] &= \mu_x \\ \mathcal{E}\left[x(t)x(t+\tau)\right] &= R_{xx}(\tau). \end{aligned}$$

Assume a random process is stationary in the wide-sense and $\tilde{x}(t)$ is a realization of the random process. In practice, there is only one sample function available for analysis. Thus, $\tilde{x}(t)$ can be used to determine time statistics such as the *time average*:

$$\mu = \langle \tilde{x}(t) \rangle \overset{\text{def}}{=} \lim_{\tau\to\infty} \frac{1}{2\tau} \int_{-\tau}^{\tau} \tilde{x}(t)\,\mathrm{d}t,$$

time variance and *time autocorrelation function*:

$$\sigma^2 = \left\langle \tilde{x}^2(t) \right\rangle - \mu^2 \quad \text{and} \quad R_{\tilde{x}\tilde{x}}(\tau) = \langle \tilde{x}(t)\tilde{x}(t+\tau) \rangle\,.$$

For the discrete case, with $\tilde{x}(m)$ as the sampled value of $\tilde{x}(t)$, with sampling period T chosen so that $MT = \tau$,

$$\langle \tilde{x}(m) \rangle = \lim_{M\to\infty} \frac{1}{2M+1} \sum_{m=-M}^{M} \tilde{x}(m)$$

and

$$R_{\tilde{x}\tilde{x}}(k) = \lim_{M\to\infty} \frac{1}{2M+1} \sum_{m=-M}^{M} \tilde{x}(m)\tilde{x}(m+k).$$

Definition 4.4 *A wide-sense stationary random process is said to be* ***ergodic*** *to the nth order if an nth order ensemble statistic is equal to the corresponding time statistic.*

Remark 4.1 *In general, it is difficult to check whether a process is ergodic. In practice, if ensemble statistics are time-independent (in some sufficiently large time interval or the time interval of interest) and corresponding time statistics for a selection of realizations are reasonably in agreement, then the process may be assumed to be ergodic.*

Under an assumption of ergodicity, ensemble statistics for a random process can be estimated from an associated time statistic. For example, if a discrete random process is wide-sense stationary and ergodic to the second order, then

$$R_{xx}(k) = \mathcal{E}\left[x(m)x(m+k)\right] = \langle \tilde{x}(m)\tilde{x}(m+k)\rangle = R_{\tilde{x}\tilde{x}}(k),$$

for any realization $\tilde{x}(k)$.

4.3.1 Properties of the autocorrelation function

Concepts of *energy* and *power* can be used for classifying stochastic signals. Assuming a random process is wide-sense stationary and ergodic, the total *energy* of the discrete process $x(m)$ is defined as

$$E(x) \overset{\text{def}}{=} \sum_{m=-\infty}^{\infty} \tilde{x}^2(m).$$

For processes which have infinite energy, the concept of power is more useful. The *average power* of a random process $x(m)$ is defined as

$$\boxed{P_{av}(x) \overset{\text{def}}{=} \mathcal{E}\left[x^2(m)\right] = \lim_{M\to\infty} \frac{1}{2M+1} \sum_{m=-M}^{M} \tilde{x}^2(m).}$$

Remark 4.2 *If $x(m)$ has finite energy, then its average power is zero.*

Thus, if $x(m)$ is an ergodic process and wide-sense stationary, then

$$\boxed{P_{av}(x) = R_{xx}(0) = R_{\tilde{x}\tilde{x}}(0).}$$

For a wide-sense stationary, ergodic, discrete random process, the autocorrelation function $R_{xx}(k) = \mathcal{E}\left[x(m)x(m+k)\right]$ has the following properties:

(AF1) $R_{xx}(-k) = R_{xx}(k)$;

(AF2) $R_{xx}(0) = \overline{x^2(m)} = \sigma_x^2 + \mu_x^2$ (from (4.1));

(AF3) $|R_{xx}(k)| \leq R_{xx}(0)$;

(AF4) $R_{(x+y)(x+y)}(k) = R_{xx}(k) + R_{yy}(k) + \mathcal{E}\left[x(m)y(m+k)\right] + \mathcal{E}\left[x(m+k)y(m)\right]$;

(AF5) If $x(m)$ contains a periodic component, with period M, then $R_{xx}(k)$ also contains a periodic component, with period M;

(AF6) $-\sigma_x^2 + \mu_x^2 \leq R_{xx}(k) \leq \sigma_x^2 + \mu_x^2$. If $x(m)$ has non-zero mean, the term μ_x^2 is known as the *dc component* of $R_{xx}(k)$.

A typical graph of an autocorrelation function is shown in Figure 4.1.

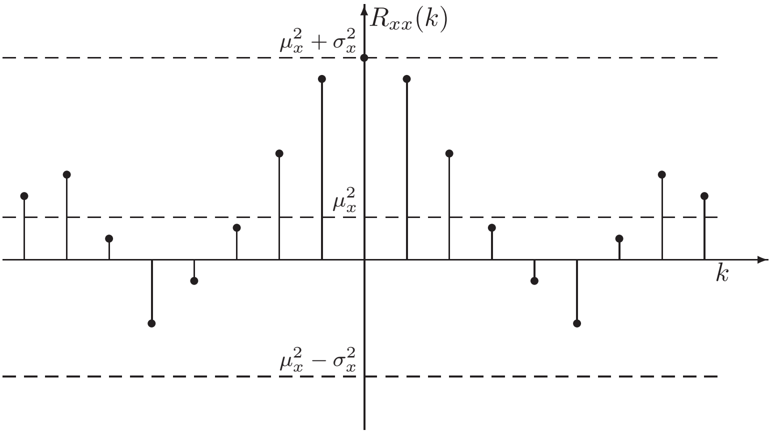


Figure 4.1: Typical graph of an autocorrelation function $R_{xx}(k)$.

Example 4.1 Obtain, using MATLAB®, graphs of the deteministic signal $\{x(k)\}$, where $x(k) = 50 + 10\sin(\frac{1}{10}\pi k)$ and $k \in \mathbb{N}$ with $1 \leq k \leq 500$, and its autocorrelation sequence. Also,

a) determine the mean value of the signal;

b) modify the graph, by a horizontal shift, so that the main peak occurs at $k = 0$;

c) overcome the masking of the periodicity in the signal by subtracting the value of the mean from the data and re-computing the autocorrelation sequence.

Solution: The following MATLAB® commands produce the graphs of the periodic sequence and its autocorrelation sequence, illustrated in Figure 4.2:

```
n=500; k=[1:n];           % set number of data points
x=50+10*sin(pi*k/10);     % generates periodic sequence
r=xcorr(x);               % determines autocorrelation sequence
subplot(211); plot(k,x); title('Periodic signal');
subplot(212); plot(r);
title('Graph of autocorrelation function');
```

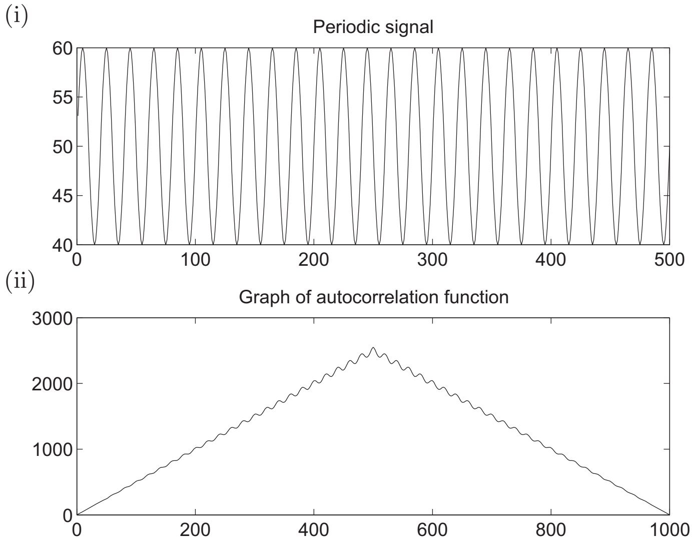


Figure 4.2: (i) Periodic signal. (ii) Autocorrelation sequence.

a) Using the MATLAB® command `mean(x)` gives the mean value of the signal as 50.0000.

b) A plot of the horizontal shift on the graph of the autocorrelation sequence can be obtained using the commands:

```
m=length(r);
y=r(((m-1)/2+1):m);
subplot(211); plot(y)
```

This graph is illustrated in Figure 4.3 (i).

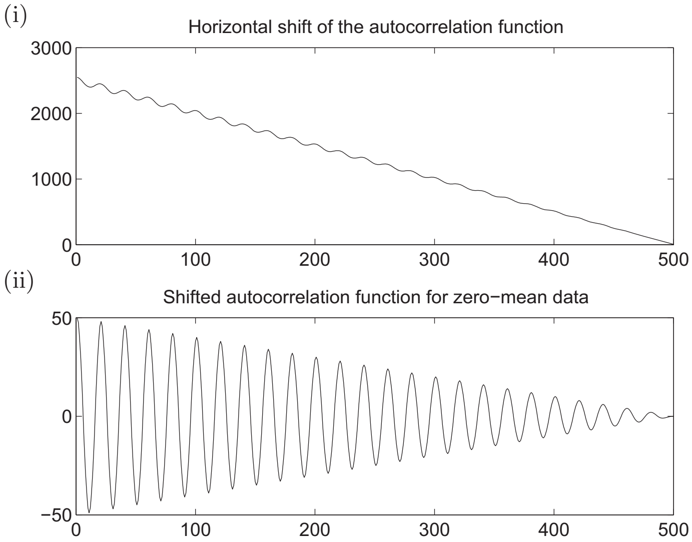


Figure 4.3: (i) Shifted autocorrelation sequence. (ii) Autocorrelation sequence with zero-mean data.

c) The MATLAB® commands:

```
xbar=mean(x)              % mean of the sequence
r=xcorr(x-xbar);          % or use:  r=xcov(x)
a=r((m-1)/2+1:m);         % autocorrelation sequence with
zero-mean data
subplot(212); plot(a);
title('Shifted autocorrelation function for zero-mean data');
```

produce the graph obtained in Figure 4.3 (ii), which clearly confirms the periodicity in the original signal.

The expected value of the estimate of the autocorrelation sequence obtained using MATLAB® is different from $R_{xx}(k)$ by a factor $N - |k|$, where N is the length of the sequence, as can be seen in Figure 4.3 (ii). This MATLAB® estimate is *biased*. An unbiased estimate is obtained using the MATLAB® command `xcorr(x-xbar,'unbiased')` (or `xcov(x,'unbiased')`), which is illustrated in Figure 4.4.

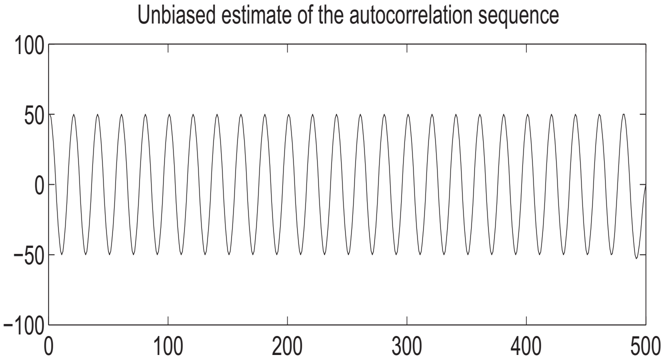


Figure 4.4: Unbiased estimate of the autocorrelation sequence for the zero-mean data.

□

4.3.2 Pitch period estimation for voiced speech

The autocorrelation function is very useful for estimating periodicities in signals (see §4.3.1 property (AF5)), including speech. In practice, a segment of speech $\{s(m);\ m = 0, 1, \ldots, M-1\}$ is available for analysis. The 'short-time' autocorrelation function (which is an estimate of the true autocorrelation function) is determined by

$$R_{ss}(m) = \sum_{j=0}^{M-1-m} s(j)s(j+m).$$

This will exhibit cyclic behaviour as a result of the quasi-periodic nature of the voiced speech segment. For voiced speech, peaks occur in $R_{ss}(m)$ approximately at multiples of the pitch period. Hence, the pitch period of a speech signal can be estimated by finding the location of the first maximum in the autocorrelation function. This is achieved by dividing each term of this sequence by the total energy of the speech segment to produce the normalised sequence $R_{ss}(m)/R_{ss}(0)$. The separation between adjacent peaks in this normalised sequence is also an estimate of the pitch period, which can be found using a peak-picking algorithm. If the value of this peak is at least 0.25, then the segment of speech is considered voiced with a pitch period equal to the value of m for which the normalised sequence has a maximum. If there are no strong periodicity peaks, then this indicates a lack of periodicity in the waveform, implying that the section of the speech signal is unvoiced.

Example 4.2 Over a short time-frame ($k = 1 : n$, $n = 1200$), a voiced speech signal is modelled as the output from a linear, causal, time-invariant filter, with system function

$$H(z) = \frac{1}{1 - 0.2z^{-1} - 0.23z^{-2} - 0.05z^{-3} - 0.12z^{-4}},$$

and the input to the filter is a weighted finite 'train' of delayed impulses

$$\left\{ 10 \sum_{j=1}^{m} \delta(k - (jT - 10)) \right\},$$

with $T = 80$ and $m = n/T$. Using MATLAB®, add zero-mean noise (with variance 4) to this signal via the MATLAB® command `randn` and plot the graph of the modelled noisy speech signal and its associated autocorrelation function. From the graph of the autocorrelation function, estimate the pitch period of the modelled speech signal, assuming the sampling period is 0.098 msec.

Solution: The following MATLAB® commands were used to obtain the graphs in Figure 4.5.

```
n=1200; T=80; m=n/T; k=[1:n]; % initial data
% impulse sequence of length T with impulse at k=10
p=[[1:T]-10 == 0];
% periodic repeat (period m) of impulse sequence
q=p’*ones(1,m);
r=10*q(:)’;          % weighted impulse train
a=[1.0 -0.2 -0.23 -0.05 -0.12];
b=[1];
y=filter(b,a,r)’;    % filter finite train of delayed impulses
% modelled voiced speech with additive noise
% (mean 0 and variance 4)
x=y+2*randn(n,1);
subplot(211);
plot(k,x), title(’Modelled noisy voiced speech signal’);
% autocorrelation function for zero-mean signal
w=xcov(x,’unbiased’); wlen=length(w);
z=w((wlen-1)/2+1:wlen);
% lag=(no.  of points)/4 for autocorrelation function
tau=1:length(z)/4;
subplot(212); plot(tau,z(1:length(tau))), grid,
title(’Autocorrelation function for signal with zero mean’)
```

From Figure 4.5 (ii), it is clear that there are dominant peaks at approximately $N = 80,\ 160,\ 240$, which suggests that the pitch period is approximately $80 \times 0.098\,\text{msec} \approx 7.84\,\text{msec}$.

□

Pitch period estimation using the power spectrum

Append the speech samples with $N - M$ zeros, where $N \geq M$ is a power of 2. Apply a N-point FFT on this new data sequence to produce the spectral sequence $\{\hat{s}(k);\ 0 \leq k \leq N-1\}$. An estimate of the power spectrum is then obtained by evaluating

$$\frac{1}{N}\sum_{k=0}^{N-1} |\hat{s}(k)|^2.$$

The inverse Fourier transform of the power spectrum then provides an estimate of the autocorrelation function. This leads to an estimate of the pitch period by the method described previously.

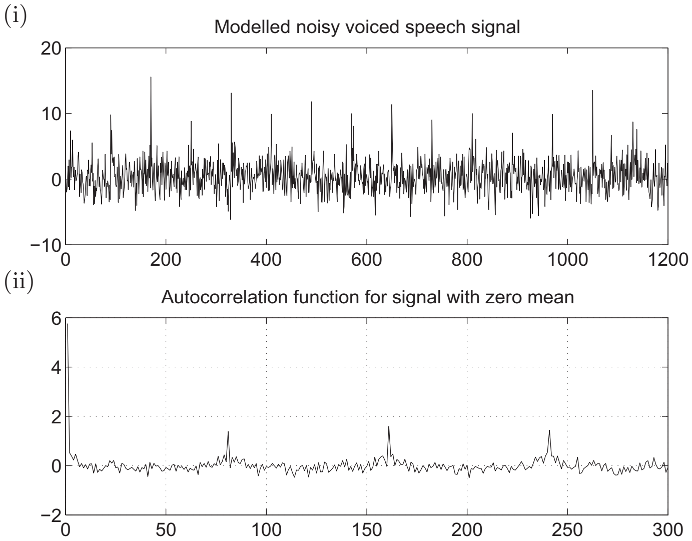


Figure 4.5: (i) Modelled noisy voiced speech. (ii) Autocorrelation function with zero-mean data.

4.4 Cross-correlation and cross-covariance functions

cross-covariance function There are other useful second order statistics. For two random processes $x(t)$ and $y(t)$, the *cross-correlation function* is defined by

$$R_{xy}(t_1, t_2) \overset{\text{def}}{=} \mathcal{E}\left[x(t_1)y(t_2)\right].$$

The *cross-covariance function* is defined by

$$C_{xy}(t_1, t_2) \overset{\text{def}}{=} R_{xy}(t_1, t_2) - \mu_x(t_1)\mu_y(t_2).$$

Two processes, $x(t)$ and $y(t)$, are said to be *uncorrelated* when

$$C_{xy}(t_1, t_2) = 0.$$

For wide-sense stationary processes,

$$R_{xy}(t_1, t_2) = R_{xy}(\tau), \qquad \text{where } \tau = t_2 - t_1,$$

and, if $x(t)$ and $y(t)$ are jointly ergodic,

$$R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} \tilde{x}(t)\tilde{y}(t+\tau)\ \mathrm{d}t.$$

For discrete processes that are wide-sense stationary and ergodic,

$$\begin{aligned} R_{xy}(k) &= \mathcal{E}\left[x(m)y(m+k)\right] = \langle \tilde{x}(m)\tilde{y}(m+k)\rangle \\ &= \lim_{M\to\infty} \frac{1}{2M+1} \sum_{m=-M}^{M} \tilde{x}(m)\tilde{y}(m+k). \end{aligned}$$

4.4.1 Properties of the cross-correlation function

Properties of $R_{xy}(k)$:

(C-CF1) $R_{xy}(-k) = R_{yx}(k)$;

(C-CF2) $|R_{xy}(k)| \leq \{R_{xx}(0)R_{yy}(0)\}^{\frac{1}{2}} \leq \frac{1}{2}\{R_{xx}(0) + R_{yy}(0)\}$;

(C-CF3) If $x(k)$, $y(k)$ are statistically independent, i.e. uncorrelated, then

$$\boxed{R_{xy}(k) = R_{yx}(k) = \mu_x\mu_y;} \tag{4.2}$$

(C-CF4) If both $x(k)$ and $y(k)$ contain a periodic component, with period M, then both $R_{xy}(k)$ and $R_{yx}(k)$ contain a periodic component, with period M.

Remark 4.3 *In view of the definition of $R_{xy}(k)$, property (AF4) of the autocorrelation function can be expressed as*

$$\boxed{R_{(x+y)(x+y)}(k) = R_{xx}(k) + R_{xy}(k) + R_{yx}(k) + R_{yy}(k)} \tag{4.3}$$

and, if x and y are uncorrelated, then

$$R_{(x+y)(x+y)}(k) = R_{xx}(k) + R_{yy}(k) + 2\mu_x\mu_y.$$

4.5 White-noise process

Definition 4.5 *A **white random process** is a wide-sense stationary, ergodic random process for which any two realizations are statistically independent.*

Remark 4.4 *If x and y are statistically independent, $\mathcal{E}\left[xy\right] = \mu_x\mu_y$.*

For a white random process,

$$\begin{aligned}R_{xx}(k) &= \mathcal{E}\left[x(m)x(m+k)\right]\\ &= \mathcal{E}\left[x(m)\right]\mathcal{E}\left[x(m+k)\right]\\ &= \mu_x^2,\end{aligned}$$

when $k \neq 0$. When $k = 0$, $R_{xx}(0) = \mathcal{E}\left[x^2(m)\right] = \sigma_x^2 + \mu_x^2$. Thus,

$$R_{xx}(k) = \mu_x^2 + \sigma_x^2\delta(k).$$

In particular, if the white process has zero mean, then

$$R_{xx}(k) = \sigma_x^2\delta(k).$$

Definition 4.6 *A random process, $n(m)$, is **white noise** if it is a white process and $\mu_n = 0$, in which case, **for white noise**,*

$$\boxed{R_{nn}(k) = \sigma_n^2\delta(k).}$$

4.6 AR models and linear predictive filters

When performing an analysis in the frequency domain, AR (parametric) modelling has a number of advantages over FFT (non-paramtric) techniques. FFT techniques assume some periodicity in the signal. In addition, a fast sampling rate must be applied and, for meaningful results, a large amount of data is required. Whereas AR modelling only assumes a weakly stationary process and the sampling rate can be less than the sampling rate using FFT techniques. Moreover, as the number of parameters is relatively small, less data is required, and an increased frequency resolution can be obtained when compared to FFT techniques.

The method of linear prediction is based on an AR model and the classical least squares method. With applications to speech processing, many authors have investigated the method of linear prediction, sometimes known as linear predictive coding (see [Goldberg and Riek, 2000, Haddad and Parsons, 1991], [Makhoul, 1975, Papamichalis, 1987] and [Rabiner and Schafer, 1978], for example. Moreover, some MATLAB® software for analysis-by-synthesis linear predictive coding is presented in [Ramamurthy and Spanias, 2009].

Suppose a discrete process has data $s(m)$, $m = 0, \ldots M-1$, where M is the number of signal samples. It is assumed that an appropriate model for the process is the AR model

$$\boxed{s(m) = x(m) + \sum_{k=1}^{p} a_k s(m-k),} \tag{4.4}$$

where a_k are the model parameters (possibly time-varying), $p \geq 1$ and $x(m)$ is the model input. If $s(m)$ is required for $m \geq M$, a linear predictor of the form

$$\tilde{s}(m) = \sum_{k=1}^{p} \alpha_k s(m-k),$$

where $\tilde{s}(m)$ is the predicted value of $s(m)$, α_k are the prediction coefficients and p is the order of the filter, may be used to predict $s(m)$. Using time averages instead of ensemble averages, the predictor coefficients may be determined by minimizing the mean square error

$$E = \sum_{m} |e(m)|^2,$$

where $e(m) = s(m) - \tilde{s}(m)$ is the error signal for the mth sample and it is assumed that the error signal is non-zero only for $0 \leq m \leq p+M-1$. A standard procedure (see, for example, [Rabiner and Schafer, 1978, Chapman et al., 1997] and [Deller et al., 2000]) can be used to determine a sufficient condition for minimizing the mean square error, with respect to the predictor coefficients. It is found that the predictor coefficients must satisfy a set of linear equations of the form

$$\boxed{\sum_{k=1}^{p} \alpha_k r(|i-k|) = r(i) \ , \qquad i = 1, \ldots p,} \tag{4.5}$$

where $r(i) = \sum_{m=0}^{M-1} s(m)s(m-i)$ is the short-time autocorrelation function, which are known as the *normal* equations. If the input to the model is white noise, the equations represented by (4.5) are sometimes referred to as the *Yule-Walker* equations. The matrix

$$\begin{bmatrix} r(0) & r(1) & r(2) & \cdots & r(p-1) \\ r(1) & r(0) & r(1) & \cdots & r(p-2) \\ r(2) & r(1) & r(0) & \cdots & r(p-3) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ r(p-1) & r(p-2) & r(p-3) & \cdots & r(0) \end{bmatrix},$$

known as the *autocorrelation matrix*, is a *Toeplitz* matrix, namely the matrix is symmetric and the elements along a diagonal from top left to bottom right are all equal. An efficient recursive method for solving equation (4.5) is the Durbin-Levinson algorithm (see [Rabiner and Schafer, 1978, Chapman et al., 1997]):

1.
$$\begin{aligned} e^{(0)} &= r(0) \\ k_1 &= r(1)/e^{(0)} \\ \alpha_1^{(1)} &= k_1 \\ e^{(1)} &= (1-k_1^2)e^{(0)} \end{aligned}$$

2. For $2 \leq i \leq p$

$$\begin{aligned}
k_i &= \left\{ r(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} r(i-j) \right\} / e^{(i-1)} \\
\alpha_i^{(i)} &= k_i \\
\alpha_j^{(i)} &= \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}, \quad 1 \leq j \leq i-1 \\
e^{(i)} &= (1 - k_i^2) e^{(i-1)}.
\end{aligned}$$

The prediction coefficients are the given by $\alpha_k^{(p)}, \quad k = 1, \ldots p$.

Remarks 4.5

(i) *The algorithm also generates prediction coefficients for predictors of order less than* p.

(ii) *The prediction coefficients can be determined using the MATLAB® command* `[c,e]=levinson(r,p)`, *where* $c = [c_1 \ c_2 \ldots c_p]$ *are the prediction coefficients for the AR model*

$$s(m) = x(m) - \sum_{k=1}^{p} c_k s(m-k), \tag{4.6}$$

e *is the...,* r *is the associated autocorrelation matrix and* p *is the order of the predictor. However, note that, in MATLAB®, the assumed model (4.6) is different to the model given in (4.4). Therefore, if using MATLAB® to determine the prediction coefficients for the model (4.4), the prediction coefficients are given by* $a_k = -c_k$ *for* $k = 1, \ldots p$.

In speech processing, the above method is known as the *autocorrelation* method. Another method, known as the *covariance* method, is based on the assumption that the signal $s(m)$ is not wide-sense stationary. However, for this method, signal values outside the interval $0 \leq m \leq M-1$ are required, specifically, $s(m)$ for $-p \leq m \leq M-1$. For more details on the covariance method see [Rabiner and Schafer, 1978].

Lattice filter formulation

In the i^{th} stage of the Durbin-Levinson algorithm, $\{\alpha_j^{(i)}, \quad j = 1, 2, \ldots i\}$ denotes the set of coefficients of the i^{th} order linear predictor. The prediction error, in this case, would be

$$e^{(i)}(m) = s(m) - \sum_{k=1}^{i} \alpha_k^{(i)} s(m-k) \ . \tag{4.7}$$

This can be interpreted as predicting $s(m)$ using the i samples $\{s(m-i+k), \quad k = 1, \ldots i\}$. Suppose, now, the i samples are used to predict $s(m-i)$. We may define the *backward* prediction error sequence

$$b^{(i)}(m) = s(m-i) - \sum_{k=1}^{i} \alpha_k^{(i)} s(m+k-i) \ . \tag{4.8}$$

It can be shown that (4.7) and (4.8) may be rewritten in the form :

$$e^{(i)}(m) = e^{(i-1)}(m) - k_i b^{(i-1)}(m-1) \tag{4.9}$$

$$b^{(i)}(m) = b^{(i-1)}(m-1) - k_i e^{(i-1)}(m). \tag{4.10}$$

Initially, $e^{(0)}(m) = b^{(0)}(m) \overset{\text{def}}{=} s(m)$. Since (4.9) represents a *forward* prediction using an i^{th} order predictor, the notation $e^{(i)}(m)$ is replaced by $f^{(i)}(m)$. The formulation :

$$f^{(i)}(m) = f^{(i-1)}(m) - k_i b^{(i-1)}(m-1)$$
$$b^{(i)}(m) = b^{(i-1)}(m-1) - k_i f^{(i-1)}(m)$$

is known as a *lattice* filter and the coefficients k_i are known as *reflection* or *PARCOR* (partial correlation) coefficients. Using this structure $e(m) = f^{(p)}(m)$. This structure is known to be stable provided $|k_i| \leq 1$ and can be illustrated diagramatically, as shown in Figure 4.7 (note that D represents delay). In particular, Burg used this lattice formulation, to develop a procedure based upon minimizing the sum of the mean square forward and backward prediction errors, that is Burg's method minimized

$$\sum_{m=0}^{M-1} [\{f^{(i)}(m)\}^2 + \{b^{(i)}(m)\}^2].$$

As in the Durbin-Levinson method, the prediction coefficients are given by $\alpha_k^{(p)}, \quad k = 1, \ldots p$.

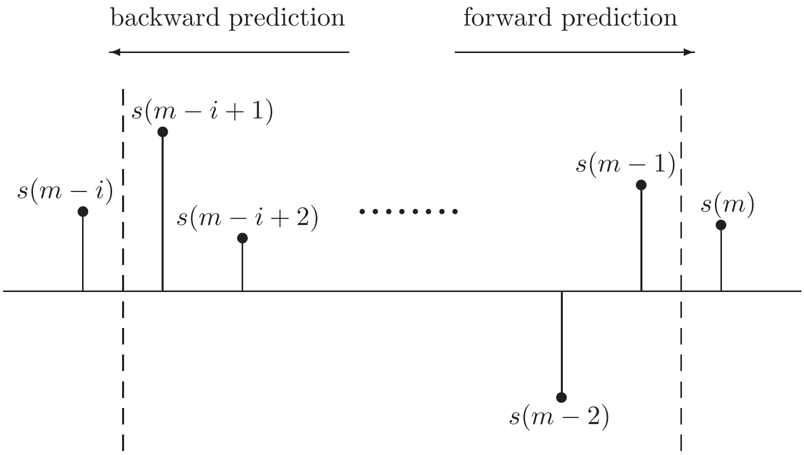


Figure 4.6: Forward and backward predictions

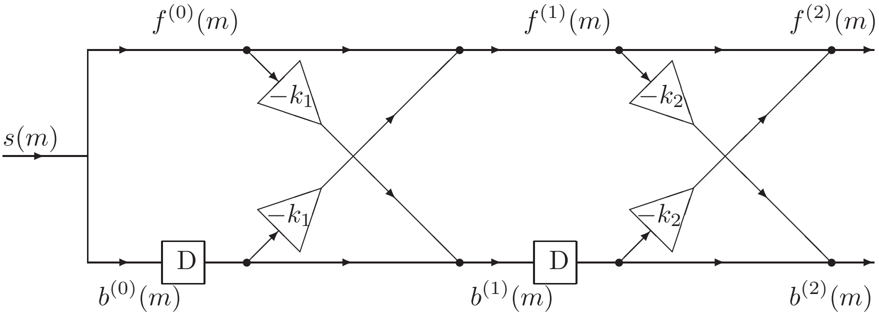


Figure 4.7: Lattice structure

Remarks 4.6

(i) *Note that Burg's algorithm does not give the exact solution to* (4.5)*; only an approximate solution. However, the method is computationally very efficient.*

(ii) *Details on lattice filters can be found in [Haddad and Parsons, 1991, Ingle and Proakis, 1997, Jackson, 1996],* and*[Rabiner and Schafer, 1978].*

4.6.1 Estimation of prediction coefficients for a speech signal and the speech gain parameter

For speech, an all-pole (AR) model is appropriate model for non-nasal voiced sounds. The composite effects of radiation, vocal tract and glottal excitation can be represented by a time-varying digital filter with system function of the form :

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

where G_s is known as the speech gain parameter. This model gives a good representation of speech sounds provided p is large enough. The filter is excited by either a impulse train for voiced speech or a random noise sequence for unvoiced speech. For this filter, the speech samples $s(m)$ are related to the excitation $x(m)$ by the difference equation

$$s(m) = G_s x(m) + \sum_{k=1}^{p} a_k s(m-k). \tag{4.11}$$

A linear predictor :

$$\tilde{s}(m) = \sum_{k=1}^{p} \alpha_k s(m-k)$$

may be used for this system, in which case the prediction error is

$$e(m) = s(m) - \sum_{k=1}^{p} \alpha_k s(m-k).$$

If the speech signal fits the model (4.11) exactly and if $\alpha_k = a_k$, then $e(m) = G_s x(m)$, that is, for voiced speech, $e(m)$ would consist of a train of impulses.

This suggests a method (known as *linear predictive coding*) for the determination of the pitch period of a voiced speech segment. First, the autocorrelation, or covariance, or lattice filter, method may be used to find the predictor coefficients. The error signal $e(m)$ is then determined and the difference between consecutive peaks in $e(m)$ is an estimate of the pitch period. By matching the energy in the signal with the energy in the predicted samples, an analysis shows that (see, for example [Rabiner and Schafer, 1978, Chapman et al., 1997] and, in addition, Chapter 10 in [Ingle and Proakis, 1997]) the speech gain parameter can be estimated from

$$G_s^2 \approx r(0) - \sum_{k=1}^{p} \alpha_k r(k) \ . \tag{4.12}$$

The estimated result (4.12) also holds for the case of unvoiced speech when the input is assumed to be white noise with unity variance.

4.6.2 Formant estimation for voiced speech

As stated earlier, a simple model for voiced speech is excitations produced by the vocal chords and filtered by the vocal tract. For an adult male, the resonances produced by the vocal tract are approximately 1kHz apart. However, the location of these resonances (or formants) can be changed by moving the tongue and lips. Normally, only the first three formants are of importance (in practice). Formant estimation is often performed using frequency-domain analysis. The segment of speech, obtained by windowing, must be long enough that the individual harmonics can be resolved, but short enough that the speech signal can assumed to be approximately stationary. Normally, a window is chosen to be at least two or three pitch periods long. The pitch period varies from a maximum of approximately 1/80s in adult males to a minimum of approximately 1/400s in adult females. For example, three periods of 100Hz voiced speech is approximately 30ms. The signal is usually sampled in the range 8-13kHz and the number of data samples is chosen to be a power of two. In the spectrum, the narrow peaks are the harmonics of the excitation, whilst the peaks in the envelope of the peaks in the spectrum indicate the formants due to the vocal tract.

Assuming an all-pole model,

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

the model parameters and speech gain parameter G_s can be estimated (as $\tilde{a}_k$ and $\tilde{G}_s$) using an appropriate technique, such as the *method of linear prediction* (see, for example, [Chapman et al., 1997]). Then, the spectrum can be obtained by evaluating $H(z)$ on the unit circle $z = e^{i\theta}$. One method of achieving this is to divide $\tilde{G}_s$ by the DFT of the sequence $\{1,\ -\tilde{a}_1,\ -\tilde{a}_2, \ldots,\ -\tilde{a}_p,\ 0,\ 0,\ 0, \ldots\}$, where the sequence is padded with zeros so that the length of the sequence, say N, is a power of two, to obtain

$$\frac{\tilde{G}_s}{1 - \sum_{k=1}^{p} \tilde{a}_k e^{-i\frac{2\pi n}{N}k}}, \qquad n = 0,\ 1, \ldots,\ N-1.$$

For sufficiently large N, the magnitude of the resulting sequence yields a high-resolution representation of the speech gain spectrum

$$\left|H\left(e^{i\theta}\right)\right| \approx \frac{\tilde{G}_s}{\left|1 - \sum_{k=1}^{p} \tilde{a}_k e^{-i\theta k}\right|}.$$

The formants are estimated by detecting the peaks in the spectrum envelope.

The peaks in the spectrum envelope can be accentuated by evaluating the spectrum on a circle with radius $\rho < 1$. This technique is important for the case in which two formants are close to each other. This modification can be achieved quite easily by premultiplying the linear prediction parameters by ρ before computing the DFT.

An example in which the log-amplitude spectrum has been determined using a linear predictive filter is illustrated in Figure 4.8.
Using $f = n/(NT)$ to determine frequency in Hertz, the formants are estimated to be 386Hz, 2699Hz and 4048Hz. Log-amplitude spectra, determined using a) the FFT; and b) the power spectrum, are illustrated in Figure 4.9.

4.7 Response of linear time-invariant systems with random inputs

The general problem is: given an input to a linear, time-invariant system (with zero initial conditions), which is a member of a wide-sense stationary, ergodic, discrete random process, find the cross-correlation function, $R_{xy}(k)$, between

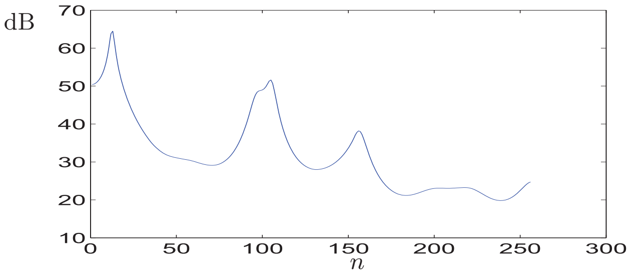


Figure 4.8: Log-amplitude spectrum using linear predictive coding.

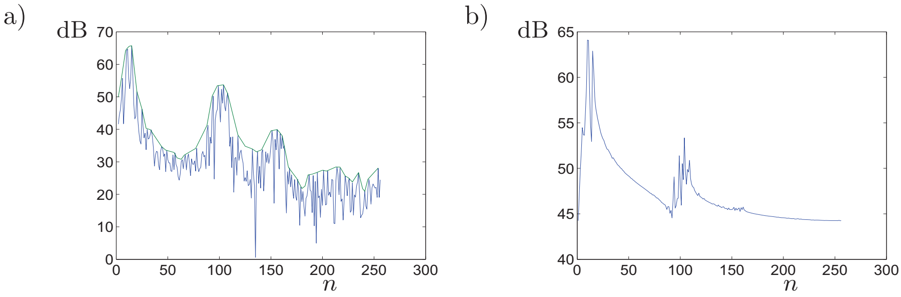


Figure 4.9: a) Log-amplitude spectrum and envelope b) Log-magnitude power spectrum.

the input, $x(k)$, and the output, $y(k)$, and, also, the autocorrelation function, $R_{yy}(k)$, in terms of $R_{xx}(k)$ and the unit impulse response, $h(k)$.

Proposition 4.1 *Suppose $x(k)$ is a wide-sense stationary random process. If, for each realization $\tilde{x}(k)$,*

$$\tilde{y}(k) = \sum_{m=-\infty}^{\infty} w(m)\tilde{x}(m,k),$$

where $w(m)$ is a finite-energy, deterministic waveform, then $y(k)$ is wide-sense stationary.

Proof: The proof of Proposition 4.1 is provided in Appendix D.

□

Let $x(k)$, a wide-sense stationary, real random process, be the input to a linear, time-invariant system with impulse response $h(k)$. Then

$$y(k) = \sum_{j=-\infty}^{\infty} h(j)x(k-j),$$

and, from the proof of Proposition 4.1,

$$\boxed{\mu_y = \mu_x \sum_{j=-\infty}^{\infty} h(j).}$$

Also,

$$\begin{aligned}
R_{xy}(k) = \mathcal{E}\left[x(m)y(m+k)\right] &= \mathcal{E}\left[\sum_{j=-\infty}^{\infty} h(j)x(m+k-j)x(m)\right] \\
&= \sum_{j=-\infty}^{\infty} h(j)\mathcal{E}\left[x(m+k-j)x(m)\right] \\
&= \sum_{j=-\infty}^{\infty} h(j)R_{xx}(k-j) \\
&= (h * R_{xx})(k).
\end{aligned}$$

Note that, since R_{xx} is an even function,

$$R_{yx}(k) = R_{xy}(-k) = (h * R_{xx})(-k) = h(-k) * R_{xx}(k). \tag{4.13}$$

Therefore,

$$\begin{aligned}
R_{yy}(k) &= \mathcal{E}\left[y(m)y(m+k)\right] \\
&= \mathcal{E}\left[y(m)\sum_j h(j)x(m+k-j)\right], \\
&= \sum_j h(j)\mathcal{E}\left[y(m)x(m+k-j)\right] \\
&= \sum_j h(j)R_{yx}(k-j) \\
&= (h * R_{yx})(k) \\
&= h(k) * \left[h(-k) * R_{xx}(k)\right], \quad \text{from (4.13).}
\end{aligned}$$

Thus, we have the important results:

$$\boxed{R_{xy}(k) = (h * R_{xx})(k)} \tag{4.14}$$

and

$$\boxed{R_{yy}(k) = [h(k) * h(-k)] * R_{xx}(k).} \tag{4.15}$$

Remark 4.7 *If the mean and variance of x are known and the mean of y is known, then the variance of y can be determined from*

$$\begin{aligned}\sigma_y^2 &= R_{yy}(0) - \mu_y^2 \\ &= ((h * h) * R_{xx})(0) - \mu_y^2 \\ &= (\mu_x^2 + \sigma_x^2)(h * h)(0) - \mu_y^2.\end{aligned}$$

Note that

$$h(k) * h(-k) = \sum_{j=-\infty}^{\infty} h(j)h(j-k)$$

and, in particular,

$$(h * h)(0) = \sum_{j=-\infty}^{\infty} h^2(j). \tag{4.16}$$

Example 4.3 White noise, with variance σ^2, is input to a linear, causal, time-invariant filter with system function

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

a) Show that, for $k \geq 0$, the cross-correlation of the output, y, with the input, x, is

$$R_{yx}(k) = \tfrac{1}{2}\sigma^2\delta(k).$$

b) Show that the autocorrelation function for the output, $R_{yy}(k)$, satisfies the recurrence relation

$$4R_{yy}(k) = \sigma^2\delta(k) - 3R_{yy}(k-1), \qquad k \geq 0. \tag{4.17}$$

c) Using the recurrence relation in (4.17), find the average power in the output signal.

d) Find the variance in the output signal in terms of σ.

Solution:

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

and, therefore, $h(k) = \frac{1}{2}\left(-\frac{3}{4}\right)^k \zeta(k)$. Since $\{x\}$ is white noise, $R_{xx}(k) = \sigma^2\delta(k)$ and so it follows, from (4.13) and using property (CD4) of §1.3, that
$$R_{yx}(k) = h(-k) * R_{xx}(k) = \sigma^2 \tfrac{1}{2}\left(-\tfrac{3}{4}\right)^{-k}\zeta(-k).$$
To determine $R_{yx}(k)$ for $k \geq 0$, note that
$$\zeta(-k) = \begin{cases} 1, & k \leq 0, \\ 0, & k > 0, \end{cases} \qquad \text{and so, for } k \geq 0,\ \zeta(-k) = \delta(k).$$

Thus, for $k \geq 0$, $R_{yx}(k) = \frac{1}{2}\sigma^2\left(-\frac{3}{4}\right)^{-k}\delta(k) = \frac{1}{2}\sigma^2\delta(k)$.

b) Since
$$H(z) = \frac{Y(z)}{X(z)} = \frac{2}{4+3z^{-1}} \qquad \Longrightarrow \qquad \left(1+\tfrac{3}{4}z^{-1}\right)Y(z) = \tfrac{1}{2}X(z),$$

then $y(k) + \frac{3}{4}y(k-1) = \frac{1}{2}x(k)$.
Now, by definition, $R_{yy}(k) = \mathcal{E}\left[y(m)y(m+k)\right]$, but
$$y(m+k) = -\tfrac{3}{4}y(m+k-1) + \tfrac{1}{2}x(m+k)$$

and so
$$R_{yy}(k) = \mathcal{E}\left[y(m)\left(\tfrac{1}{2}x(m+k) - \tfrac{3}{4}y(m+k-1)\right)\right] = \tfrac{1}{2}R_{yx}(k) - \tfrac{3}{4}R_{yy}(k-1).$$

Hence, using the result of part a), it follows that, for $k \geq 0$,
$R_{yy}(k) = \frac{1}{4}\sigma^2\delta(k) - \frac{3}{4}R_{yy}(k-1)$.

c) The average power in the output signal can be determined from $P_{av}(y) = R_{yy}(0)$.
$k=0$: $R_{yy}(0) = \frac{1}{4}\sigma^2 - \frac{3}{4}R_{yy}(-1) = \frac{1}{4}\sigma^2 - \frac{3}{4}R_{yy}(1)$, since R_{yy} is even
$k=1$: $R_{yy}(1) = -\frac{3}{4}R_{yy}(0)$
Hence, $R_{yy}(0) = \frac{1}{4}\sigma^2 + \frac{9}{16}R_{yy}(0) \iff R_{yy}(0) = \frac{1}{4}\sigma^2/\frac{7}{16} = \frac{4}{7}\sigma^2$.

d) Using the difference equation obtained in (4.17),

$$\begin{aligned}\mu_y = \mathcal{E}\left[y(m)\right] &= \mathcal{E}\left[-\tfrac{3}{4}y(m-1) + \tfrac{1}{2}x(m)\right] \\ &= -\tfrac{3}{4}\mathcal{E}\left[y(m-1)\right] + \tfrac{1}{2}\mathcal{E}\left[x(m)\right] = -\tfrac{3}{4}\mu_y + \tfrac{1}{2}\mu_x.\end{aligned}$$

Since $\mu_x = 0$, it follows that $\mu_y = 0$ and so, since $R_{yy}(0) = \mu_y^2 + \sigma_y^2$, it follows that

$$\sigma_y^2 = R_{yy}(0) - \mu_y^2 = R_{yy}(0) = \tfrac{4}{7}\sigma^2.$$

□

If $x(k) = f(k) + g(k)$, where $f(k)$ is a deterministic signal and $g(k)$ is a member of a wide-sense stationary random process, the output is

$$y(k) = (h * (f + g))(k) = v(k) + w(k),$$

where $v(k)$ is the deterministic output signal and $w(k)$ is a member of a wide-sense stationary random process. The 'amount' of noise present in the output signal can be quantified by using the output *signal-to-noise* (S/N) ratio, defined by

$$\boxed{(\mathrm{S/N})_{\mathrm{out}} = \frac{P_{av}(v)}{P_{av}(w)} = \frac{R_{vv}(0)}{R_{ww}(0)}.} \tag{4.18}$$

Sometimes this ratio is expressed in *decibels* (dB), i.e.

$$10\log_{10}\left(\frac{P_{av}(v)}{P_{av}(w)}\right).$$

A 'high' value of the (S/N) ratio (much greater than 1) indicates that the amount of noise present is relatively low compared to the signal content, namely it indicates a relatively clear signal; whilst a low value of the (S/N) ratio implies that the signal is contaminated by a significant amount of noise.

If $g(k)$ is an uncorrelated, zero-mean random process, then

$$R_{xy}(k) = \left(h * \left(R_{ff} + R_{gg}\right)\right)(k)$$

and

$$R_{yy}(k) = \left[h(k) * h(-k)\right] * \left(R_{ff} + R_{gg}\right)(k).$$

Now consider a linear, time-invariant system with input $x(k) = f(k) + g(k)$, where both $f(k)$ and $g(k)$ are members of random processes which are wide-sense stationary. If the impulse response is $h(k)$, then the output is

$$y(k) = ((f+g) * h)(k).$$

Invoking (4.3),

$$R_{xy}(k) = (h * R_{xx})(k) = (h * (R_{ff} + R_{fg} + R_{gf} + R_{gg}))\,(k)$$

and

$$R_{yy}(k) = [h(k) * h(-k)] * (R_{ff} + R_{fg} + R_{gf} + R_{gg})\,(k).$$

If f and g are from different sources, i.e. the signals are uncorrelated, then, as a consequence of (4.2),

$$R_{gf}(k) = \mu_g \mu_f = R_{fg}(k).$$

Hence,

$$\boxed{R_{xy}(k) = (h * (R_{ff} + R_{gg}))\,(k) + 2\mu_f \mu_g} \tag{4.19}$$

and

$$\boxed{R_{yy}(k) = [h(k) * h(-k)] * [(R_{ff} + R_{gg})\,(k) + 2\mu_f \mu_g]\,.} \tag{4.20}$$

For this case, an important problem is the construction of a linear filter (the *Wiener* filter), whose impulse response, $h(k)$, is designed to minimize

$$R_{ee}(0) = \mathcal{E}\left[e^2(k)\right],$$

where $e(k) \overset{\text{def}}{=} y(k) - f(k)$ denotes the error signal, i.e. the effect of noise is minimized.

Example 4.4 Consider a discrete system with impulse response

$$h(k) = \tfrac{3}{2}\delta(k) + 2\delta(k-1) + \tfrac{1}{2}\delta(k-2)$$

which has an input consisting of the deterministic sequence $\{\zeta(k)\}$ with additive, uncorrelated, white noise $\{n(k)\}$, whose variance is $\frac{1}{2}$.

a) Find the output signal independent of noise.

b) Determine the output signal-to-noise ratio.

Solution:

a) $f(k) = (h * \zeta)(k) = [\frac{3}{2}\delta(k) + 2\delta(k-1) + \frac{1}{2}\delta(k-2)] * \zeta(k)$.
Since, for any $x(k)$ and $m \in \mathbb{Z}$,

$$x(k) * \delta(k-m) = \sum_{j=-\infty}^{\infty} x(k)\delta(k-m-j) = x(k-m)$$

and so

$$f(k) = \tfrac{3}{2}\zeta(k) + 2\zeta(k-1) + \tfrac{1}{2}\zeta(k-2).$$

b) Let $g(k) = (h * n)(k)$, then, from (4.18),

$$(\mathrm{S/N})_{\mathrm{out}} = \frac{R_{ff}(0)}{R_{gg}(0)}.$$

Now, by definition,

$$R_{ff}(0) = \mathcal{E}\left[f^2(k)\right] = \lim_{M\to\infty} \frac{1}{2M+1} \sum_{m=-M}^{M} f^2(m).$$

Since $f(k) = 0$, $k < 0$, and $f(0) = \frac{3}{2}$, $f(1) = \frac{7}{2}$, $f(k) = 4$, for $k \geq 2$, then

$$\mathcal{E}\left[f^2(k)\right] = \lim_{M\to\infty} \frac{1}{2M+1} \left[\left(\tfrac{3}{2}\right)^2 + \left(\tfrac{7}{2}\right)^2 + (M-1)4^2\right] = 8.$$

Also, in view of (4.15) and (4.16),

$$R_{gg}(0) = ((h * h) * R_{nn})(0) = \tfrac{1}{2} \sum_{j=-\infty}^{\infty} h^2(j) = \tfrac{1}{2}\left[\left(\tfrac{3}{2}\right)^2 + 2^2 + \left(\tfrac{1}{2}\right)^2\right] = \tfrac{13}{4}.$$

Thus,

$$(\mathrm{S/N})_{\mathrm{out}} = 8/\left(\tfrac{13}{4}\right) = \tfrac{32}{13}.$$

□

4.8 Power spectral density

Consider a wide-sense stationary random process $x(k)$. In general, the discrete Fourier transform of a realization $\tilde{x}(k)$ of the random process $x(k)$ will not exist. However, defining

$$\tilde{x}_N(k) \overset{\text{def}}{=} \begin{cases} \tilde{x}(k), & -N \leq k \leq N \\ 0, & \text{otherwise,} \end{cases}$$

the DTFT of $\tilde{x}_N(k)$, namely

$$\tilde{X}_N(e^{i\theta}) = \sum_{k=-N}^{N} \tilde{x}_N(k)e^{-i\theta k},$$

exists and is a random variable. The *power spectral density*, which measures the power present in a signal as a function of frequency, is defined as

$$S_{xx}(\theta) \overset{\text{def}}{=} \lim_{N\to\infty} \left\{ \frac{1}{2N+1} \mathcal{E}\left[\left| \tilde{X}_N(e^{i\theta}) \right|^2 \right] \right\}.$$

An important theorem concerning the power spectral density is the following.

Theorem 4.1 (Wiener-Khinchine) *Suppose $x(k)$ is a wide-sense stationary random process, then*

$$S_{xx}(\theta) = \sum_{k=-\infty}^{\infty} R_{xx}(k)e^{-i\theta k}.$$

Remarks 4.8

(i) *Sometimes $S_{xx}(\theta) \overset{\text{def}}{=} S_{xx}(z)\big|_{z=e^{i\theta}}$, where $S_{xx}(z) = \mathcal{Z}[\{R_{xx}(k)\}]$, is used as a definition for $S_{xx}(\theta)$.*

(ii) *Note that since R_{xx} is an even function, i.e. $R_{xx}(-k) = R_{xx}(k)$, the region of convergence of $S_{xx}(z)$ has the form $\{z \in \mathbb{C} : a < |z| < a^{-1},\ \text{with } 0 < a < 1\}$.*

From Theorem 4.1, since

$$S_{xx}(\theta) = \mathcal{Z}[\{R_{xx}(k)\}]\Big|_{z=e^{i\theta}},$$

$$R_{xx}(k) = \frac{1}{2\pi i} \oint_C S_{xx}(z) z^{k-1} \, \mathrm{d}z,$$

where C is the contour $z = e^{i\theta}$. With $z = e^{i\theta}$,

$$R_{xx}(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\theta) e^{i\theta k} \, \mathrm{d}\theta. \tag{4.21}$$

This same technique can be used to obtain (3.2) from (3.1). In particular, putting $k = 0$ in (4.21),

$$R_{xx}(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\theta) \, \mathrm{d}\theta.$$

However, $R_{xx}(0) = P_{av}(x)$ (see §4.3.1) and, therefore, it is for this reason that $S_{xx}(\theta)$ is known as the power spectral density.
Properties of the power spectral density are:

(PSD1) $S_{xx}(-\theta) = S_{xx}(\theta)$ for a real process;

(PSD2) For real θ, $S_{xx}(\theta)$ is real and $S_{xx}(\theta) \geq 0$ for all θ;

(PSD3) $S_{xx}(\theta + 2\pi n) = S_{xx}(\theta), \quad n \in \mathbb{Z}$;

(PSD4) $\displaystyle \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\theta) \, \mathrm{d}\theta = R_{xx}(0) = \overline{x^2(k)} = P_{av}(x)$.

Corresponding to the cross-correlation function $R_{xy}(k)$, the *cross spectral density* is defined by

$$S_{xy}(\theta) \overset{\text{def}}{=} S_{xy}(z)\Big|_{z=e^{i\theta}}, \quad \text{where } S_{xy}(z) = \mathcal{Z}\left[\{R_{xy}(k)\}\right].$$

Since $R_{xy}(k) = R_{yx}(-k)$,

$$\begin{aligned} S_{xy}(z) &= \mathscr{Z}\left[\{R_{yx}(-k)\}\right] \\ &= \sum_{k=-\infty}^{\infty} R_{yx}(-k)z^{-k} \\ &= \sum_{m=-\infty}^{\infty} R_{yx}(m)z^{m} \\ &= \sum_{m=-\infty}^{\infty} R_{yx}(m)(z^{-1})^{-m} \\ &= S_{yx}(z^{-1}). \end{aligned}$$

Hence,

$$S_{xy}(\theta) = S_{yx}(z^{-1})\Big|_{z=e^{i\theta}}.$$

The cross spectral density $S_{xy}(\theta)$ is not necessarily a real, even, or positive function of θ.

Example 4.5 For the modelled voiced speech data of Example 4.2, determine and plot the power spectral density against frequency (in Hz) using MATLAB®, by specifying a Welch spectrum estimator (using the command: `spectrum.<ESTIMATOR>`) and selecting power spectral density (using the command `psd`), given the sampling frequency 10.204 kHz. Hence, estimate the first three formants for the voiced speech data.

Solution: The graph of the power spectral density, illustrated in Figure 4.8, was obtained using the MATLAB® commands:

```
n=1200; T=80; m=n/T; k=[1:n]; % initial data
% impulse sequence of length T with impulse at k=10
p=[[1:T]-10 == 0];
% periodic repeat (period m) of impulse sequence
q=p'*ones(1,m);
r=10*q(:)';        % weighted impulse train
a=[1.0 -0.2 -0.23 -0.05 -0.12];
b=[1];
y=filter(b,a,r)';  % filter finite train of delayed impulses
% modelled voiced speech with additive noise
% (mean 0, variance 4)
x=y+2*randn(n,1);
figure(1); subplot(211);
plot(k,x), title('Modelled noisy voiced speech signal');
%
fs=10204;
h=spectrum.welch        % creates a Welch spectral estimator
pxx=psd(h,x,'fs',fs)    % calculates power spectral density
subplot(212); plot(pxx)
```

The formants are obtained from the location of the peaks in the power spectral density and the first three formants are estimated as 287 Hz, 1273 Hz, and 1833 Hz.

□

4.9 Response of a linear, time-invariant filter

It is assumed that the filter is causal, that is $h(k) = 0$ for $k < 0$. If $x(k)$ is the input (a wide-sense stationary random process) and $y(k)$ the output to the filter, then, as shown by (4.15),

$$R_{yy}(k) = h(k) * h(-k) * R_{xx}(k).$$

Thus,

$$\boxed{S_{yy}(z) = H(z)H(z^{-1})S_{xx}(z),} \tag{4.22}$$

where $H(z)$ is the system function for the digital filter. In particular, the average power in the output is

$$\overline{y^2(k)} = R_{yy}(0) = \mathcal{Z}^{-1}\left[H(z)H(z^{-1})S_{xx}(z)\right]\Big|_{k=0}.$$

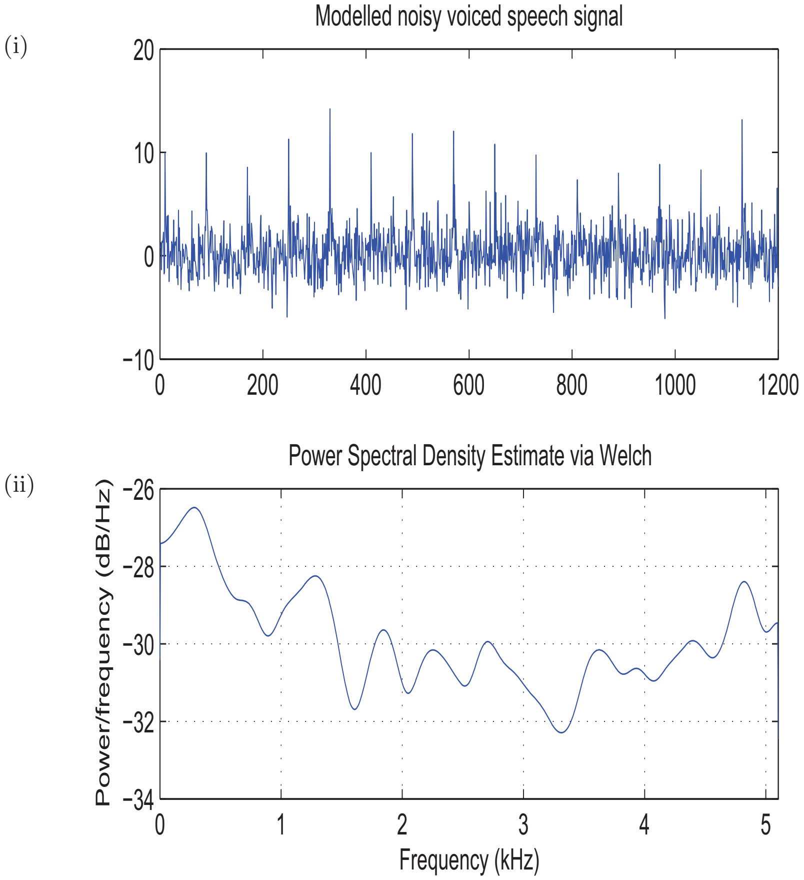


Figure 4.10: (i) Modelled noisy voiced speech. (ii) Power spectral density.

Remark 4.9 *Since the region of convergence of $H(z)H(z^{-1})$ is of the form $\{z \in \mathbb{C} : \alpha < |z| < \alpha^{-1}\}$, then, if the region of convergence of $S_{xx}(z)$ is $\{z \in \mathbb{C} : \beta < |z| < \beta^{-1}\}$, the region of convergence for $H(z)H(z^{-1})S_{xx}(z)$ is $\{z \in \mathbb{C} : \gamma < |z| < \gamma^{-1}\}$, where $\gamma = \max\{\alpha, \beta\}$.*

Also, from (4.14),

$$R_{xy}(k) = (h * R_{xx})(k)$$

and, therefore,

$$\boxed{S_{xy}(z) = \mathcal{Z}\left[\{R_{xy}(k)\}\right] = H(z)S_{xx}(z).} \tag{4.23}$$

Consider the case when the input to the filter consists of a deterministic input $f(k)$ and uncorrelated noise $n(k)$, assumed to be wide-sense stationary. From (4.22), the z-transform of the power spectral density of the output noise $g(k) = (n * h)(k)$ is given by

$$S_{gg}(z) = \mathcal{Z}\left[\{R_{gg}(k)\}\right] = H(z)H(z^{-1})S_{nn}(z)$$

and, using (4.23), the z-transform of the cross spectral density of the input and output noise is given by

$$S_{ng}(z) = \mathcal{Z}\left[\{R_{ng}(k)\}\right] = H(z)S_{nn}(z).$$

When the input $x(k)$ to the filter consists of a random signal $f(k)$ plus zero-mean, uncorrelated noise $n(k)$, which are both members of wide-sense stationary random processes, then, if $y(k)$ is the output, it follows from (4.19) and (4.20) that

$$R_{yy}(k) = [h(k) * h(-k)] * (R_{ff} + R_{nn})(k)]$$

and

$$R_{xy}(k) = (h * (R_{ff} + R_{nn}))(k).$$

Hence, taking z-transforms,

$$S_{yy}(z) = H(z)H(z^{-1})(S_{ff} + S_{nn})(z)$$

and

$$S_{xy}(z) = H(z)[S_{ff}(z) + S_{nn}(z)].$$

Example 4.6 Consider a discrete, linear, time-invariant system with system function

$$H(z) = \frac{1}{1 - 0.6z^{-1}}, \quad |z| > 0.6,$$

which has as input the deterministic signal $x(k) = \sum_{m=0}^{\infty} \delta(k - m)$ plus uncorrelated noise, $n(k)$ (assumed to be wide-sense stationary), with power spectral density given by

$$S_{nn}(z) = \frac{2z}{(z + \frac{1}{2})(z + 2)}, \quad \left\{z \in \mathbb{C} : \tfrac{1}{2} < |z| < 2\right\}.$$

a) Show that the z-transform of $\{x(k)\}$ is $X(z) = \dfrac{z}{z-1}$, $|z| > 1$.

b) Find the deterministic output $f(k) = (h * x)(k)$, where $h(k)$ is the impulse response.

c) Given that the output noise, $g(k)$, is given by $g(k) = (h * n)(k)$, determine the output noise power spectral density, $S_{gg}(z)$, and indicate the region of convergence.

d) Find the mean square output noise fluctuations $\overline{g^2(k)}$.

Solution:

a) By definition, $x(k) = \zeta(k)$ and hence, using tables, $X(z) = \dfrac{z}{z-1}$, $|z| > 1$.

b)
$$\begin{aligned}\mathcal{Z}\left[\{f(k)\}\right] &= H(z)X(z) \\ &= \left(\frac{z}{z - 0.6}\right)\left(\frac{z}{z-1}\right) = \frac{z^2}{(z - 0.6)(z - 1)}\end{aligned}$$

with region of convergence:

$$\{z \in \mathbb{C} : |z| > 0.6\} \cap \{z \in \mathbb{C} : |z| > 1\} = \{z \in \mathbb{C} : |z| > 1\}.$$

Therefore,

$$\{f(k)\} = \mathcal{Z}^{-1}\left[F(z)\right] = \mathcal{Z}^{-1}\left[z\left(\frac{\frac{5}{2}}{z-1} - \frac{\frac{3}{2}}{z - 0.6}\right)\right]$$

and so

$$f(k) = \left[\tfrac{5}{2} - \tfrac{3}{2}(0.6)^k\right]\zeta(k).$$

c) From (4.22),

$$\begin{aligned} S_{gg}(z) &= H(z)H(z^{-1})S_{nn}(z) \\ &= \left(\frac{z}{z-0.6}\right)\left(\frac{z^{-1}}{z^{-1}-0.6}\right)\left(\frac{2z}{(z+\frac{1}{2})(z+2)}\right) \\ &= \left(\frac{z}{z-0.6}\right)\left(\frac{1}{1-0.6z}\right)\left(\frac{2z}{(z+\frac{1}{2})(z+2)}\right) \\ &= \frac{2z^2}{(z-0.6)(1-0.6z)(z+\frac{1}{2})(z+2)}. \end{aligned}$$

The region of convergence for $S_{nn}(z)$ is $\left\{z \in \mathbb{C} : \frac{1}{2} < |z| < 2\right\}$, whilst the region of convergence for $H(z)H(z^{-1})$ is $\left\{z \in \mathbb{C} : \frac{3}{5} < |z| < \frac{5}{3}\right\}$. Hence, the region of convergence for $S_{gg}(z)$ is

$$\left\{z \in \mathbb{C} : \tfrac{1}{2} < |z| < 2\right\} \cap \left\{z \in \mathbb{C} : \tfrac{3}{5} < |z| < \tfrac{5}{3}\right\} = \left\{z \in \mathbb{C} : \tfrac{3}{5} < |z| < \tfrac{5}{3}\right\}.$$

d) Now

$$\overline{g^2(k)} = R_{gg}(0) = \mathcal{Z}^{-1}\left[\frac{2z^2}{(z-0.6)(1-0.6z)(z+\frac{1}{2})(z+2)}\right]\Bigg|_{k=0}.$$

Hence,

$$\begin{aligned} R_{gg}(0) &= \sum_{\substack{\text{all poles of } S_{gg}(z)z^{-1} \\ \text{inside } C}} [\text{residue of } S_{gg}(z)z^{-1}] \\ &= \frac{2z}{(1-0.6z)(z+\frac{1}{2})(z+2)}\Bigg|_{z=0.6} + \frac{2z}{(z-0.6)(1-0.6z)(z+2)}\Bigg|_{z=-0.5} \\ &= \frac{\frac{6}{5}}{\left(\frac{16}{25}\right)\left(\frac{15}{10}\right)\left(\frac{13}{5}\right)} - \frac{1}{\left(-\frac{11}{10}\right)\left(\frac{13}{10}\right)\left(\frac{3}{2}\right)} \\ &= \frac{125}{132} \approx 0.9470\,. \end{aligned}$$

□

4.10 Application to system identification

system identification Consider a linear, time-invariant, causal filter with system function $H(z)$ and suppose it is required to identify $H(z)$ or, equivalently, the impulse response, $\{h(k)\}$. If the input to the filter, $x(k)$, is a wide-sense stationary random process and the output is $y(k)$, then, from (4.23),

$$S_{xy}(z) = H(z)S_{xx}(z).$$

In addition, suppose the input is a zero-mean white process, i.e. white noise, then

$$R_{xx}(k) = \sigma_x^2 \delta(k) \qquad \text{and} \qquad S_{xx}(z) = \sigma_x^2.$$

For this input,

$$H(z) = S_{xy}(z)/\sigma_x^2.$$

Thus, the system function for the filter can be identified by computing σ_x^2 and $S_{xy}(z)$.
Alternatively, since

$$\begin{aligned} R_{xy}(k) &= (h * R_{xx})(k) \\ &= (h * (\sigma_x^2 \delta))(k) \\ &= \sigma_x^2 h(k), \end{aligned}$$

the unit impulse response can be determined by evaluating $R_{xy}(k)/\sigma_x^2$.

4.11 Speech quantization and encoding

Some techniques, generally known as *waveform encoding*, are considered for digitizing an analogue speech waveform. A digital signal can be generated by sampling the continuous speech waveform to produce a sequence of sampled values. In addition, in order to obtain a digital representation, it is necessary to *quantize* the sampled values, that is restrict the sampled values to some specified finite set of values. This process can give rise to quantization error (sometimes known as quantization noise). Further details can be found in [Bellanger, 1990, Goldberg and Riek, 2000, Haddad and Parsons, 1991] and [Oppenheim et al., 1999].

4.11.1 Pulse code modulation (PCM)

Let $s(t)$ denote an analogue speech waveform and $f_{\max}$ the highest frequency in the spectrum of $s(t)$. The speech signal is then sampled at a frequency f_s which, to reduce the effect of aliasing, is chosen to satisfy $f_s > 2f_{\max}$. Each sampled value, say $s(n)$, is then quantized to one of 2^b finite set of values, where b is the number of bits used to represent each sample value, and, thus, each sample value of the digital signal can be represented by a sequence of b bits. Therefore, the rate at which the digitized speech signal can be transmitted is bf_s bits per second (bps). A mathematical model of the quantization process is

$$\tilde{s}(n) = s(n) + q(n),$$

where $\tilde{s}(n)$ represents the quantized value of $s(n)$, and $q(n)$, the *quantization error*, is assumed to be additive noise. It is also assumed that the quantization is uniform in the sense that the absolute difference between consecutive quantized levels is always the same. In this case, assuming that the number of quantized levels is sufficiently large, the quantized error is characterised statistically by the uniform probability density function

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

where $\Delta = 2^{-b}$ is the step size of the quantizer (for more details on this, see [Deller et al., 2000]). Since the quantised error is characterised statistically by the uniform probability density function, its mean is zero and, therefore,

$$\sigma_q^2 = \mathcal{E}\left[q^2\right] = P_{av}(q).$$

It is known that the mean square value of the quantization error, sometimes known as *quantization noise power*, is

$$\sigma_q^2 = \mathcal{E}\left[q^2\right] = \frac{\Delta^2}{12} = \frac{1}{12}(2^{-2b}),$$

and, in decibels (with units dB), the mean square value of the quantization error is

$$\begin{aligned} 10\log_{10}\left(\frac{\Delta^2}{12}\right) &= 10\left[\log_{10} 2^{-2b} - \log_{10} 12\right] \\ &= 10\left[b\log_{10} 2^{-2} - \log_{10} 12\right] \approx -6.02b - 10.79 \text{ (dB)}. \end{aligned}$$

Thus, the quantization noise decreases approximately by $6.02b$ dB/bit for each bit used in a uniform quantizer. High-quality speech requires a minimum of 12 bits per sample. Therefore, for high quality speech, a bit rate of $12\,bf_s$ is required and the quantization noise power is approximately $-6.02 \times 12 - 10.79 = -83.03$ dB.

Usually, speech signals have the property that small signal amplitudes occur more frequently than large signal amplitudes. Due to this characteristic, a non-uniform quantizer, which produces more closely spaced levels at low signal amplitudes and more widely spaced levels at large signal amplitudes, is more useful. A non-uniform quantized effect can be obtained by compressing the signal amplitude, utilizing a nonlinear device, and then using a uniform quantizer. An example of a nonlinear *compressor* is the logarithmic compressor, defined by

$$|y| = \frac{1 + \log(A|s|)}{1 + \log(A)} \qquad \text{and} \quad A = 87.56,$$

which is used in European communications systems, and is known as the A-*law*. In encoding a speech waveform at a sampling rate of 8000 samples per second and with 8 bits per sample, a bit rate of 64000 bps can be achieved using a logarithmic compressor for the PCM encoded speech. However, a uniform PCM quantizer requires 12 bits per sample to achieve a similar level of fidelity. To reconstruct the signal from the quantized values, obtained using a logarithmic compressor, an inverse logarithmic relation is used to revert to the 'normal' form of the signal amplitude.

4.11.2 Differential pulse code modulation (DPCM)

There is usually a significant correlation between successive samples of a speech signal, when sampled at the Nyquist rate or higher. Thus, the change in the average amplitude between successive samples is relatively small. DCPM is a scheme that exploits this characteristic and it results in a lower bit rate for the speech signal. Differential pulse code modulation is designed so that it encodes the differences between successive samples, rather than the samples themselves. In general, few bits are required to represent the differences between samples, since these differences are usually smaller than the actual signal amplitudes. Thus, in DPCM, the input to the quantizer is the signal obtained by the difference $e(n) = s(n) - \tilde{s}(n)$, where $s(n)$ is the unquantized input sample and $\tilde{s}(n)$ is an estimate of the input sample. The method of linear prediction (see Section 4.6) can be used to estimate $\tilde{s}(n)$. Thus, it is assumed that $\tilde{s}(n)$ can be estimated using a p^{th} predictor, namely

$$\tilde{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k),$$

where α_k are the prediction coefficients. The general design technique for DPCM is the following. The 'difference' signal $\{e(n)\}$ is input to the quantizer, which has output $\{d(n)\}$, and each value of the quantized prediction error is encoded into a sequence of binary digits and transmitted over a channel to a receiver. Suppose the input to the predictor is $\{x(n)\}$ and the output is $\{\tilde{x}(n)\}$. Then $x(n)$ is designed to represent $\{s(n)\}$ modified by the quantization process, and satisfies $x(n) = d(n) + \tilde{x}(n)$. Finally, the input to the quantizer is defined to be the difference $e(n) = s(n) - \tilde{x}(n)$. If the prediction is good, then a quantizer with a given number of levels can be adjusted to give a smaller quantization error than would be possible when quantizing the input directly. A block diagram illustrating a DPCM encoder is given in Figure 4.11. To decode, the same linear predictor that was used in the encoding technique is synthesized and its output, $\tilde{x}(n)$, is used to determine the signal sample modified by the quantization process, $x(n)$, with $x(n) = d(n) + \tilde{x}(n)$. A block diagram illustrating a DPCM decoder is given in Figure 4.12.

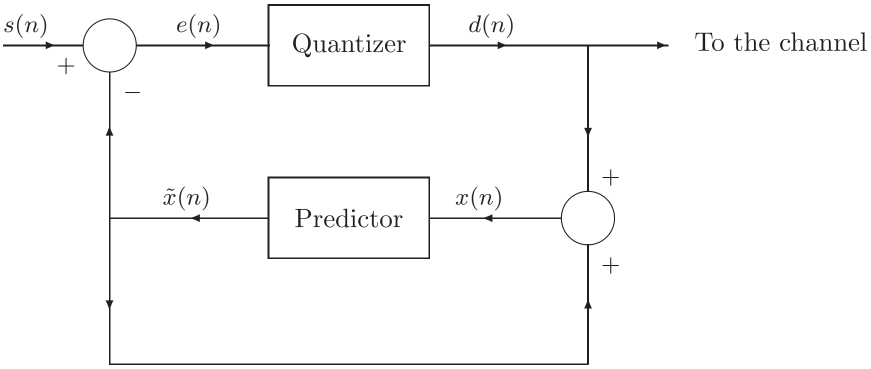


Figure 4.11: DPCM encoder.

4.11.3 Response of linear filters to quantization noise

When an analogue signal is passed through an analogue-to-digital converter this gives rise to quantization errors (usually known as *quantization noise*). Consider the effect of the quantization error, say $\{q(k)\}$, being passed through a linear, time-invariant, causal, stable filter with system function $H(z)$. It is known

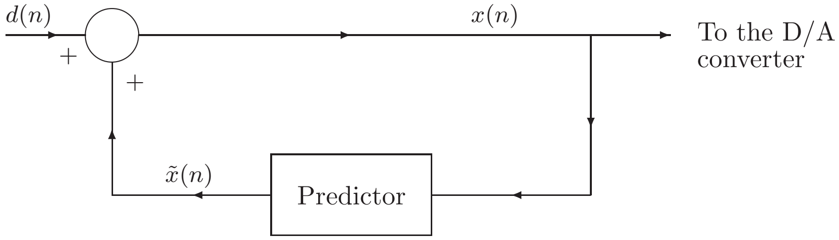


Figure 4.12: DPCM decoder.

that if the quantization error has zero mean and $\{h(k)\}$ is the impulse response sequence associated with $H(z)$, then the (average) power in the quantization output noise, say $\{q_0(k)\}$, is given by

$$P_{av}(q_0) = R_{q_0 q_0}(0) = \sigma_q^2 \sum_{k=0}^{\infty} h^2(k) = \sigma_q^2 R_{hh}(0),$$

where $\{R_{hh}(k)\} = \mathcal{Z}^{-1}\left[H(z)H\left(z^{-1}\right)\right]$ denotes the autocorrelation sequence associated with $\{h(k)\}$ and σ_q^2 is the variance of the quantization noise.

Example 4.7 The output of a 4-bit quantizer is input to a first order, causal, IIR filter with system function

$$H(z) = \frac{1}{z - \frac{1}{4}}.$$

Assuming the quantization error has zero mean, estimate the quantization noise power in the output to the filter.

Solution: For a b-bit quantizer the quantization noise power is $\sigma_q^2 = \frac{1}{12}(2^{-2b})$ and so, for a 4-bit quantizer, $\sigma_q^2 = \frac{1}{12}(2^{-8}) \approx 3.255 \times 10^{-4}$.

Since $H(z) = \left(z - \frac{1}{4}\right)^{-1}$, $H\left(z^{-1}\right) = \left(z^{-1} - \frac{1}{4}\right)^{-1} = z/\left(1 - \frac{1}{4}z\right)$ and so

$$\begin{aligned} H(z)H\left(z^{-1}\right) &= \frac{z}{\left(z - \frac{1}{4}\right)\left(1 - \frac{1}{4}z\right)} = z\left\{\frac{-\frac{1}{4}}{\left(z - \frac{1}{4}\right)(z-4)}\right\} \\ &= \frac{16}{15}\left(\frac{z}{z - \frac{1}{4}} - \frac{z}{z-4}\right) \end{aligned}$$

with region of convergence $\left\{z \in \mathbb{C} : \frac{1}{4} < |z| < 4\right\}$.
From Table B.1 (Appendix B),

$$\mathcal{Z}^{-1}\left[H(z)H\left(z^{-1}\right)\right] = \tfrac{16}{15}\left[\left(\tfrac{1}{4}\right)^k \zeta(k) - 4^k \zeta(-k-1)\right].$$

Therefore,

$$R_{hh}(0) = \tfrac{16}{15}\left[\left(\tfrac{1}{4}\right)^k \zeta(k) - 4^k \zeta(-k-1)\right]\Big|_{k=0} = \tfrac{16}{15}.$$

Hence,

$$P_{av}(q_0) \approx 3.255 \times 10^{-4} \times \tfrac{16}{15} \approx 3.5 \times 10^{-4}.$$

□

4.12 Exercises

Exercise 4.1 The input, $x(k)$, to a linear, stable, discrete system, with unit sample response $h(k)$, is assumed to be a white process with zero mean and variance σ_x^2. Express the autocorrelation function for the output, $R_{yy}(k)$, in terms of its sample response and the variance of the input, and write down the average power in the output.

Exercise 4.2 The input to a linear, time-invariant system, with impulse response function $h(k) = \delta(k) + \frac{1}{2}\delta(k-3)$, is $\{x(k) + n(k)\}$, where $\{x(k)\}$ is a deterministic signal and $\{n(k)\}$ is uncorrelated random white noise with variance $\frac{1}{4}$. The output from the system is of the form $\{y(k) + m(k)\}$, where the sequence $\{y(k)\}$ represents the deterministic output signal and $\{m(k)\}$ is the noise output.
Find the cross-correlation function $R_{nm}(k)$ and show that the output noise autocorrelation function is

$$R_{mm}(k) = \tfrac{1}{8}\delta(k+3) + \tfrac{5}{16}\delta(k) + \tfrac{1}{8}\delta(k-3).$$

Exercise 4.3 A linear, causal, time-invariant, digital filter, with system function

$$H(z) = \frac{1}{1 - \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}},$$

has white noise with variance σ_x^2 as input.

a) Show that, for $k \geq 0$, the cross-correlation of the output with the input is:

$$R_{yx}(k) = \sigma_x^2 \delta(k).$$

b) Show that $R_{yy}(k) = \mathcal{E}\left[y(m)y(m+k)\right]$, the autocorrelation for the output, satisfies the recurrence relation:

$$R_{yy}(k) = \tfrac{1}{4}R_{yy}(k-1) + \tfrac{1}{8}R_{yy}(k-2) + \sigma_x^2\delta(k), \qquad k \geq 0.$$

c) Using the recurrence relation, above, with $k = 0,\ 1,\ 2$, find the mean square value of the output.

Exercise 4.4 A discrete filter, known as a 'averager', has output

$$y(k) = \tfrac{1}{2}[x(k) + x(k-1)]$$

in response to an input $x(k)$. If the input is a white process with mean μ_x and variance σ_x^2, determine the average power in the output of the 'averager'.

Exercise 4.5 Given that the input, $x(k)$, to a linear, discrete system, with $h(k) = \delta(k) - \delta(k-3)$, is the sum of the deterministic signal $f(k) = k\zeta(k)$ and uncorrelated white noise $n(k)$ with $R_{nn}(k) = \delta(k)$, find

a) $R_{ww}(k)$ and $R_{nw}(k)$

b) $\overline{w^2(k)}$ and $g(k)$

c) the output signal-to-noise ratio,

where $x(k) = f(k) + n(k)$, $y(k) = g(k) + w(k)$ is the output, $g(k)$ is the deterministic output signal, and $w(k)$ is the noise output.

Exercise 4.6 Show that, if $x(k) = A + \hat{x}(k)$, where A is a constant and $\mathcal{E}\left[\hat{x}(k)\right] = 0$, then

$$R_{xx}(k) = A^2 + R_{\hat{x}\hat{x}}(k).$$

Exercise 4.7 If $x(k)$ is a wide-sense stationary, random process and $h(k)$ is the unit impulse response of a linear, time-invariant system with output $y(k)$, which is also wide-sense stationary, show that

$$\mu_y = \mu_x \sum_{k=-\infty}^{\infty} h(k).$$

Exercise 4.8 Prove that $R_{xy}(k) = R_{yx}(-k)$.

Exercise 4.9 Evaluate the autocorrelation function

$$R_{xx}(\tau) = \int_{-\infty}^{\infty} x(s)x(s-\tau)\,\mathrm{d}s$$

for the deterministic, finite-energy signal $t \mapsto x(t) \overset{\text{def}}{=} e^{-t}\zeta(t)$.

Exercise 4.10

a) Which of the following functions could be a power spectral density:

$$(i)\ \frac{2\theta}{5+\theta^2} \qquad (ii)\ \frac{6}{\theta^2+\pi^2} \qquad (iii)\ \frac{1}{1+2\cos(\theta)}$$

$$(iv)\ 4-\cos(\theta) \qquad (v)\ 3\cos(\theta)-i\ ?$$

b) For those processes in Exercise 4.10 a) which qualify as a power spectral density, evaluate the mean square value of the process.

Exercise 4.11 If $x(k)$ is a zero-mean, ergodic process with $R_{xx}(k) = ca^{|k|}$, where $a,\ c \neq 0$ are real constants, find $S_{xx}(z)$.

Exercise 4.12 Given the power spectral density $S_{xx}(z) = (0.5z^2+z+0.5)/z$, with $z = e^{i\theta}$, find its autocorrelation sequence. Hence, determine the average power of the process.
Given that the mean of the process is $\frac{1}{2}$, find the variance of the process.

Exercise 4.13 Consider the system function:

$$H(z) = \frac{z}{z-\frac{1}{2}} + \frac{z}{z+\frac{1}{4}}, \qquad |z| > \tfrac{1}{2}.$$

a) Find $H(z)H(z^{-1})$ and indicate the region of convergence.

b) Suppose $x(k)$, assumed to be a wide-sense stationary random process, is the input to a linear, time-invariant system, with system function given above in Exercise 4.13. a).
If $R_{xx}(k) = 2\delta(k)$, find the mean square value of the output.

Exercise 4.14 An input sequence, with power spectral density $(0.4z^2+0.7z+0.4)/z$, is applied to a digital filter. The cross spectral density between the input and output is found to be

$$\frac{z^2+1.75z+1}{(z-0.2)(z-0.6)}.$$

What are the system function and impulse response for the filter?

Exercise 4.15 The z-transform of a power spectral density of a wide-sense stationary noise process, $\{n(k)\}$, with zero mean, is

$$S_{nn}(z) = 12z/((z+2)(2z+1)),$$

with region of convergence $\left\{z \in \mathbb{C} : \frac{1}{2} < |z| < 2\right\}$. Suppose the noise process is passed through a linear, causal, time-invariant, pole-zero filter with system function $H(z)$. It is found that the z-transform of the cross-spectral density between the input noise, $\{n(k)\}$, and the output noise, $\{v(k)\}$, for the digital filter is

$$S_{nv}(z) = \frac{3z}{(z-\frac{3}{4})(z+2)}.$$

a) Determine $H(z)$.

b) Deduce that the z-transform of the power spectral density for the output noise is

$$S_{vv}(z) = 12z/((4z-3)(4-3z)),$$

with region of convergence $\left\{z \in \mathbb{C} : \frac{3}{4} < |z| < \frac{4}{3}\right\}$.

c) Find the mean square output noise fluctuations, $\overline{v^2(k)} = R_{vv}(0)$, where $R_{vv}(k)$ denotes the output noise autocorrelation function.

Exercise 4.16 Find the power spectral density, $S_{yy}(\theta)$, for the output from the causal, MA filter:

$$y(k) = \frac{1}{2}[x(k) + x(k-1)],$$

given that the input is white noise with variance σ_x^2. Also, determine the z-transform of the cross spectral density of the input with the output, namely $S_{xy}(z)$.

Exercise 4.17 Assume that $x(k)$, a zero-mean, wide-sense stationary, white process, has a variance of 2. Suppose that $x(k)$ is the input to:

a) an ideal lowpass filter with

$$H(e^{i\theta}) = \begin{cases} 2, & 0 < |\theta| < \tfrac{1}{2}\pi, \\ 0, & \tfrac{1}{2}\pi < |\theta| < \pi, \end{cases}$$

and $H(e^{i\theta}) = H(e^{i(\theta+2\pi)})$;

b) an ideal bandpass filter with

$$H(e^{i\theta}) = \begin{cases} 2, & \tfrac{1}{4}\pi < |\theta| < \tfrac{1}{2}\pi, \\ 0, & \text{otherwise}, \end{cases}$$

and $H(e^{i\theta}) = H(e^{i(\theta+2\pi)})$.

Evaluate and sketch (i) $R_{xx}(k)$ (ii) $S_{xx}(\theta)$ (iii) $S_{yy}(\theta)$, where y is the output from the filter.
Determine $\overline{y^2(k)}$ (hint: use (4.21)).

Exercise 4.18 Consider a first order, causal filter with difference equation

$$y(k) = -\frac{1}{2}y(k-1) + x(k),$$

where $\{x(k)\}$ is the input sequence and $\{y(k)\}$ is the corresponding output sequence. Given that the output of a 8-bit quantizer is input to the filter and assuming the quantization error has zero mean, estimate the quantization noise power in the output to the filter.

Part II

Image Processing

Chapter 5

Digital image processing fundamentals

5.1 Introduction

A digital image can be obtained from many different sources and imaging modalities exploiting different principles, such as incident light, x-ray, ultrasound or infrared radiation to name a few. Readers wishing to have more information on the different imaging modalities can refer to the introductory Chapter of [Bernd, 2004]. Those readers wishing to have more information on the human vision system and machine vision technology can refer to the introductory chapters in [Nixon and Alberto, 2002] and [Forsyth and Ponce, 2003]. It is important to note that, irrespective of the imaging modality exploited, an image is only a possible representation of reality, it is not the actual object or phenomenon being imaged. Image processing is, as such, an extension of signal processing where the signal obtained by a camera can be noisy, affected by the sampling or spatial resolution of the acquisition device as well as the quantization for its digital representation in terms of grey levels or colours. Such a realisation is particularly important in medical images where the images are not the organs. Recent medical imaging techniques aim to register different types of images together in order to exploit their relative benefits. Computed Tomography (CT) provides the best geometrical accuracy, however, magnetic resonance imaging (MRI) provides better soft tissue imaging, making it the modality of choice for brain imaging. Functional imaging, such as functional

MRI, has been used to provide information on the brain activity. Position Emission Tomography (PET) imaging can provide information on the tumour activity, but is not geometrically accurate; hence, it has to be merged with a CT image to be exploitable. The technologies adopted to locate tumour and patient motion, just before or during the treatment, include mega-voltage X-ray, kilo-voltage X-ray, optical, electromagnetic, ultrasound, as well as air flow and volume, using a spirometer. These on-line imaging and monitoring techniques are required to be correlated to the images used for planning. Correlating information between different modality, using image registration, requires assumptions to be made on the anatomical changes that may have occurred, resulting in organs, and to a lesser extent tumour volume, being geometrically different. Non-rigid registration is currently the method of choice for both; performing such a mapping automatically delineates organs based on a deformable ATLAS [Commowick et al., 2008, Sims et al., 2009]. The main issue is to verify that the resulting distorted image, created to 'fit' the new patient geometry, corresponds to the actual shape of the tumour and surrounding tissues, which can only be truly assessed during surgery. The current challenge of medical imaging is not only to provide a 'true' representation of reality, but to do so at the right time. Three dimensional 'movies' of organs can now be reconstructed off-line, using techniques such as respiratory correlated computed tomography, commonly referred to as four dimensional (4D) CT (see [Pan et al., 2004]), as well as four dimensional cone beam imaging (discussed in [Sonke et al., 2005]).

5.2 Image representation

An image is traditionally expressed as a continuous function of two, three or even four variables. A still image taken, for example, by a camera will be represented by a function of two variables: `MyStillImage = f(x,y)` where `x` and `y` can be coordinates within a standard coordinate system, with `x` representing the row and `y` the column if `f(x,y)` is expressed in a matrix format (for example, using MATLAB®). The element of the image at the location (x, y) is referred to as a pixel and is associated a particular colour.

Remarks 5.1

a) *In MATLAB® the origin of the coordinate system is the top left of the image, such that the coordinate x represents the row element starting from the top and y represents the column starting from the left.*

b) *Due to MATLAB® matrix indexing rules, the top left pixel is represented by the element* `ImageName(1,1)` *as opposed to* `ImageName(0,0)` *(see [Gonzalez et al., 2004]).*

A movie or video can be represented by a function of three (black and white video) variables, where x and y indicate the pixel location associated with a grey level or colour and the variable t is the time at which each image is acquired. MATLAB® organises each video using structures. Colour video have a RGB image associated with each movie frame, where each pixel, located at (x, y), is associated with three colours. 3D images are a series of concatenated 2D images that form a volume. Each element (x, y, z) of the volume is referred to as a *voxel*.

Respiratory correlated computed tomography produces 4D images, which are 3D spacial images as a function of time.

This book will focus on 2D images, as the aforementioned image representations can all be expressed as a set of 2D images. Note, however, that some of the 2D imaging techniques have been readily extended to volumetric images. To represent a 2D image into a computer, it is necessary to discretise both `x` and `y` in terms of depth of colour and pixel size. A convenient representation for 'geometrical' discretisation is that of data in matrix form. The image `f` is sampled and quantized such that it results in a digital image $\mathbf{A}$ containing M rows, N columns associated with one grey level value in the case of grey level images, or three colour values R, G and B in the case of RGB images.

$$\text{Sampling } f \text{ produces} \qquad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{M1} & a_{M2} & \dots & a_{MN} \end{bmatrix}.$$

Each element of $\mathbf{A}$ is referred to as a pixel, image element, or picture element (see [Gonzalez et al., 2004]).

In grey-level images, each element of $\mathbf{A}$, namely a_{ij}, is associated with a grey-level value, that is an integer from a prescribed set of integers that represents different shades of grey. The most important decision to take is to determine the most appropriate number of grey levels. In practice, the number of distinct grey levels L is an integer, which is a power of 2, that is $L = 2^k$, where $k \in \mathbb{N}$. The discrete levels are usually equally spaced in the interval $[0, L-1]$. Such an image representation is traditionally referred to as a k-bit image, or having a dynamic range $[0, L-1]$.

Exercise 5.1 Determine the range of grey level available in an 8-bit image.

The number of bits required to store an image is $M \times N \times k$.

Exercise 5.2 How many bits are required to store a 512 by 512 images containing 256 grey levels?

5.3 Type of images with MATLAB®

The image processing toolbox supports binary, indexed, intensity, and Red-Green-Blue (RGB) image types. Note that [Forsyth and Ponce, 2003] offers a good description of the theoretical and technical background associated with different types of light source and colour representation of an image.

The description of the MATLAB® functions and data format are based on (The MathWorks, Inc., 2010).

Binary image: An image, containing only black and white pixels, represented as a logical array of 0's and 1's, respectively.

Indexed image: An image whose pixel values are represented by an array, containing an integer values between 1 and n, where n is the number of distinct set of RGB values within the colour-map (usually called a *colormap*). It is, in effect, a look-up table associating an integer, representing a colour, with the element within the colormap, that is the colour to be used when displaying the image. The colormap is always an $n \times 3$ array of class double. Note that, if the colours displayed are found to be unsuitable, it is possible to modify the colormap to obtain different colours without modifying the information in the original image.

Grey scale or grey level images: An image (previously referred to in the literature as an *intensity image*) consisting of intensity values between 0 (black) and 1 (white) representing grey levels within the range $[0, 2^n]$ (see Section 5.2).

RGB or truecolour image: An image in which each pixel is specified by three values (as opposed to an index referring to an element in a colour map , see indexed image) – one each for the red, green, and blue components of the pixel's color. In MATLAB®, an RGB image is represented by an $m \times n \times 3$ array. A pixel with $(0, 0, 0)$ is black, (k, k, k) is white, $(k, 0, 0)$ is red and so on. (k is the quantization level.) For more information of colour images see [Sonka et al., 1999], p23.

Remark 5.2 *By default, MATLAB® stores most data in arrays of class double (64-bit) floating-point numbers. For image processing, however, this data representation is not appropriate due to the large amount of mathematical operations to be performed on images and the relatively small values of each element representing information within an image. To reduce memory requirements and speed up image processing algorithms, it is standard to use 8-bit or 16-bit unsigned integers, or boolean for black and white images.*

5.4 Converting data classes and image types

5.4.1 Data types and conversions

By default MATLAB® data type will be double. Using MATLAB®, a double data type can be converted to:

8-bit unsigned integer with the command -

```
UnsignedInteger8bits = uint8(DoubleMatrix );
```

16-bit unsigned Integer -

```
UnsignedInteger16bits = uint16(DoubleMatrix );
```

These commands output an integer by taking only the integer part of the double data type and applying restrictions such that the resulting number is in the range $\left[0, 2^{8}\right]$ or $\left[0, 2^{16}\right]$.

To convert to boolean type, use the MATLAB® command:

```
LogicalNumber = logical(DoubleMatrix );
```

If the argument `DoubleMatrix` contains elements different from 0 or 1, they are converted to 1 by default. It is good practice to ensure that logical numbers are either 0 or 1 when created as a matrix. For example, the MATLAB® command:

```
DoubleInRangeZeroToOne =mat2gray(DoubleMatrixA, [Amin, Amax] );
```

converts a double data type to a double data type in the range $[0, 1]$, where 0 denotes black, 1 denotes white, and values in `DoubleMatrixA` less than `Amin` are set to 0 and values in `DoubleMatrixA` greater than `Amax` are set to 1.

5.4.2 Image types and conversions

An image is read using the command `imread` and can be converted to

(i) 8-Bit Unsigned Integer type with MATLAB® command:

```
UnsignedInteger8bitsImage = im2uint8(ImageType);
```

(ii) 16 Bit Unsigned Integer type with command:

```
UnsignedInteger16bitsImage = im2uint16(ImageType);
```

(iii) logical (binary image) type with MATLAB® command:

```
BlackAndWhiteImage = im2bw (ImageType);
```

(iv) double data type using:

```
DoubleImage = im2double (ImageType);
```

Some other conversions are as follows:

`demosaic`: This converts a Bayer pattern encoded image to a truecolor (i.e. RGB) image. A Bayer filter mosaic, or colour filter array, refers to the arrangement of colour filters that let each sensor, in a single-sensor digital camera, record only red, green, or blue data. The Bayer pattern emphasizes the number of green sensors to mimic the human eye's greater sensitivity to green light. A MATLAB® command has the form `RGB = demosaic(Image, sensorAlignment)`, where `sensorAlignment` is one of the following text strings `'gbrg'`, `'grbg'`, `'bggr'`, `'rggb'` that specifies the Bayer pattern. Each string represents the order of the red, green, and blue sensors by describing the four pixels in the upper-left corner of the image (left-to-right, top-to-bottom).

`dither`: This converts a grey-scale image to a binary image or converts a RGB image to an indexed image. The MATLAB® command

```
X = dither(ImageRGB, map, Qm, Qe)
```

creates an indexed image from RGB, where Qm specifies the number of quantization bits to use along each colour axis for the inverse colormap, and Qe specifies the number of quantization bits to use for the colour space error calculations. If Qe $<$ Qm, dithering cannot be performed, and an undithered indexed image is returned in X. If one omits these parameters, dither uses the default values Qm $= 5$ and Qe = 8. If all arguments except the image are omitted, `dither` converts to a binary image (black and white).

`gray2ind`: This converts a grey-scale image to an indexed image. The command

```
[X,map] = gray2ind(I,ColormapSize)
```

converts the image I to an indexed image X. The argument ColormapSize specifies the size of the colormap.

For example, gray(ColormapSize) generates a gray colormap with ColormapSize grey levels where ColormapSize must be an integer between 1 and 65536 for grey scale images, with default ColormapSize $= 64$. If the image is a binary image then the default for ColormapSize is 2.

`grayslice`: This converts a grey-scale image to an indexed image by using multi-level thresholding. The command X = grayslice(I,n) thresholds the intensity image I returning an indexed image in X. Threshold values can be specified using X = grayslice(I,v), which thresholds the intensity image I using the values of v, where v is a vector of values between 0 and 1, returning an indexed image in X.

`im2bw`: This converts a grey-scale image, indexed image, or truecolor image, to a binary image, based on a luminance threshold.

`ind2gray`: This converts an indexed image to a grey-scale image.

`ind2rgb`: This converts an indexed image to a truecolor image.

`mat2gray`: This converts a data matrix to a grey-scale image, by scaling the data.

`rgb2gray`: This converts a RGB image to a grey-scale image.

`hsv2rgb`: This converts a hue-saturation-value (HSV) colormap to a red-green-blue (RGB) colormap

Remark 5.3 *To work with images that use other color spaces, such as HSV, first convert the image to RGB, process the image, and then convert it back to the original color space. For more information on the means by which MATLAB® handles different colour images please refer to the Section: Help >Image Processing Toolbox >User's Guide >Color.*

`rgb2ind`**:** This converts a truecolor image to an indexed image.

5.5 Reading and writing images

Image files contain generally two parts: a header, where the format is described together with relevant file information, and the data. To read the image header, the following function is used: `info = imfinfo(filename,fmt)`, which returns a structure, `info`, whose fields contain information about an image in a graphics file.

The most widely used command to load an image in the workspace is `imread`, where specific image formats can be read with specialised reading functions. At the time of writing this book, other supported formats include three medical image formats: DICOM (most widely used), Mayo Analyze 7.5 and Interfile Files, as well as a 'photographer's' format: High Dynamic Range Images. This book will focus on reading and writing typical image formats using `imread/imwrite` as well as medical data using `dicomread/dicomwrite`.

The format for `imread` is as follows:

```
[A, map] = imread(’imagefilename’,’extension’);
or
[A, map] = imread(’imagefilename.extension’);
```

Similarly, to create an image, the `imwrite` command can be used. The structure

```
imwrite(A,map,filename,fmt.,Param1,Val1,Param2,Val2...)
```

writes the image **A** to the file specified by `filename`, in the format specified by `fmt`. The image **A** can be an $M \times N$ (grey-scale image) or an $M \times N \times 3$ (truecolor image) array. The parameters are optional and relate to a specific image format. For example, with `jpeg` images, two useful parameters are the `’Bitdepth’`, which gives the number of bits used for a grey level or a colour image, and `’Quality’`, where a value of 100 denotes the best quality and the largest file size.

Using `imread/imwrite` it is possible to read/ write the following image file formats:

‘bmp’	Windows Bitmap
‘cur’	Windows Cursor resources
‘gif’	Graphics Interchange Format
‘hdf’	Hierarchical Data Format (HDF)
‘ico’	Windows Icon resources (ICO)
‘jpg’ or ‘jpeg’	Joint Photographic Experts Group (JPEG)
‘JP2’	JPEG 2000 Joint Photographic Experts Group 2000
‘JPF’	JPEG 2000 Joint Photographic Experts Group 2000
‘JPX’	JPEG 2000 Joint Photographic Experts Group 2000
‘J2C’	JPEG 2000 Joint Photographic Experts Group 2000
‘J2K’	JPEG 2000 Joint Photographic Experts Group 2000
‘pbm’	Portable Bitmap (PBM)
‘pcx’	Windows Paintbrush (PCX)
‘pgm’	Portable Greymap (PGM)
‘png’	Portable Network Graphics (PNG)
‘pnm’	Portable Anymap (PNM)
‘ppm’	Portable Pixmap (PPM)
‘ras’	Sun Raster (RAS)
‘tif’ or ‘tiff’	Tagged Image File Format (TIFF)
‘xwd’	X Windows Dump (XWD)

Today, one of the most common file formats for digital camera has extension ‘jpg’. The following example illustrates the typical steps of obtaining information about the image, reading the image into MATLAB® and displaying it.

Example 5.1 To read an image stored in `greens.jpg` and display the result, the following MATLAB® commands may be used:

```
% Example:  reading jpg images
close all,            % close all figures
clear all,            % clear all data in the workspace
% Define the path where all the test images are located
impath = 'C:\MATLABimagproc\SampleImages\';
% display information about the image
% (not for use with DICOM images)
info = imfinfo([impath,'greens.jpg']);
% read a jpeg image using two equivalent expressions
Igreens= imread([impath,'greens.jpg']);
IgreensB = imread([impath,'greens'],'jpg');
% Creates one figure at a time and displays one image within
% each figure.  The images displayed will be identical.
figure(1);imshow(Igreens);
figure(2);imshow(IgreensB);
```

Specific image formats require other commands to be used. For example, to read a high dynamic range image into the MATLAB$^{®}$ workspace, use the `hdrread` function as in

```
hdr_image = hdrread('office.hdr');
```

High Dynamic Range (HDR) images attempt to capture the whole tonal range of real-world scenes (called *scene-referred*), using 32-bit floating-point values to store each colour channel. The output image `HDR_image` is an $m \times n \times 3$ image of type single. To view an HDR image, you must first convert the data to a dynamic range that can be displayed correctly on a computer. Use the `tonemap` function to perform this conversion. The command `tonemap` converts the high dynamic range image into an RGB image of class uint8. Suitable commands are:

```
rgb = tonemap(hdr_image);
imshow(rgb);
```

To create a high dynamic range image from a group of low dynamic range images, use the `makehdr` function as in `hdr_image = makehdr(files)`. To write a high dynamic range image from the MATLAB$^{®}$ workspace into a file, use the `hdrwrite` function in the form: `hdrwrite(hdr_image,'filename')`.

5.5.1 Reading and writing DICOM images

The second example deals with Digital Imaging and Communications in Medicine (DICOM) images. DICOM is a standard that was created to overcome difficulties associated with each manufacturer having their own proprietary format. It is a standard which was instigated in 1938 by the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA). They released in 1993 the version 3.0 of the standard, which was then renamed DICOM. The DICOM standard has become widely accepted, but it is still in evolution to accommodate issues raised by users. Similarly, the behaviour of the MATLAB® command `dicomread` has evolved, since it was first introduced. To write DICOM images the command: `dicomwrite(..., 'ObjectType', IOD,...)`, where `IOD` refers to the DICOM information object that is required for a particular type of image. The `IOD` supports: `'Secondary Capture Image Storage'` (which is the default), `'CT Image Storage'`, `'MR Image Storage'`, but not DICOMrt for radiotherapy, which is a specific DICOM format than contains additional information (or metadata) relating to radiotherapy treatment planning. A MATLAB® user can, however, create their own version of DICOMrt by adding the required metadata such as the 'outlined target volume' or'treatment plan information' using the `dicomwrite` command with the `CreateMode` option set to `'Copy'`. In this mode, `dicomwrite` writes the image data to a file, also including the metadata that is specified by the user as a parameter. Note that MATLAB® does check whether the modified DICOM format contains the set of required metadata to be appropriately supported by third party software.

Example 5.2 To read an image stored in `US-PAL-8-10x-echo.dcm` and display the result, the following MATLAB® commands may be used:

```
% Define the path for the location of the test images.
impath = ’C:\MatlabImagProc\SampleImages\’;
%
% Read the dicom image, dicom info.
infoUS = dicominfo(’US-PAL-8-10x-echo.dcm’);
[ImOriginal, map] = dicomread(’US-PAL-8-10x-echo.dcm’);
infoCT = dicominfo(’CT-MONO2-16-ankle.dcm’);
ImCT = dicomread(infoCT);
%
% Write DICOM images.
% Write CT image X to a DICOM file along with its metadata.
% Use the dicominfo function to retrieve metadata ’infoCT’
% from a DICOM file.
dicomwrite(ImCT, [impath,’CTfile.dcm’], infoCT)
%
% Display the dicom images.
imtool(ImCT)           % Display image using interative tool
[ImCTRead, map] = dicomread([impath,’CTfile.dcm’]);
figure(1);
subplot(121),imshow(ImCT,’InitialMagnification’,100)
subplot(122),imshow(ImCTRead,’InitialMagnification’,100)
%
% Obtain and modify the file information by reading the file
% information.
dicomanon([impath,’CT-MONO2-16-ankle.dcm’],...
...[impath,’UScopy01anonym.dcm’]);
infoAnonym = dicominfo([impath,’CT-MONO2-16-ankle.dcm’]);
%
% Write the character xyz as the family name.
infoCT.PatientName.FamilyName=’xyz’;
%
% Anonymise the data (by removing the name).
infoAnonym.PatientName
implay(ImOriginal)         % play US dicom movie
```

Analyze 7.5 is a file format, developed by the Mayo Clinic, for storing MRI data. An Analyze 7.5 data set consists of two files:

Header file (filename.hdr) Provides information about dimensions, identification, and processing history. You use the `analyze75info` function to read the header information.

Image file (filename.img) Image data, whose data type and ordering are described by the header file. You can use `analyze75read` to read such image data into the MATLAB® workspace:

```
info = analyze75info('brainMRI.hdr');
X = analyze75read(info);
```

Interfile is a file format that was developed for the exchange of nuclear medicine image data. An Interfile data set consists of two files:

Header file (filename.hdr) Provides information about dimensions, identification and processing history. You use the `interfileinfo` function to read the header information.

Image file (filename.img) Image data, whose data type and ordering are described by the header file. You use `interfileread` to read the image data into the MATLAB® workspace.

The MATLAB® commands:

```
nfo = interfileinfo('dyna');
X = interfileread('dyna');
```

Examples of such images can be downloaded from the Interfile Archive maintained by the Department of Medical Physics and Bioengineering, University College, London, UK:
http://www.medphys.ucl.ac.uk/interfile/.

5.6 Manipulating images obtained from movies in MATLAB®

It is now possible to read and play a wide range of different formats for movies, provided that the operating system you use supports the appropriate format and codec. Movie files can be quite large and, hence, it is a good idea to check the size of the movie before opening it or playing it using the command `mmfileinfo`. For example, the MATLAB® commands:

```
info = mmfileinfo('xylophone.mpg');
info
```

produce the following results:

```
Filename:  'xylophone.mpg'
Path:  'C:\MATLAB_R2010a\toolbox\matlab\audiovideo'
Duration:  4.7020
Audio:  [1x1 struct]
Video:  [1x1 struct]
```

and

```
info.Video   % Access the video format information
```

gives

```
ans =
Format:  'MPEG1'
Height:  240
Width:  320
```

There are two MATLAB® commands to play a movie: `implay` and `movie`. The command `implay` opens and plays a compatible movie file. It is the simplest command to use for this purpose. For example, one could write

```
implay('xylophone.mpg'); % Read and play the mpg movie
```

The MATLAB® command `movie` can only play a movie which is defined by a matrix whose columns are 'movie frames'. To generate a MATLAB® movie, it is, therefore, necessary to create the movie frame data structure. To extract one image from a movie, i.e. create a frame, use the command `getframe`, which returns a snapshot (namely a *pixmap*) of the current axes or figure, namely a movie frame. A movie frame can also be created from a 4D array using the following sample code, where `cdata` is one of the required properties of the image object which contains the data, that is the image for each frame. For example, movie frame data structure creation can be achieved using the MATLAB® commands:

```
for k = 1:numberOfFrames
mov(k).cdata = a4DarrayCreatedByReadCommand(:,:,:,k);
mov(k).colormap = [];
end
```

To be able to perform some image processing, it is, however, more important to be able to access information on each frame of the movie than to play it. The MATLAB® command `aviread` enables one to create, directly, a MATLAB® movie structure. The kth image or frame in the movie can then be extracted using the command `[ImageData,Map] = frame2im(mov(k))`. MathWorks is currently recommending to use `mmreader` instead of the function `aviread`, which only reads AVI files. The function `mmreader` creates a multimedia reader object for reading video files. It is a new MATLAB® function created to handle a wide variety of movie file formats or containers, which describes the layout of the file (for example, `.avi`, `.mpg`, `.mov`), and various codec support; here, codec describes how to code/decode the data. Note that if the codec is not available, MATLAB® may not be able to read the file, even if it is one of the listed formats. This is a common occurrence for `avi` files created on a 32 bit system with Microsoft XP operating system using `Indeo5` codec. Indeed, `Indeo` does not provide support for the Windows 7 operating system. To obtain file formats supported by the `mmreader`, one can use the MATLAB® command: `mmreader.getFileFormats`. Some possible output might be:

```
Video File Formats:
   .asf - ASF File
   .asx - ASX File
   .avi - AVI File
   .mj2 - Motion JPEG2000
   .mpg - MPEG-1
   .wmv - Windows Media Video
```

Note that these formats are available in a Windows operating system. Assuming that the movie file is compatible, there are three steps to access each image frame:

(i) Create a multimedia reader object using the function `mmreader`.

(ii) Read the video frames from the multimedia reader object to create a 4D matrix, with dimension $H \times W \times B \times F$, where H is the image frame height, W is the image frame width, B is the number of bands in the image (for example, 3 for RGB), and F is the number of frames that are read.

(iii) Extract an image or a set of images from the movie by:

   a) using an array matrix, indexing from the 4D matrix created in (ii);

b) creating a movie frame data structure and subsequently extracting a frame using `[ImageData,Map] = frame2im(mov(k))`.

Remark 5.4 *Note that to create the appropriate data structure, in order to load each movie frame into the structure, it is a good idea to pre-allocate the memory.*

The following example, namely Example 5.3, illustrates the use of the principal MATLAB® commands to read a movie and then extract the individual frame information.

Example 5.3 Consider the problem of reading a movie file and extracting image frames. The following MATLAB® commands may be used to create a MATLAB® movie structure from the video frames:

```
% Obtain general information on the movie.
%
info = mmfileinfo(’xylophone.mpg’);
%
% open and play a compatible movie file
%
implay(’xylophone.mpg’);
%
% Create the MATLAB object containing the movie information.
%
xyloObj = mmreader(’xylophone.mpg’);
%
% Pre-allocate the movie structure.
% Determine the number of frames and image size.
%
nFrames = xyloObj.NumberOfFrames;
vidHeight = xyloObj.Height;
vidWidth = xyloObj.Width;
%
% Create a structure of cdata or movie frames:
% cdata is a property of an image object, it contains the data
% array.
%
mov(1:nFrames) = struct(’cdata’, zeros(vidHeight, vidWidth,...
...3,’uint8’),’colormap’, []);
%
% Read in all video frames.
%
vidFrames = read(xyloObj);
%
% Create a MATLAB movie structure from the video frames.
%
for k = 1 : nFrames
mov(k).cdata = vidFrames(:,:,:,k);
mov(k).colormap = [];
end
```

To create an `avi` file from the movie frames and play the `avi` movie, the following commands can be used:

```
% Create a figure to display the video.
hf = figure;
%
% Resize the figure based on the video’s width and height
set(hf, ’position’, [150 150 vidWidth vidHeight])
%
% Play back the movie once at the video’s frame rate.
% mov is the movie defined by a matrix whose columns are movie
% frames.
movie(hf, mov, 1, xyloObj.FrameRate);
%
% Create an avi file from the movie frames and an avi file
% object, which can be used to build an avi movie.
% Once all the frames have been added to the movie object,
% the object must be closed to enable the file to be written in
% the appropriate format, otherwise the avi file created will
% be unreadable.
% The frame rate specified is 10 per second, which is less than
% the initial frame rate.  This will play the original movie in
% slow motion.
aviobj = avifile(’xylophone.avi’,’compression’,’None’,’fps’,10);
%
% Create a figure.
hf = figure;
%
% Resize the figure based on the video’s width and height
set(hf, ’position’, [150 150 vidWidth vidHeight])
%
% Append the frame(s) contained in the MATLAB movie MOV to the
% avi file.
aviobj = addframe(aviobj,mov);
%
% Finishes writing and closes the movie (IMPORTANT LAST STEP).
aviobj = close(aviobj);
%
% Play the avi movie.
implay(’xylophone.avi’)
```

Exercise 5.3 Write down the command used to load into MATLAB® bmp, tif and jpg image files present in the directory:
W:\EC\STUDENT\m19mse\SampleImages.

Exercise 5.4 Write down the MATLAB® command to load a jpeg file and display it as a grey-level image file.

5.7 Displaying images with MATLAB®

5.7.1 Image Tool Navigation Aids: `imtool`

The MATLAB® command `imtool` can be used to display image data **I** using the commands:

```
imtool(I)
% specify the range of grey levels in [LOW  HIGH]
imtool(I,[LOW  HIGH])
```

or load a file containing an image within the format listed in Section 5.3, namely `imtool(filename)`.

A useful command to close all the image viewers is: `imtool close all`. A screen-view illustrating the use of `imtool` is shown in Figure 5.1.

5.7.2 Displaying an image using '`imshow`'

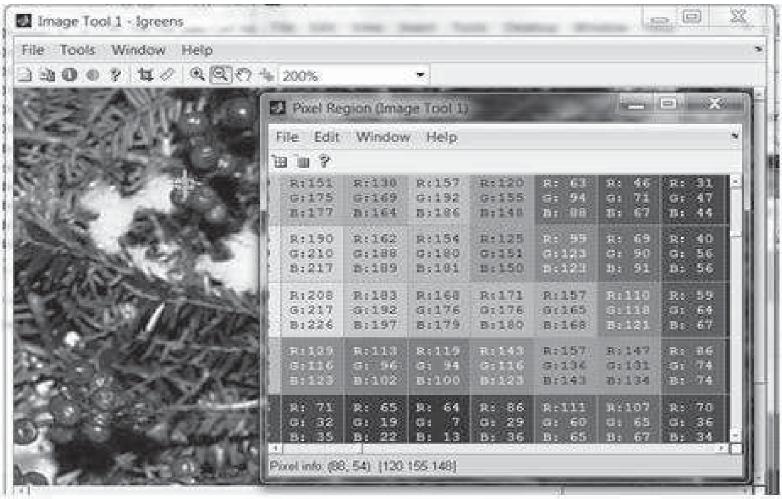
`imshow(A,L)` displays the intensity image **A** with `L` discrete levels of grey. If you omit `L`, `imshow` uses 256 grey levels on 24-bit displays, or 64 grey levels on other systems.

OLD version: `imshow( A,L,'truesize')` This command calls the true-size function, which maps each pixel in the image to one screen pixel. The command `imshow( A,L,'notruesize')` does not call the truesize function.

NEW version: `imshow(A,'InitialMagnification',100)` shows the actual size of the image.

`truesize(A)` uses the image height and width for [MROWS MCOLS]. This results in the display having one screen pixel for each image pixel.

`image(A)` displays matrix **A** as an image. Each element of **A** specifies the colour of a rectangular segment in the image.



```
R:151  R:130  R:157  R:120  R: 63  R: 46  R: 31
G:175  G:169  G:192  G:155  G: 94  G: 71  G: 47
B:177  B:164  B:186  B:148  B: 88  B: 67  B: 44

R:190  R:162  R:154  R:125  R: 99  R: 69  R: 40
G:210  G:188  G:180  G:151  G:123  G: 90  G: 56
B:217  B:189  B:181  B:150  B:123  B: 91  B: 56

R:208  R:183  R:168  R:171  R:157  R:110  R: 59
G:217  G:192  G:176  G:176  G:165  G:118  G: 64
B:226  B:197  B:179  B:180  B:168  B:121  B: 67

R:129  R:113  R:119  R:143  R:157  R:147  R: 86
G:116  G: 96  G: 94  G:116  G:134  G:131  G: 74
B:123  B:102  B:100  B:123  B:143  B:134  B: 74

R: 71  R: 65  R: 64  R: 86  R:111  R:107  R: 70
G: 32  G: 19  G:  7  G: 29  G: 60  G: 65  G: 36
B: 35  B: 22  B: 13  B: 36  B: 65  B: 67  B: 34
```

Pixel info: (88, 54) [120 155 148]

Figure 5.1: A screen-view illustrating the use of the MATLAB® image viewer, called by using the command `imtool`, to inspect pixels RGB values.

Remark 5.5 *The* `image(A)` *function may not display the expected image colour. The colour associated with each pixel will be determined by the specified colormap. If you do not specify the colormap, MATLAB® will create one for you that may change depending on other functions that you are using (as the other functions may create other colormaps). See Section 5.3 on image type and, in particular, indexed image.*

`imagesc(A)` The `imagesc` function scales image data to the full range of the current colormap and displays the image.

`subimage(A,map)` This command displays multiple images in the same figure, even if the images have different colormaps. The command `subimage` works by converting images to `truecolor` for display purposes, thus avoiding colormap problems, for example,

```
subplot(1,2,1), subimage(X1,map1)
subplot(1,2,2), subimage(X2,map2)
```

5.8 Sampling

Definition 5.1 ***Sampling*** *is defined as the process of creating a finite size image, comprising M rows and N columns.*

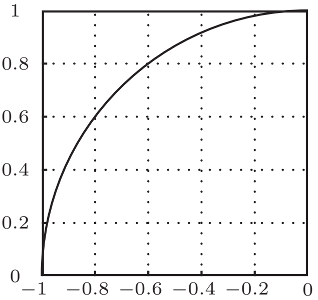
Remark 5.6 *To define the appropriate level of sampling, i.e. the number of pixels per millimetre, it is necessary to ensure that all required regions of interest (ROI) are represented by at least two pixels (see [Sonka et al., 1999]). Such an approach would help separate noise from actual feature. Note that selecting the image sampling to have 5, as opposed to 2, pixels to describe the smallest ROI is advantageous from a practical perspective.*

Example 5.4 Suppose that an image, which measures 10 cm by 10 cm, is available for analysis. To display the image on a computer, the image has to be divided into a number of rows and columns. This process is called sampling. Note that it may be of advantage to use a lower resolution than visually acceptable to quickly segment images using automated algorithms. The elements i, j characterising the location of the centres of each of the matrix elements of $\mathbf{A}$ are expressed as integer values i.e. $(i, j) \in \{1,\ 2 \ldots N\} \times \{1,\ 2, \ldots M\}$. However, these coordinates may not correspond exactly to the actual value of the physical coordinates.

Suppose the image consists of a digital photograph of a mug, showing a 9 cm tall mug. Depending on the resolution of the photograph, 9 cm may be represented by 100 pixels, for example. In this case, the third pixel from the bottom of the mug would be located at $3/100 \times 9\,\text{cm} = 0.27\,\text{cm}$.

□

Exercise 5.5 A quarter circle is to be digitised using a grid with a resolution of 0.2 by 0.2 units and represented as a black and white image. Write down the rules you are using to colour the pixels inside the quarter circle. How could you improve the representation of the quarter circle?



Exercise 5.6 A person is subject to a CT scan that takes transversal images of the body. The cross section of the person is encompassed within a rectangle of 35 cm by 25 cm. The resolution of the scanner is $0.2 \times 0.2\,\text{cm}$. How many pixels are required to image the cross section of the patient?

Exercise 5.7 Suppose that a 10cm by 10cm image is divided into 512 rows and 512 columns. What is the area of a pixel?

5.8.1 Re-sampling an image with the aid of MATLAB®

Re-sampling an image can be achieved using the commands:

```
B = imresize (A, M,METHOD)
B = imresize (A,[MROWS  MCOLS],METHOD)
```

Using `imresize`, one can specify the size of the output image in two ways.

(i) by specifying the factor to be used on the image:

```
% For a ReductionFactor satisfying 0 < ReductionFactor ≤ 1
A_1 = imresize(A,ReductionFactor)
```

(ii) by specifying the dimensions of the output image:

```
A_2 = imresize(A,[Number of Rows, Number of Columns])
```

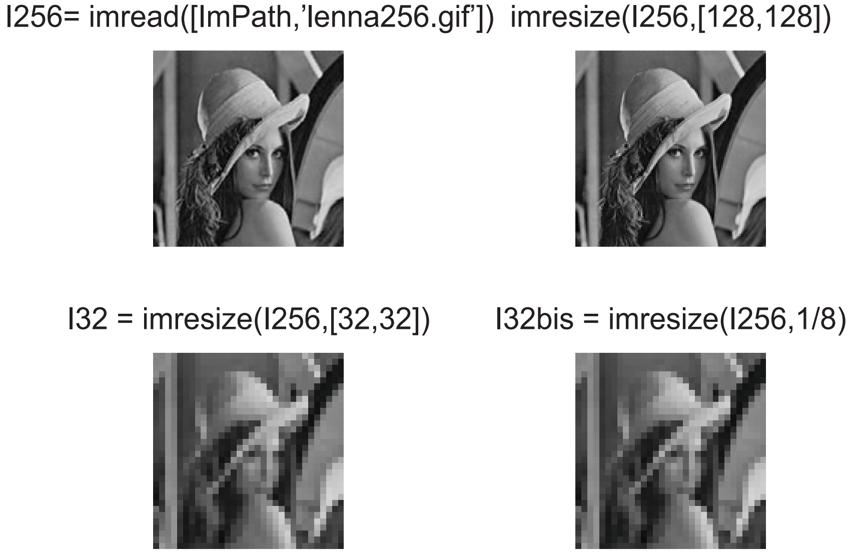
The argument ‘`METHOD`’ is used to choose the interpolation method that can be selected from:

- ’`nearest`’ (default) - nearest neighbour interpolation;
- ’`bilinear`’ - bilinear interpolation;
- ’`bicubic`’ - bicubic interpolation.

There follows some MATLAB® code to format sampling and quantization examples, and illustrate the use of `InitialMagnification`.

```
% Sampling and quantization examples
% close all figures and clear workspace
close all             % close all figures
imtool close all      % closes all the image viewers
clear all              % clear all data in the workspace
%
% Define the path where all the test images are located
ImPath = ’C:\MATLABImagProc\SampleImages\’;
%
% read the images
I256 = imread([ImPath,’lenna256.gif’]);
%
% resample one of the images to reduce its resolution from
% the initial 256 by 256 to 128 by 128, and 32 by 32
% (8 times less than the original image size.
I128 = imresize(I256,[128,128]);
I32 = imresize(I256,[32,32]);
%
% equivalent command using ratio or scale instead of number
of rows and column
I32bis = imresize(I256,1/8);
%
% Illustrating the use of sampling
% Using subplot, the magnification is set automatically.
figure(1);
subplot(221); imshow(I256)
title([’I256=imread([ImPath,’’lenna256.gif’’])’])
subplot(222); imshow(I128)
title([’imresize(I256,[128,128])’])
subplot(223); imshow(I32)
title([’I32 = imresize(I256,[32,32])’])
subplot(224); imshow(I32bis)
title([’I32bis = imresize(I256,1/8)’])
```

Programming such a code in MATLAB® would produce the images illustrated in Figure 5.2.



```
I256= imread([ImPath,'lenna256.gif'])
imresize(I256,[128,128])
I32 = imresize(I256,[32,32])
I32bis = imresize(I256,1/8)
```

Figure 5.2: Images illustrating the effect of sampling on an image.

Consider an example that illustrates the use of `InitialMagnification`.

```
% Illustrating the use of InitialMagnification
% To increase or decrease the image seen within the
% figure when there is only one image per figure
%
figure(1);
imshow(I256,'InitialMagnification',100)
title('imshow(I256,''InitialMagnification'',100)')
figure(2);
imshow(I128,'InitialMagnification',200)
title('imshow(I128,''InitialMagnification'',200)')
figure(3);
imshow(I32,'InitialMagnification',800)
title('imshow(I32,''InitialMagnification'',800)')
figure(4);
h=imshow(I128,'InitialMagnification',100)
title('imshow(I128,''InitialMagnification'',100)')
```

To illustrate the benefit of applying sampling, consideration is given to a CT image where the aim is to identify regions of interest. It is possible to apply a so called coarse-to-fine approach by initially considering an image with a low resolution to obtain the main image features. In the example, described in Figure 5.3, the image is down sampled from a 512 to a 256, then to a 32, and, finally, to a 16 pixels square image. If can be observed that, even with as little as 16 pixels, the main bony structures (in white) can still be observed, together with the shape of the bladder, which corresponds to the light grey regions between the bones. Once the main feature location has been identified on an image, which is quick to process (as it is smaller than the original), the search for features can be directed to the regions identified using finer resolution until all the required details have been found.

Exercise 5.8 Assume that a digital image, comprising 512 columns and 512 rows, represents an area of 10 cm by 10 cm. What is the area represented by a pixel?

Exercise 5.9 Read/load an image from:
W:\EC\STUDENT\m19mse\SampleImages
and determine the coarsest sampling acceptable to display that image on the screen. Repeat the process with other images. What do you observe?

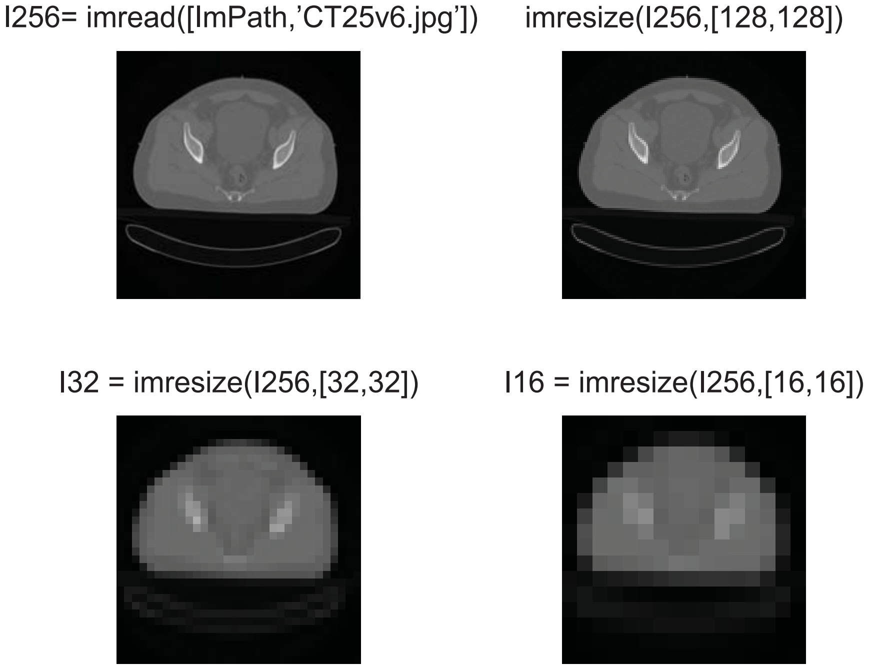
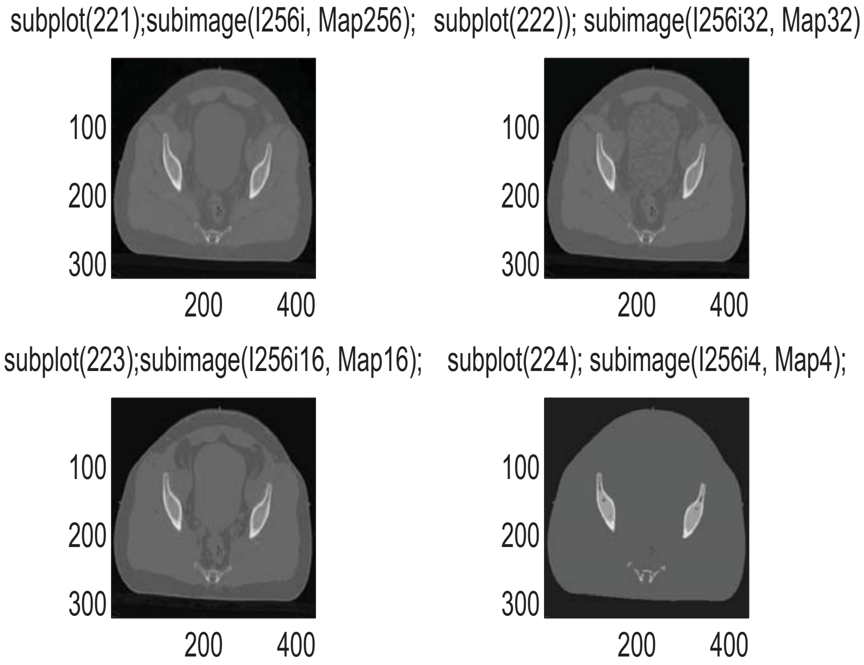

I256= imread([ImPath,'CT25v6.jpg'])
imresize(I256,[128,128])
I32 = imresize(I256,[32,32])
I16 = imresize(I256,[16,16])

Figure 5.3: Images illustrating the effect of sampling on a CT image.

5.9 Quantization

Quantization is equivalent to digitizing the amplitude value of the signal, namely the grey level value within an image.

In the example, shown in Figure 5.4, an original CT image, obtained using 256 different grey levels, is modified by reducing the number of grey levels to 64, 16 and, finally, 4.



subplot(221);subimage(I256i, Map256);

subplot(222)); subimage(I256i32, Map32)

subplot(223);subimage(I256i16, Map16);

subplot(224); subimage(I256i4, Map4);

Figure 5.4: Images illustrating the effect of quantization.

Example 5.5 Making use of Figure 5.4, quantization is used to simplify the image and extract the main features, or regions of interest, assuming that these regions of interest have a distinct range of grey levels.

```
% Close all figures and clear workspace
close all,           % close all figures
imtool close all     % closes all the image viewers
clear all,           % clear all data in the workspace
%
% Define the path where all the test images are located
ImPath = ’C:\MATLABImagProc\SampleImages\’;
%
% read the images
I256= imread([ImPath,’CT25v6.jpg’]);
I256=imcrop(I256,[30 40 440 320 ]);
%
% convert grey level to indexed image with 256 grey levels
[I256i, I256map256] = gray2ind(I256,2^8);
%
% reduce the number of grey levels from 256 to 16
% using minimum variance quantization
% Note that it is difficut for the human eye
% to differentiate between 256 bits and 16 bits images
[I256i32,I256map32] = imapprox(I256i,I256map256,32,’nodither’);
```

Consider reducing the number of grey levels from 256 to 4,and 2, using minimum variance quantization.

```
% Reduce the number of grey levels from 256 to 4
% using minimum variance quantization
[I256i16,I256map16] = imapprox(I256i,I256map256,16,’nodither’);
%
% reduce the number of grey levels from 256 to 2
% using minimum variance quantization
[I256i4,I256map4] = imapprox(I256i,I256map256,4,’nodither’);
figure(1);
subplot(221); subimage(I256i, I256map256);
title(’subplot(221);subimage(I256i,I256map256);’)
subplot(222); subimage(I256i32,I256map32);
title(’subplot(222)); subimage(I256i32,I256map32)’);
subplot(223);subimage(I256i16,I256map16);
title(’subplot(223);subimage(I256i16,I256map16);’)
subplot(224); subimage(I256i4,I256map4);
title(’subplot(224); subimage(I256i4,I256map4);’)
%
% convert back to grey level
Igray=ind2gray(I256i4,I256map4);
figure(2); imshow(Igray)
```

Note that the MATLAB® algorithms implemented in `imapprox.m` to perform quantization may not give rise to the regions that are required. In order to quantize the image into relevant levels, it is recommended to study the histogram of the image and identify a series of grey level thresholds that could be used to separate the regions of interest. This can be performed quickly by using the contrast tool within `imtool` and selecting a small grey level window and dragging it across the histogram, whilst checking the pixels highlighted in the image display window. In the following example, three grey level thresholds are selected manually to separate the regions of interest based on their typical grey level values. By contrast to the MATLAB® approach, four distinct regions are now visible representing the air, the soft tissues, the organs and the bones, see Figure 5.5.

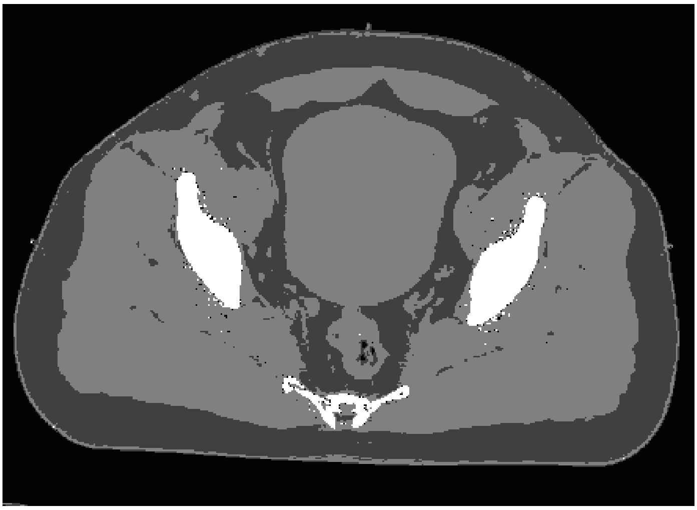


Figure 5.5: Illustrating the quantization of a CT image using manually selected quantization levels.

```
% Illustrating the effect of histogram based quantization
%
% Allocates 4 grey levels based on histogram information
% and looks for pixels within four distinct ranges.
% Changes their grey levels to four distincive ones
% and converts to double type.
Idouble = im2double(I256,’indexed’);
%
% create the output image
I256_4 = uint8(zeros(size(I256)));
%
% centre of first significant valley around grey level 75
% create first region for grey levels less than 75
I1=find(Idouble<75);
I256_4(I1)=0;
%
% centre of second significant valley around grey level 100
% create second region for grey levels between 75 and 100
I2=find(Idouble>=75 & Idouble <100 );
I256_4(I2)=64;
%
% centre of third significant valley around grey level 115
% create third region for grey levels between 100 and 115
I3=find(Idouble>=100 & Idouble <115 );
I256_4(I3)=128;
%
% create fourth region for grey levels above 115
I4=find(Idouble>115);
I256_4(I4)=255;
%
% display the resulting image
figure,imshow(I256_4)
```

The next Chapter will present, in more detail, the meaning of a histogram and how to exploit the information that is contained within it.

Exercise 5.10 Explain in your own words, and using examples, the concept of sampling and quantization.

Exercise 5.11 An image file is named `mypicture.jpg`. Write down some appropriate MATLAB® commands to load the image and plot, on the same figure,

- the original image;
- an image re-sampled such that it contains half of the original number of pixels;
- an image quantised such that it contains a quarter of the grey level of the original image;
- a 'scaled' image in the MATLAB® sense.

5.10 Conclusions

This chapter has introduced the concept of digital images and videos. It has described the tools and functions available within MATLAB® to read a wide variety of image and video files, provided that the appropriate codec are installed on the computer. MATLAB® associates a variable to each file read and such variable can be subsequently manipulated using a wide range of image processing functions to analyse, pre-process and extract information from the image. Video can be read and converted to a series of image frame that can be manipulated as easily as any other image within MATLAB®. A suggested starting point when designing an algorithm to analyse and extract relevant information from an image is to use the graphical tool `imtool`. Such tool provide the means to inspect the range of grey levels within the image as well as within specific regions of interest that may need to be analysed. The concepts of sampling and quantization have been reviewed in the context of image processing and means to reduce the number of grey level or resize the image have been illustrated. The next chapter will assume that one can read an image, manipulate the image data, and display it. The focus of Chapter 6 will be histogram processing.

Chapter 6

Histogram processing

A *histogram* of a digital image with grey levels in the range $[0, L-1]$ is a discrete function $h(r_k) = n_k$, where r_k is the kth grey level and n_k is the number of pixels in the image having grey level r_k. It is common practice to normalise the histogram by dividing each of its values by the total number of pixels in the image (see [Gonzalez et al., 2004]).

To obtain an image histogram it is, therefore, required to plot the number of pixels, `NoPixels`, in the image, `Img`, having the same grey level against that grey level values, `GreyValue`. For image data types, the histogram can be displayed, using `[NoPixels,GreyValue] = imhist(Img,NoOfBins)`

The number of bins in the histogram is specified by the image type. If `Img` is a grey-scale image, then the command `imhist` uses a default value of 256 bins. The commands `imhist` can also return the histogram counts, that is the number of times a pixel within a specific range, determined by the number of bins used, is encountered, namely `NoOfBins`, and the bin locations in `GreyValue`.

Example 6.1 Let us assume that there are 100 different grey levels, from grey level value 0 up to grey level value 99, and that the bin size is 5. The range of grey levels for each bin will therefore be: 0 to 4 for the first bin, 5 to 9 for the second bin, and 95 up to 99 for the last bin. The count, or number of pixels per bin, will be determined by adding one to a particular bin count each time a pixel is within the range is encountered. For example, if there are 6 pixels with grey levels between 0 and 4, the first bin count will be equal to 6.

Note that MATLAB® commands to produce histograms depend on the data type. An histogram can also be obtained for `double` type data using the command `[n,xout] = hist(Y,nbins)`. However, such a syntax does not work with images.

Remark 6.1 *Note the when a histogram is characterised by having a large number of pixels in a single bin (normally representing the background), it is possible to use the log scale for the y-axis or to normalise the histogram with reference to the highest peak outside the background.*

The following MATLAB® commands illustrate the calculation of image histogram with various number of bins or set of range of grey levels. Initially, the path where all the test images are located is defined and the colour is converted to the grey level of the cropped RGB image.

```
% Close all figures and clear workspace.
close all,                 % close all figures
imtool close all,          % closes all the image viewers
clear all,                 % clear all data in the workspace
% Define the path where all the test images are located.
DIRimages='C:\MatlabImagProc\SampleImages';
[RGB,map]=imread([DIRimages,'\CoventryMoon\IMG_0160.jpg']);
% Convert colour to grey level of the cropped RGB image.
% It is better to work with the smallest possible image
% and, therefore, crop the image, if necessary, immediately
% after reading the file.
GREY = rgb2gray(RGB(400:1700,800:2050,:));
```

Then, the image histogram is calculated and displayed.

```
% Grey level information criteria (based on image histogram
% processing).
%
% Calculate the image histogram to look at the distribution of
% grey levels for 256, 64, 32, 16 and 8 different bins of
% grey levels.
% Calculate the image histogram and return:
% greyLevelValue and NoOccurence,
% which is the number of times this grey level is encountered.
[COUNTS256,X256]=imhist(GREY,256);
[COUNTS64,X64]=imhist(GREY,64);
[COUNTS32,X32]=imhist(GREY,32);
[COUNTS16,X16]=imhist(GREY,16);
[COUNTS8,X8]=imhist(GREY,8);
figure(1);clf;             % create a figure and clear it
subplot(221),imshow(GREY)  % display the original image in grey
% Calculate and display the image histogram,
% which uses a default value of 256 bins.
subplot(222),imhist(GREY);
% Where the region of interest is relatively small compared to
% the whole image, it may be beneficial to display
% the histogram on a semi-log scale.
%
% Display the histograms for different grey level resolutions.
subplot(223),semilogy(X256,COUNTS256)
hold on,semilogy(X64,COUNTS64,'r')
semilogy(X32,COUNTS32,'g')
semilogy(X16,COUNTS16,'c')
semilogy(X8,COUNTS8,'k')
```

To create a new image with a reduced number of grey levels (see Section 5.9 on quantization in Chapter 5), it is possible to use the grey levels corresponding to the centre of the bins used to calculate the image histogram. The following program will set the grey level values in the vector $X8$ returned by the commands `[COUNTS8,X8]=imhist(GREY,8);` as new grey level to all the pixels that are within the range $[X8(i), X8(i+1)]$.

```
% Create a new image based on the grey levels corresponding to
% the number of bins used in the histogram.
% 1) Use 8 grey levels.
MOONdouble = double(GREY);
MOONthresh8 = GREY;
[COUNTS8,X8]=imhist(GREY,8);
for i=1:length(X8)
I=find(MOONdouble>=round(X8(i))-round(X8(1)));
MOONthresh8(I)=round(X8(i));
end
[COUNTS8b,X8b]=imhist(MOONthresh8,256);
subplot(224),imshow(MOONthresh8);
subplot(223)
hold on
semilogy(X8b,COUNTS8b,’*r’)
axis([0 256 100 max(max(COUNTS8b))+100])
% 2) Use 16 grey levels.
MOONdouble = double(GREY);
MOONthresh16 = GREY;
for i=1:length(X16)
I=find(MOONdouble>=round(X16(i))-round(X16(1)));
MOONthresh16(I)=round(X16(i));
end
[COUNTS16b,X16b]=imhist(MOONthresh16,256);
figure(2);
subplot(221),imshow(MOONthresh16);
subplot(223), imhist(MOONthresh16,256)
subplot(222),imshow(GREY) % display the original image in grey
% Calculate and display the image histogram,
% where the default value is 256 bins.
subplot(224),imhist(GREY);
```

An example showing some images and their corresponding histograms is shown in Figure 6.1.

Making use of information from a histogram, it is possible to identify the background and other objects. Usually the background is represented by a large number of pixels around a limited number of grey levels, thereby creating a peak in the histogram. Regions of interest (ROI) that are also characterised by distinct ranges of grey levels can also be recognised by peaks in the histogram.

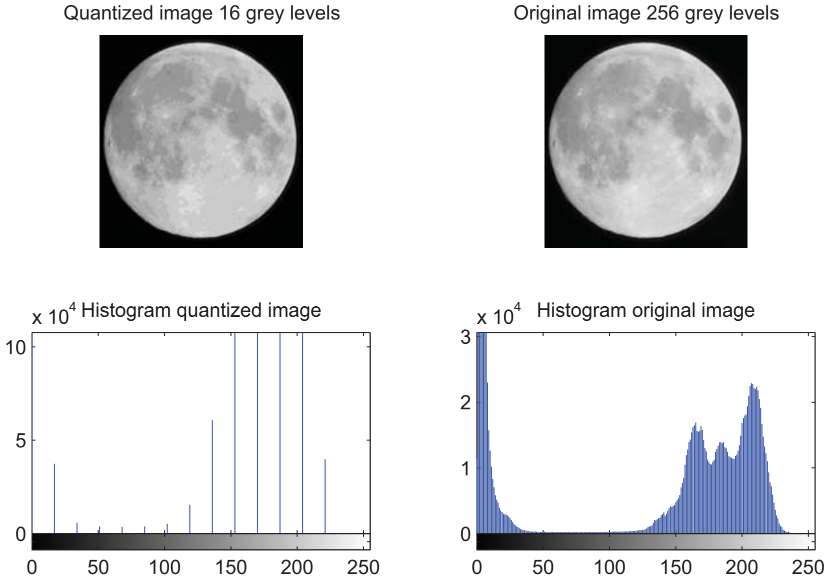


Figure 6.1: Images illustrating the use of histograms.

To separate the ROI from the background, or different ROI, one technique that can be used involves detecting the valleys between the peaks characterizing the ROI considered. Alternatively, it is possible the choose the mid-point between two successive peaks. These points, that are identified, correspond to a grey level value that could be used as threshold to separate the ROIs, or a ROI from the background.

6.1 Thresholding

Thresholding is a method widely used to segment images using grey level properties of the objects within the image. For example, thresholding could be used to extract a light object from a dark background, or vice versa.

Example 6.2 Bone segmentation in CT images
In the example shown in Figure 6.2, a threshold value of 132 was selected to segment the bony structure in a CT image. This threshold value was selected by analysing the shape of the histogram of the original image shown in Figure 6.2a) and looking at the change in gradient of the histogram (see Figure 6.2d)). The criteria considered are as follows: the bony structure is represented by a wide range of white or light grey levels. The grey level of the bony structure reflects the fact that bones are of high density, but marrow regions are of lesser density. The grey level value of the various organs belongs to a narrow range, say between 100 and 130. The number of pixels defining the regions of interest, such as the bladder, is much larger than the number of pixels defining the bony structures. This is clearly illustrated in the histogram of the image (Figure 6.2a)) and the gradient of the histogram (Figure 6.2d)).

Remark 6.2 *Using the thresholding method, small white regions, that are not bones, have been wrongly identified as bone outside the bony structures. This is a known side effect of thresholding, that does not use information with respect to the location of the pixels. To remove such small regions, it is possible to use mathematical morphology (discussed in Chapter 10).*

6.1.1 An example illustrating the segmentation of coins based on histogram information

Consider an image of four coins (shown in Figure 6.3b)), of similar grey level, which are to be segmented automatically. Figure 6.3 shows the histogram (il-

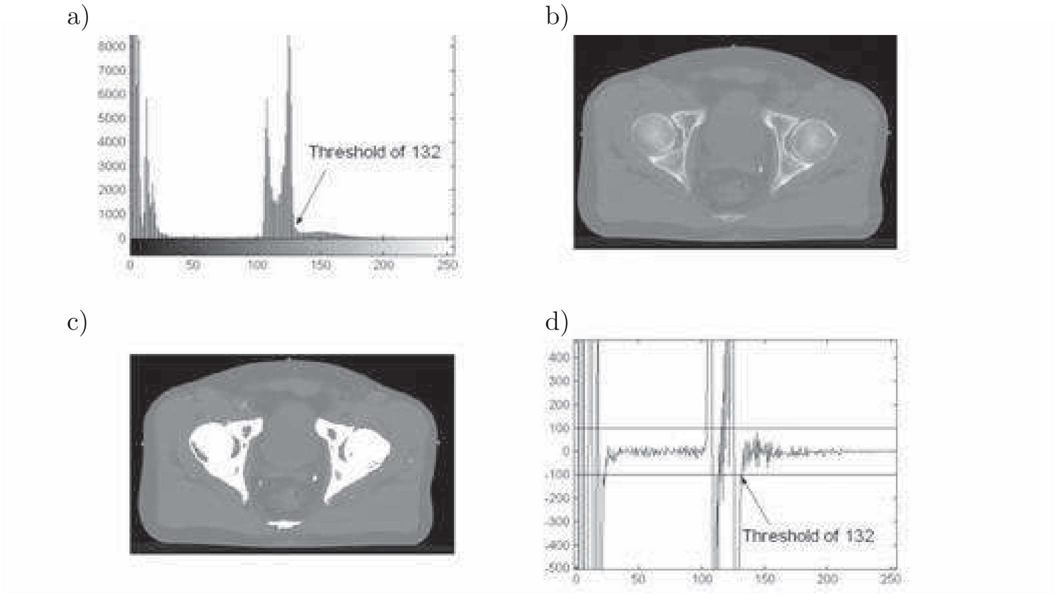


Figure 6.2: Illustrating the use of histogram and derivative of histogram to select a threshold value for bony structures in an image.

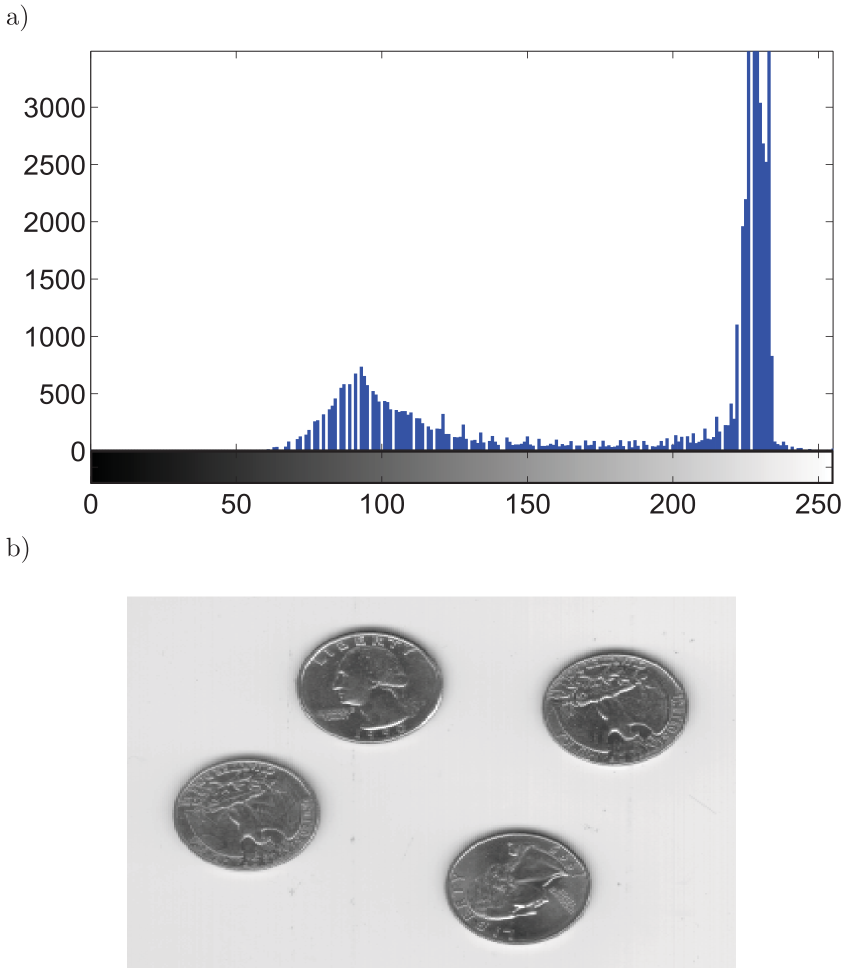


Figure 6.3: a) Illustrating the knowledge gained from the shape of the image histogram to select the threshold automatically; b) Image of four coins, of similar grey level, to be segmented automatically.

lustrated in Figure 6.3a) of an image representing four coins (shown in Figure 6.3b). The objects to be segmented are much darker than the background and are represented by the small 'hill' at a grey levels around 94. The light background covers a much larger region centred on grey level 229. There are different techniques to differentiate between two objects of different sizes and grey levels in an image. One simple approach is to calculate the minimum and the maximum grey level in the image and use a grey level value in the middle as threshold. The problem with such a method is that there may just be a few noisy pixels at a low and/or high grey level. To alleviate this problem one could consider the minimum and maximum grey level for a number of pixels greater than a given number. None of the above, however, consider the grey levels of the objects in order to determine the appropriate threshold, and such a technique assumes that the region of interest will be either lighter or darker than other regions in the image. A widely accepted technique, to determine the most appropriate threshold, is described in the following subsection.

6.1.2 Global thresholding

The `graythresh` function uses Otsu's method, which chooses the threshold to minimise the variance of the black and white pixels (for more details see [Otsu, 1979]). The MATLAB® commands to convert a grey level image to black and white using Otsu's method are:

```
Level = graythresh(A);
% Converts an image to a binary image,
% based on threshold value 'Level'
BW = im2bw(A, Level);
```

For example, the following commands show how to separate the coins within the image eight.tiff (illustrated in Figure 6.3) from the background.

```
% Separation of coins
[A,map] = imread([ImPath,'eight'],'tif');
%
% Calculate the threshold using Otsu's method
level = graythresh(A);
BW = im2bw(A,level);
BW1 = im2bw(I,210/255);
%
% Generate the images
imshow(BW);
subplot(222);
xlabel('Threshold:  Otsu''s method');
imshow(BW);
subplot(224);
xlabel('Threshold:  manual');
imshow(BW1)
```

The resulting images are shown in Figure 6.4.

6.1.3 Adaptive thresholding

Thresholding with a single threshold can fail to extract a whole object, depending on the background illumination, for example. One approach is to divide the image into sub-images such that the illumination in each of the sub-images is approximately uniform. Then a threshold is calculated for each sub-image. The adaptivity comes from the dependence of the threshold on the pixel location. For more information regarding threshoding, please refer to Chapter 10 in [Gonzalez et al., 2004] and Chapter 2 in [Bovik, 2005].

Exercise 6.1 Design your own thresholding technique based on a priori knowledge of the image histogram to extract an object of your choice from one of the image in:
W:\EC\STUDENT\m19mse\SampleImages.
Compare the result of your algorithm with some MATLAB® built-in functions.

6.2 Histogram stretching

In some circumstances the range of grey level within an histogram does not cover the whole range of possible grey levels. For example, if an image is underexposed

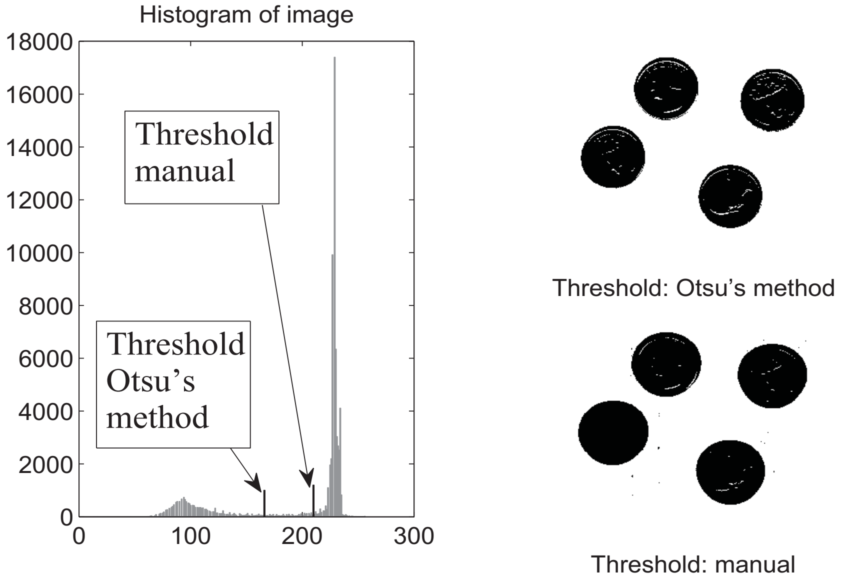


Figure 6.4: Example of separating coins using Otsu's method.

it will look dark and the highest pixel value may not reach 255 (assuming an 8 bit image). It is possible to increase the contrast in such an image by ensuring that the grey levels are re-scaled, such that they start at zero (by subtracting the minimum grey level in the image from all the pixels within the image and then multiplying the resulting image by the ratio of the ideal range of grey levels, that is $256 - 1 = 255$, divided by the current range of grey levels).

The following MATLAB® example illustrates a possible implementation, within MATLAB®, of *histogram stretching* based on Chapter 2 of the Handbook of image and video processing by Al Bovik [Bovik, 2005]. Note that histogram stretching is also referred to as *contrast stretching*. It will change the image grey levels into a number of distinct grey levels separated from each other. This will result in a non-smooth histogram appearing as an inverted comb with separated peaks. If it is then necessary to use an algorithm to detect peaks and valleys, it will be necessary to ignore all the bins within the histogram that do not have any pixels, that is ignore the zero in the histogram 'count'.

```
% Load the image into MATLAB
DIRimages='C:\MATLABImagProc\SampleImages\';
[A,map]=imread([DIRimages,'rice.png']);
%
% Histogram stretching
% uses g(n) = FSHS(f) = (K-1)/(B-A)*[f(n)-A]
% see p28 Handbook of image and video processing where
% K = 256 no of grey levels, A= min grey value,
% B = max grey value and f(n) is the image
NoGreyLevels=256;
[Apix,Aval]=imhist(A,NoGreyLevels);
%
% Convert to double to be able to use min,max and arithmetic
% operators to calculate min and max grey levels
Adouble=double(A);
MinGrey = min(min(Adouble)); MaxGrey = max(max(Adouble));
% Multiply the original image matrix by the inverse of the
% ratio between the current range of grey level divided by the
% required range of grey levels.
Astretched=uint8((NoGreyLevels-1)/(MaxGrey-MinGrey)*...
...(double(A)-MinGrey));
%
% Check the result on the histogram
[ASpix,ASval]=imhist(Astretched,NoGreyLevels);
MinGreyS = min(min(Astretched)); MaxGreyS = max(max(ASval));
%
% plot the graphs
figure(1); clf
stem(Aval,Apix)
hold on
stem(ASval,ASpix,'c');
title('Effect of histrogram stretching');
legend('Original image histogram','Stretched image histogram');
figure(2); clf
subplot (121),imshow(A),title('Original image')
subplot (122),imshow(Astretched),title('Stretched image')
```

Histogram stretching can be obtained using the MATLAB® command `J = imadjust(I)`. It maps the values in intensity image **I** to new values in **J** such that 1% of data is saturated at low and high intensities of **I**. Note that saturating 1% of the data enables one to ignore outliers, namely infrequent pixels at 0 or 255, which have no significance. The following syntax can be written to perform contrast adjustment:

```
J = imadjust(I,[low_in; high_in],[low_out; high_out])
J = imadjust(...,gamma)
newmap = imadjust(map,[low_in high_in],[low_out high_out],gamma)
RGB2 = imadjust(RGB1,...)
```

This increases the contrast of the output image **J**. The syntax `J = imadjust(I)` is equivalent to `imadjust(I,stretchlim(I))`. To find limits to stretch the contrast of an image, use the function `LOW_HIGH = stretchlim(I,TOL)`, which returns a pair of intensities that can be used by `imadjust` to increase the contrast of an image. The command `TOL = [LOW_FRACT HIGH_FRACT]` specifies the fraction of the image to saturate at low and high intensities. An example illustrating the histograms of an original image and the stretched images is shown in Figure 6.5. In addition, an image of some grains of rice, together with the corresponding image after performing histogram stretching, is illustrated in Figure 6.6.

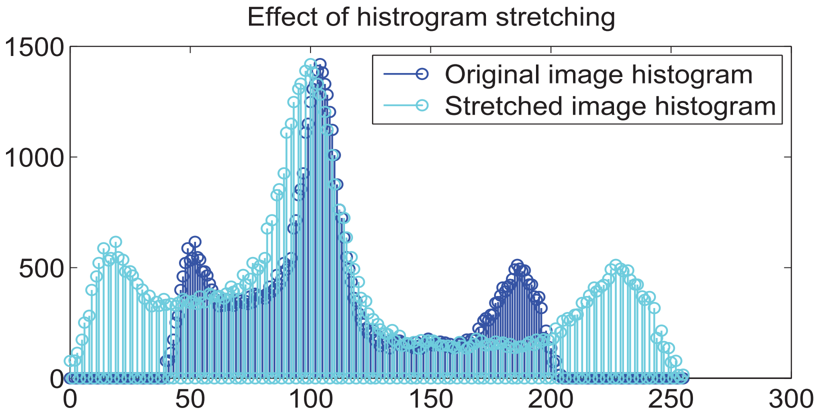


Figure 6.5: Histograms of the original and stretched images.

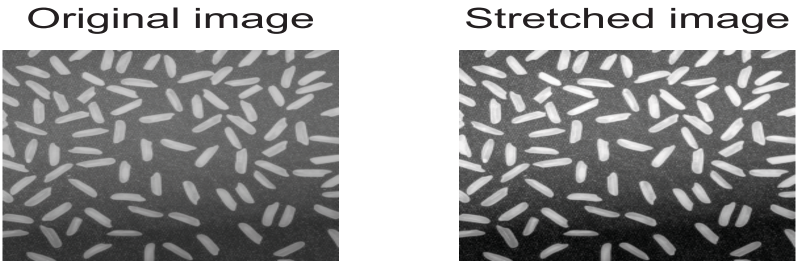


Figure 6.6: Images illustrating the use of a MATLAB® function to adjust the image grey levels.

Exercise 6.2 Write your own MATLAB® code to implement contrast or histogram stretching such that 5% of the pixels within the image becomes saturated. Check the outcome by displaying the histogram of the resulting image and comparing it with the original image.
Hint: use the command `stretchlim` or the function `find` to identify the pixels with minimum, as well as maximum, values.

The following section deal with a further refinement of the idea of re-mapping the range of grey levels within an image, using histogram equalization techniques.

6.3 Histogram equalisation

Histogram equalisation is useful for interpreting images and differentiating between grey levels that are close together. It can happen that most of the pixels within an image have only a small number of distinct grey levels, thereby not using the full range of possible grey levels for that particular image type. To improve the contrast of the image, it is possible to spread the grey levels over the whole range of possible values, either equally or with particular histogram shapes. By default, histogram equalisation aims to produce a flat histogram. It is, however, possible to shape the histogram with other distributions, than a uniform one, in order to adapt to specific environments; for example, underwater, night, neon lighting. Note that what may look good for human understanding may not add significant advantages for automatic segmentation and object recognition. Hence, histogram equalization should be used with care. Indeed, histograms that have only a two or three small grey value range may not look any better once equalised.

The main idea of histogram equalisation is to even out the number of grey levels within an image by regrouping them, or separating them, to increase the contrast between regions of interest that may have similar grey levels. Histogram equalisation performs the following steps:

1. calculate the image histogram;

2. calculate the cumulative image histogram (namely the cumulative sum of pixels in the various bins);

3. set $\mathbf{B}(x, y) = round((2^k - 1)/(N \times M) \times Hc(\mathbf{A}(x, y)))$, where k is the number of bits in the image $\mathbf{A}$, N and M are the number of rows and columns in the image, respectively, Hc is the cumulative histogram.

This formulation can be implemented as a look-up table, which maps an input (the range of grey level values from 0 to 255) to an output (the normalised cumulative image histogram such that its maximum value is 255). Each image pixel can then be transformed by changing its current value (the input of the look-up table) to equal its corresponding values in the normalized cumulative histogram. For example, let us assume that an image contains 1000 pixels with values between 0 and 255 and that 50 pixels have grey levels between 0 and 25, and 500 pixels had grey levels between 0 and 100. Normalising the cumulative histogram is equivalent to dividing 255 by the number of pixels in the image and multiplying the result by the number of pixels between 0 and a particular grey level. Hence, $50 \times 255/1000$ and $500 \times 255/1000$ gives 13 and 128, respectively, after rounding. This means that the grey level 25 and 100 on the original image will be replaced by the grey levels 13 and 128, respectively, on the equalised image. The grey level 25 has been reduced as only a small proportion of the pixels were between 0 and 25, whereas the grey level 100 has been increased because half of the pixels have a grey level of 100 (and 100 is less than half of 255).

```
[Aw,Bw]=imhist(A,256); % calculate the histogram using 256 bins
HistCumSum =cumsum (Aw); % calculate the cumulative sum
% normalise the cummulative histogram between 0 and 255
H=round((256-1)*HistCumSum/prod(size(A)));
% Compare the image grey values to the input of the look-up
% table [0 255] to generate a matrix whose values correspond
% to the output H of the look-up table.
B=uint8(interp1([0:1:255],H,double(A)));
```

The above MATLAB® code is equivalent to the MATLAB® function `histeq`. The process of histogram equalisation is illustrated in Figure 6.7.

Other MATLAB® commands that are useful are the following.

- `J = histeq(I,NoOfBins)` enhances the contrast of images by transforming the values in an intensity image, or the values in the colormap of an indexed image, so that the histogram of the output image approximately matches a specified histogram. (By default, `histeq` tries to match a flat histogram with 64 bins, although one can specify a different histogram instead; see the reference page in MATLAB® for `histeq`.)

- `J =adapthisteq(I,'NumTiles',Val1,'ClipLimit',Val2,'NBins',..`
  `.....Val3,'Range',Val4,'Distribution',Val5,'Alpha',Val6)`
  enhances the contrast of the intensity image `I` by transforming the values using contrast-limited adaptive histogram equalization (CLAHE); see, for example, pp. 474–485 in [Zuiderveld, 1994]. CLAHE operates on small regions in the image, called tiles, rather than the entire image. Each tile's contrast is enhanced, so that the histogram of the output region approximately matches the histogram specified by the 'distribution' parameter which can be:

  'uniform' (default) - this produces a flat histogram;

  'rayleigh' - a bell-shaped histogram is obtained in this case;

  'exponential' - this displays a curved histogram.

  After performing the equalization, `adapthisteq` combines neighbouring tiles using bilinear interpolation to eliminate artificially-induced boundaries. To avoid amplifying any noise that might be present in the image, one can use `adapthisteq` optional parameters (see MATLAB® help) to limit the contrast, especially in homogeneous areas. An example of the

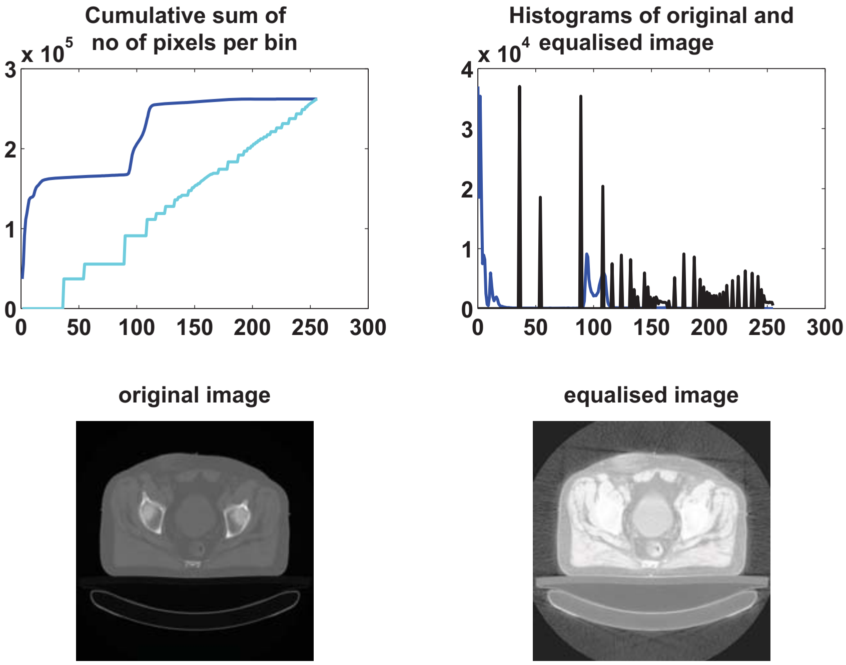


Figure 6.7: Images illustrating the use of a MATLAB® function to adjust the image grey levels.

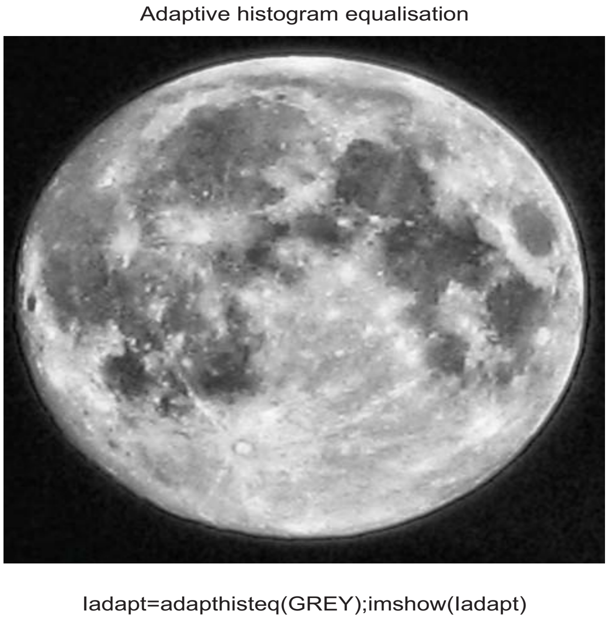

Adaptive histogram equalisation
Iadapt=adapthisteq(GREY);imshow(Iadapt)

Figure 6.8: Images illustrating the use of a MATLAB$^{\circledR}$ function to adjust the image grey levels.

use of adaptive histogram equalisation is given with the example of an image of the moon over Coventry (U.K.), see Figure 6.8.

The following MATLAB® commands illustrate histogram shaping. Initially, the histogram of the original image is obtained, then histogram equalisation is performed and the histogram of the equalised image calculated. Then, shaping the histogram is performed and the histogram of the equalised image is obtained.

```
% Illustrating histogram shaping
DIRimages=’C:\MATLABImagProc\SampleImages\’;
[A,map]=imread([DIRimages,’v21.bmp’]);
%
% Calculate the histogram of the original image.
[NoPix, GreyVal]=imhist(A);
%
% Use either uint8 or indexed images.
% By default, uses a flat distribution to remap the grey levels
% such that the resulting histogram should have the same number
% of pixels in each bin.
NoBins=8;
NoPixInImg=prod(size(A));
%
% Flat histogram
hgram1 = ones(1,NoBins)*NoPixInImg/NoBins;
%
% Histogram equalisation
I1=histeq(A,hgram1); % same as:  I1=histeq(A);
%
% Calculate the histogram of the equalised image.
[NoPix1, GreyVal1]=imhist(I1,length(hgram1));
%
% Shaped histogram
hgram2 = [1 0.8 0.3 0.6 0.4 0.8 0.6 0.2]*prod(size(A))/NoBins;
%
% Histogram equalisation (shaping)
I2=histeq(A,hgram2);
%
% Calculate the histogram of the equalised (histogram shaping)
% image
[NoPix2, GreyVal2]=imhist(I2,length(hgram2));
```

Finally, the images are generated, including plots of the normalised histograms.

```
% Display results in figures
%
figure(1); clf
subplot(221), imshow(A), title('Original image');
subplot(223), imshow(I2), title('Histogram shaping')
subplot(122),
% nomalising the histogram
%
plot(GreyVal1,NoPix1/max(NoPix1))
hold on
plot(GreyVal,NoPix/max(NoPix),'r')
plot(GreyVal2,NoPix2/max(NoPix2),'g')
plot(GreyVal2,hgram2/max(hgram2),'go')
plot(GreyVal1,hgram1/max(hgram1),'+')
axis([0 255 0 1.01])
title('Effect of different equalisation strategies')
```

Some images illustrating the use of histogram shaping are shown in Figure 6.9.

Exercise 6.3 Plot the histograms of an image, from the directory W:\EC\STUDENT\m19mse\SampleImages, stretch the image histogram and compare the resulting histogram with the original image. Comment on the outcome of the operation on the resulting image.

Exercise 6.4 Apply histogram equalisation techniques, with images from from the directory W:\EC\STUDENT\m19mse\SampleImages, and verify the effect of the equalisation by displaying the histogram of the equalised image and comparing it with the histogram of the original image. Try to improve the results obtained by using the default settings.

6.4 Summary

This chapter has presented some methods to analyse the information within an image and enhance the image using its histogram. It has been seen that a histogram can provide useful information as to the nature of the image taken and the number of regions of interest. A histogram can be used to select the range

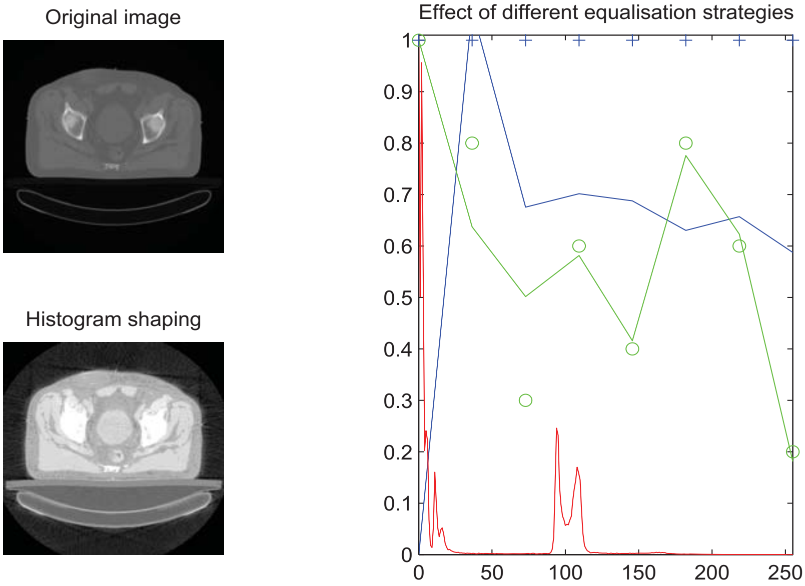


Figure 6.9: Images illustrating the use of histogram shaping.

of grey level for regions of interest to be segmented and analysed in order to be automatically identified. Certain operations, such as histogram stretching and equalisation, can effectively improve the contrast of an image. Adaptive histogram equalisation is the technique of choice when the image contains regions with significantly different grey levels. Histogram equalisation and stretching is, however, mainly a tool to improve the visual aspect of an image and may not be appropriate if automatic techniques, exploiting the shape of the histogram, are used to identify regions characterised by a set of valleys surrounding a peak in the histogram (the peak corresponding to a group of pixels with similar grey levels). For more information see Section 3.2 in [Gonzalez et al., 2004], Chapter 4 in [Sonka et al., 1999] and Chapter 2 in [Bovik, 2005].

Chapter 7

Grey level operations

7.1 Point operation on images

Point operations or grey level operations perform a grey level transformation on each pixel of the form:

$$\begin{array}{ll} \mathbf{B}(x,y) = f(\mathbf{A}(x,y)) \text{ for each } (x,y), & \text{if } Min \leq \mathbf{B}(x,y) \leq Max, \\ \mathbf{B}(x,y) = Min, & \text{if } \mathbf{B} < Min, \\ \mathbf{B}(x,y) = Max, & \text{if } \mathbf{B} > Max. \end{array}$$

where f is called a transfer (or mapping) function, **A** the original image, **B** the transformed image, (x, y) the pixel coordinates, Min represents the lowest possible grey level and Max the maximum allowed grey level. It is important to note that throughout this section the above Min- Max *constraints are assumed.*

When used with `uint8` or `uint16` data, each arithmetic function rounds and saturates its result before passing it on to the next operation. This can significantly reduce the precision of the calculation. A better way to perform this calculation is to use the `imlincomb` function. In this case, `imlincomb` performs all the arithmetic operations, in the linear combination, in double precision and only rounds and saturates the final result.

Remark 7.1 *Such operations are independent on the neighbourhood of the pixel, or its location.*

Since images are represented by matrices, addition and subtraction operations can be performed in a straightforward manner. Other operations such as multiplication and division can be achieved in MATLAB® using `.*` and `./` .

Remark 7.2 *To use standard arithmetic operators, you must convert the images to class* `double`*. The Image Processing Toolbox includes a set of functions that implement arithmetic operations for all numeric, non-sparse data types. The advantages are that no conversion to the* `double` *data type is necessary and overflow is handled automatically via application of the Min-Max constraints, although this can be an issue when several operations are carried out one after another.*

7.2 Arithmetic operators in the MATLAB® image processing toolbox

The image processing toolbox in MATLAB® supports the following operations.

imabsdiff(Img1, Img2): The absolute difference of two images.

imsubtract(Img1, Img2): Subtraction of two images.

imadd(Img1, Img2): Addition of two images.

imcomplement(Img1): The complement an image.

imdivide(Img1, Img2): Division of two images.

immultiply(Img1, ImgOrNumber): Multiplication of images.

imlincomb(Scalar1,Img1,Scalar2,Img2. . .): Computes the linear combination of two images.

7.2.1 Subtraction: brightness, negative, and movement

Brightness adjustment

When a constant positive value is removed from the pixels of $\mathbf{A}(x, y)$ it results in a darker image, as shown in Figure 7.1.

Exercise 7.1 Select an image from the SampleImages directory and assess the effect of

(i) `OutputImage = imabsdiff (Img1, Img2)`: Absolute difference of two images;

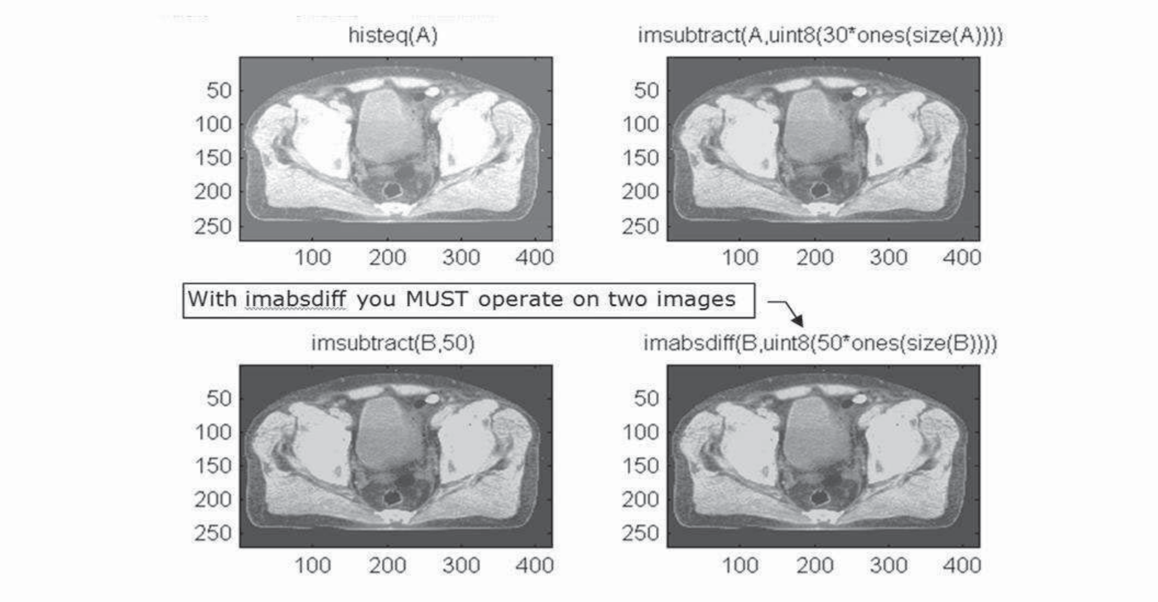

histeq(A)
imsubtract(A,uint8(30*ones(size(A))))
50
100
150
200
250
100
200
300
400
With imabsdiff you MUST operate on two images
imsubtract(B,50)
imabsdiff(B,uint8(50*ones(size(B))))

Figure 7.1: Images illustrating the use of subtraction to darken an image.

(ii) `OutputImage = imsubtract(Img1, Img2)`: Subtraction of two images.

Negative image

Since the maximum value of grey level is: $2^k = L - 1$, where k is the number of bits, typically 8, such that L is 256, with grey levels from 0 to 255, the appropriate MATLAB® commands for producing a negative image are: `B(x,y) = L-1-A(x,y)` or `NegativeImage = 2∧k-1-OriginalImage`. Images, illustrating the use of alternative MATLAB® commands to calculate a negative image, are shown in Figure 7.2.

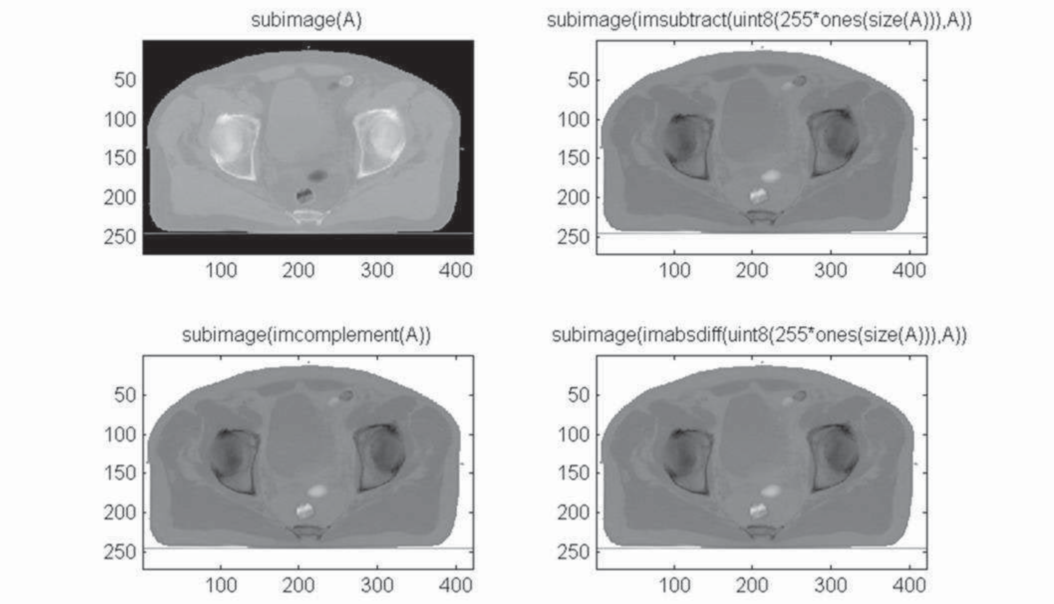

subimage(A)
subimage(imsubtract(uint8(255*ones(size(A))),A))
subimage(imcomplement(A))
subimage(imabsdiff(uint8(255*ones(size(A))),A))
50
100
150
200
250
100
200
300
400

Figure 7.2: Images illustrating the use of alternative MATLAB® commands to calculate a negative image.

Exercise 7.2 Select an image from W:\EC\STUDENT\m19mse\SampleImages and find its negative using `imcomplement`, `imabsdiff`, `imsubtract` and standard matrix operations (i.e. in `double` to be converted to `unit8` for display).

Movement detection

Let us consider a video sequence. When an image $\mathbf{A}(x, y, t1)$ taken at time $t1$ is subtracted from another image $\mathbf{A}(x, y, t2)$ taken at time $t2$ using the same *fixed* camera, it is possible to detect objects that have moved in the scene. A common application of this technique are speed cameras. An appropriate MATLAB® command is `B(x,y) = A(x,y,t1) - A(x,y,t2)`. The following commands are useful.

`imabsdiff (Img1, Img2)`: Absolute difference of two images;
`imsubtract(Img1, Img2)`: Subtraction of two images.

Images illustrating the use of `imabsdiff` to detect movement are given in Figure 7.3.

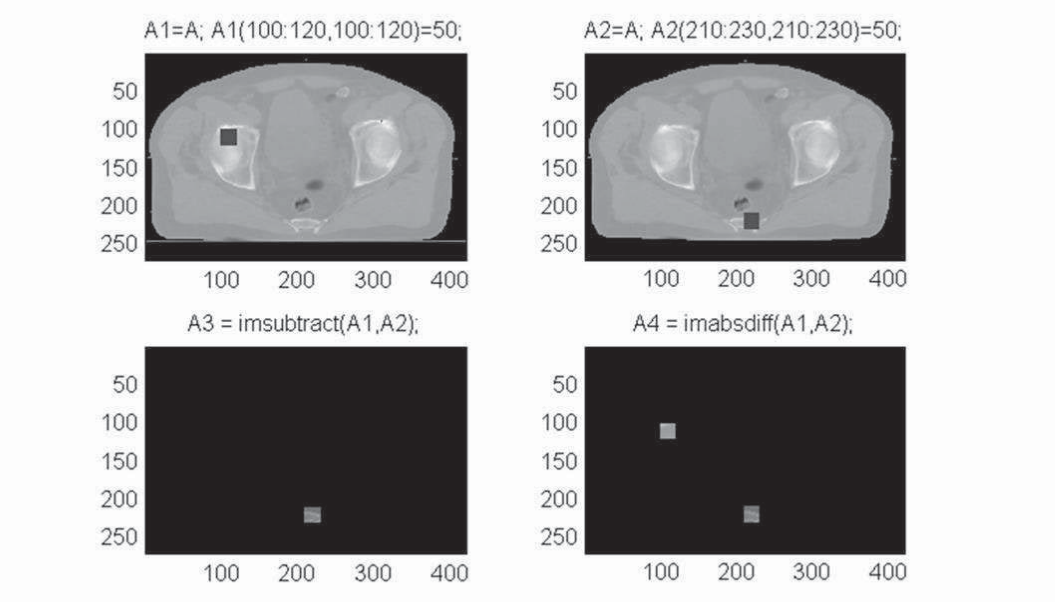


Figure 7.3: Images illustrating the use of `imabsdiff` to detect movement.

Exercise 7.3 To find a square located at different position within an image, the MATLAB® commands `imabsdiff` and `imsubtract` can be used. Comment on the effect of the these two functions.

Exercise 7.4 Study the effect of the location and grey level value of the square shape superimposed on various images, such as the image stored in the file `cameraman.tiff`. You should notice that depending on the grey level value selected, you may not always be able to detect it.

Note that image subtraction is an effective tool to remove unwanted objects from a picture. For example, suppose that one is trying to take a photograph of a monument and there are pedestrians, cyclists or cars passing in front of the monument. If one takes several images of the monument from the same location, then it is possible to subtract the different images to identify the region of the image that has changed. To create an image without an unwanted object, one can then replace the part of the image with an object by the same part of another image without that object.

7.2.2 Addition: brightness adjustment

When a constant positive value is added to the pixels of $\mathbf{A}(x, y)$ it results in a brighter image, which is illustrated in Figure 7.1. The MATLAB® command:

```
imadd (Img1, ImgOrScalar)
```

can be used to perform the calculation $\mathbf{B}(x, y) = \mathbf{A}(x, y) + \text{constant}$.

7.2.3 Multiplication: contrast adjustment

To determine $\mathbf{B}(x, y) = \mathbf{A}(x, y). \times \text{constant}$, the command:

```
immultiply(Img1, ImgOrScalar)
```

can be used.

7.2.4 Division: contrast adjustment

To adjust contrast, the MATLAB® command:

```
imdivide(Img1, ImgOrScalar)
```

can be used to determine $\mathbf{B}(x, y) = \mathbf{A}(x, y))./\text{constant}$.

7.2.5 Linear combination: stretching and contrast adjustment

For contrast adjustment and stretching, the command:

```
imlincomb(K1,A1,K2,A2,...,Kn,An)
```

can be used to determine

$$\mathbf{B}(x, y) = \text{constant1} \times \mathbf{A1}(x, y)) + \text{constant2} \times \mathbf{A2}(x, y)) + \dots,$$

where **A1** and **A2** denote images, which may be identical.

Remarks 7.3 *Now* `Z = imlincomb(K1,A1,K2,A2,...,Kn,An)` *computes*

$$K1 \times \mathbf{A1} + K2 \times \mathbf{A2} + ... + Kn \times \mathbf{An},$$

where $K1$*,* $K2, \dots Kn$ *are real, type* `double` *scalars and* **A1***,* **A2***, . . .* **An** *are real, nonsparse, numeric arrays with the same class and of the same size. Note that* **Z** *has the same class and size as* **A1***. Values above 255, in the case of* `unit8` *images, are automatically truncated. In fact, the command* `imlincomb` *can be use to prevent some of this truncation happening at an early stage. Here,* `imlincomb` *performs the addition and division in double precision and only truncates the final result.*

In the version that uses nested arithmetic functions, `imadd` *adds 255 and 50 and truncates the result to 255 before passing it to* `imdivide`*. The average returned in* $\mathbf{Z}(1, 1)$ *is 128. To illustrate this, consider the following. The commands:*

```
X = uint8([255,10,75;44,225,100]);
Y = uint8([50,20,50;50,50,50]);
Z = imdivide(imadd(X,Y),2)
```

produces the results

```
Z =
128 15 63
47 128 75
```

and

```
Z2 = imlincomb(.5,X,.5,Y)
```

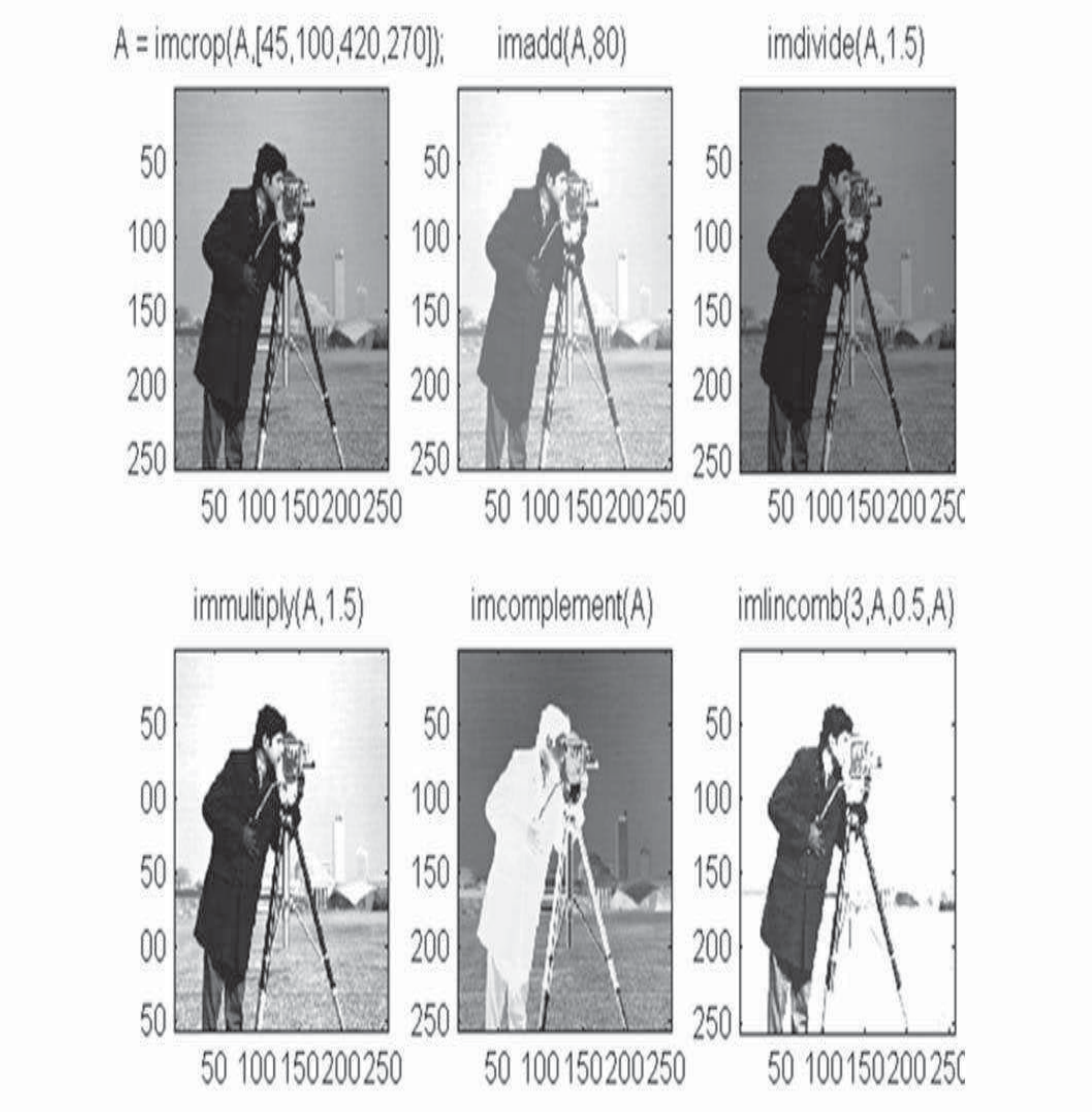


Figure 7.4: Images illustrating the use of Addition, Division, Multiplication, Complement and Linear combination applied to the image stored in `cameraman.tiff`.

produces the results

```
Z2 =
153 15 63
47 138 75
```

7.3 Log transformation

The *log transform* has the following structure:

$$s = c\log(r+1), \quad \text{where } c \text{ is a constant and } r \geq 0.$$

An example of a corresponding MATLAB® command having this form is

```
B = c*log10(A + 1 )
```

A particular example of a MATLAB® command is

```
DisplayFrequencySpectrum = 70*log10( abs( fft2(A) )+1 )
```

The log transform maps a narrow range of low grey level values in the input into a wider range of output levels. The opposite is true of higher values of input levels.

7.4 Power law transformation

The *power law transform* has the following structure:

$$s = c(r+\varepsilon)^{\gamma}, \quad \text{where } c \text{ and } r \text{ are positive constants and } \varepsilon \text{ is an offset.}$$

An example of a corresponding MATLAB® command having this form is

```
B = c*(A + e).∧g
```

Note that if $\gamma < 1$, then it maps a narrow range of dark input values into a wider range of output values. Also, the power law transform is more flexible than the log transform.

The square root transformation function:

$$\mathbf{B}(x,y) = \sqrt{\mathbf{A}(x,y) \times 255}$$

can be used to brighten the image, stretch dark regions and reduce contrast of bright regions.

Chapter 8

Spatial and frequency domain filtering

Many applications, such as edge detection, smoothing, etc., make use of *filters*, also referred to as masks, kernels, templates or windows.

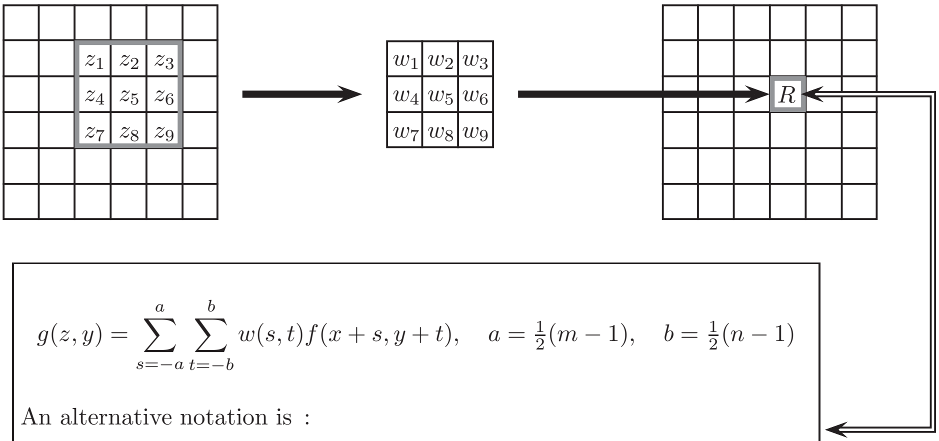
Filters in the spatial and the frequency domain constitute a Fourier transform pair. This means that it is possible to design a filter in the frequency domain and then obtain a corresponding filter in the spatial domain.

Remark 8.1 *According to Section 4.2 in [Gonzalez et al., 2004], it can be more effective and efficient to start designing a filter in the frequency domain and then find an equivalent filter in the spatial domain. The main reason for this is that it is possible to represent* ***spatial filters*** *by relatively small matrices (of order* 3×3*) in the spatial domain.*

8.1 Spatial filters

The basic idea is to move a filter or mask over each pixel in the original image. At each point, calculate the response of the filter using a predefined relationship (for more details see Section 3.5 in [Gonzalez et al., 2004]).

For *linear filters*, the relationship is calculated as the sum of the products of the filter coefficients by their corresponding pixels in the image. This process for obtaining the filtered image $g(z, y)$ (or R) is called *correlation*.



$$g(z,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t), \quad a = \tfrac{1}{2}(m-1), \quad b = \tfrac{1}{2}(n-1)$$

An alternative notation is :

$$R = \sum_{i=1}^{m \times n} w_i z_i = \sum_{i=1}^{9} w_i z_i$$

8.2 Two dimensional convolution and correlation in MATLAB®

8.2.1 Convolution

Linear filtering of an image is accomplished through an operation called *convolution*. In convolution, the value of an output pixel is computed as a weighted sum of neighbouring pixels. The matrix of weights is called the *convolution kernel*, also known as the *filter*.

For example, suppose the image is

$$\mathbf{A} = \begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

and the convolution kernel is

$$\mathbf{h} = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}.$$

The $\mathbf{A}(2,4)$ output pixel is calculated as follows: Rotate the convolution kernel 180° about its center element to produce

$$\mathbf{H}_{\text{rotated}} = \begin{bmatrix} 2 & 9 & 4 \\ 7 & 5 & 3 \\ 6 & 1 & 8 \end{bmatrix}.$$

Slide the center element of the convolution kernel so that it lies on top of A(2,4), the element on the second row and fourth column of **A**. Multiply each weight in the rotated convolution kernel by its corresponding grey level in the image pixel **A**.

$$\begin{bmatrix} 1 & 8 & 15 \\ 4 & 14 & 16 \\ 13 & 20 & 22 \end{bmatrix} \boldsymbol{.}* \begin{bmatrix} 2 & 9 & 4 \\ 7 & 5 & 3 \\ 6 & 1 & 8 \end{bmatrix}.$$

In this case, applying the convolution to calculate the filtered value for the position (2,4) would give 575. Such a value, being larger than 255 (the maximum possible value for an 8 bit image), will be truncated to 255.

8.2.2 Correlation

The operation called *correlation* is closely related to convolution. In correlation, the value of an output pixel is also computed as a weighted sum of neighbouring pixels. The difference is that the *matrix of weights*, in this case called the *correlation kernel*, is not rotated during the computation.

The MATLAB® implementation is as follows.

```
A = [1 8 15;7 14 16;13 20 22]
B = [8 1 6;3 5 7;4 9 2]
```

Note that, in this case the matrix representing the filter has been rotated by 180 degrees.

$$\begin{bmatrix} 1 & 8 & 15 \\ 4 & 14 & 16 \\ 13 & 20 & 22 \end{bmatrix} \boldsymbol{.}* \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}.$$

In the case of correlation, the sum of the individual products is equal to 585, which would also be truncated to 255. It is common practice to design *rotationally invariant spatial filters* such that the convolution or correlation would lead to an identical result. The MATLAB® command `C = conv2(A,B,'shape')` uses a straightforward formal implementation of the two-dimensional convolution equation in spatial form. If a and b are functions of two discrete variables, n_1 and n_2, then the formula for the two-dimensional convolution of a and b is

$$c(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2).$$

In practice however, `conv2` computes the convolution for finite intervals. If the `'shape'` argument is specified as `'Full'`, MATLAB® returns the full two-dimensional convolution, which is the default. If `'shape'` is specified as `'Same'`, MATLAB® returns the central part of the convolution, which is of the same size as **A**. If `'shape'` is replaced by `'Valid'`, MATLAB® returns only those parts of the convolution that are computed without the zero-padded edges. Using the `'Valid'` option, the resulting image, which is smaller than the original image (of size $ma \times na$), is dependent on the filter dimension ($mb \times nb$), and has size $(ma - mb + 1) \times (na - nb + 1)$.

8.2.3 Filtering

The MATLAB® command `Y = filter2(H,X,'shape')` filters the data in **X** with the two-dimensional FIR filter in the matrix **H**. It computes the result (**Y**) using two-dimensional correlation, and returns the central part of the correlation that is the same size as **X**. The command `filter2` uses `conv2` to compute the full 2-D convolution of the FIR filter. By default, `filter2` then extracts the central part of the convolution that is the same size as the input matrix, and returns this as the result. If the `'shape'` parameter specifies an alternate part of the convolution for the result, `filter2` returns the appropriate part. Note that `filter2` rotates the filter matrix 180° to create a convolution kernel. It then calls `conv2`, the two-dimensional convolution function, to implement the filtering operation. The rotation of the matrix, prior to calling `conv2`, cancels out the rotation of the kernel that occurs in `conv2` when the convolution is implemented.

The MATLAB® command `B = imfilter(A,H,Options)` filters the multi-dimensional array **A** with the multi-dimensional filter **H**.

Boundary Options

Input array values, outside the bounds of the array, are implicitly assumed to have the value `X`, with default value zero.

`'symmetric'`: This option mirror-reflects the array across the array border.

`'replicate'`: This option adds additional rows and column where the individual pixel values are identical to that of the border of the image. This process is repeated for each row and column to be added to the image.

`'circular'`: This option adds additional rows and column to the edge of the image, assuming the input grey level pattern in the image is periodic. For example, if a pattern of grey level intensity follows a sine wave, then the padded element will carry on the sine wave pattern.

Size Options

These adjust the size of the output array.

`'same'`: This results in the same size as the input array (which is the default).

`'full'`: This produces the 'full' filtered result, and so is larger than the input array.

Correlation and Convolution Options

`'corr'`: This option filters using correlation, which is the default, and produces the same result as `filter2`.

`'conv'`: This filters using convolution and so rotates the filter matrix by 180°.

Example 8.1 This example illustrates the use of the MATLAB® commands: `conv2`, `filter2` and `imfilter`, applied to an image.

Image	Size of array	Data type
A	246 × 300	73800 uint8 array
B1	248 × 302	599168 double array
B2	246 × 300	590400 double array
B3	246 × 300	73800 uint8 array

Note that, using `conv2` and `filter2`, the output matrix is bigger than the input matrix (which represents the original image). These functions require the user to specify the filter matrix, which is to be applied to the image. Problem

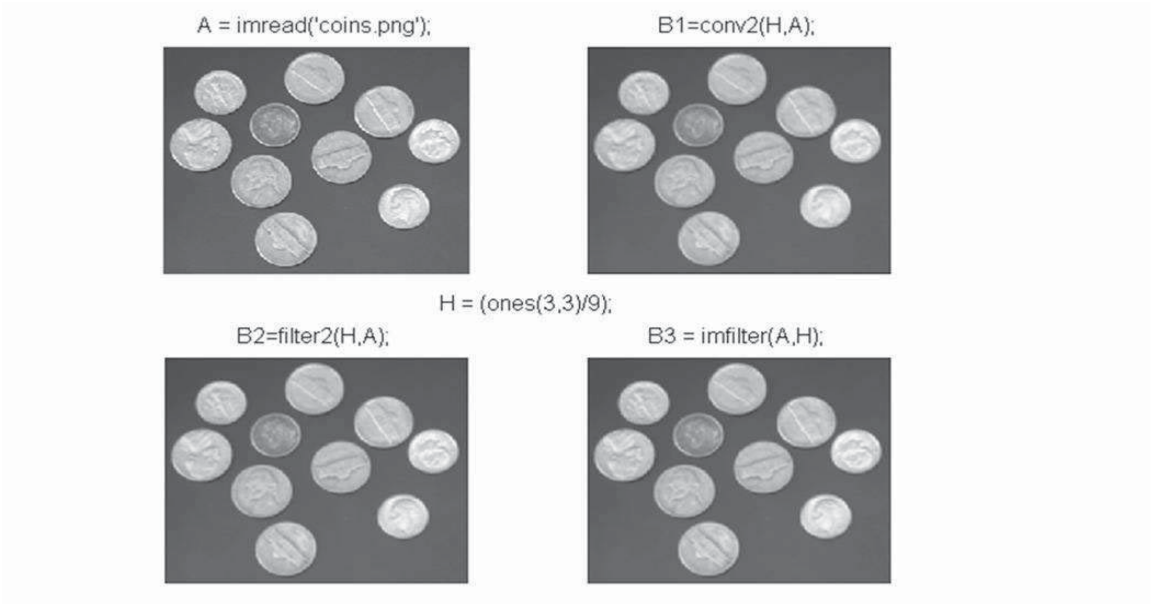


Figure 8.1: Filtered images produced by the MATLAB® functions `conv2`, `filter2` and `imfilter`.

specific filters can thus be defined if the filters, implemented in MATLAB®, are not appropriate. The function `imfilter` produces an output whose data type is the same type as the input image. It converts the image and the filter matrices to double type, in order to perform the convolution and then convert the output to the same format as the image to be filtered.

8.3 Using a MATLAB® pre-defined filter in the spatial domain

MATLAB® offers a number of filters to perform low pass or high pass filtering. Low pass filters are used to smooth images, whilst high pass filter enhance details such as edges. Some useful MATLAB® commands are the following.

`H=fspecial(filter_type, parameters)` This creates a two-dimensional filter **H**, for use with `imfilter(A,H)`, of the type specified in Table 8.3.

Command	Function
`gaussian`	Gaussian lowpass filter
`sobel`	Sobel horizontal edge-emphasizing filter
`prewitt`	Prewitt horizontal edge-emphasizing filter
`laplacian`	Filter approximating the two-dimensional Laplacian operator
`log`	Laplacian of Gaussian filter
`average`	Averaging filter
`unsharp`	Unsharp contrast enhancement filter

Table 8.1: MATLAB® image processing filter types.

`BW = edge(A,edge_finding_methods)` This returns a binary image `BW` of the same size as **A**, with 1's where the function finds edges in **A** and 0's elsewhere. To determine edges, MATLAB® uses edge-finding methods which look for places in the image where the intensity changes rapidly, by finding the first derivative of the intensity larger in magnitude than some threshold, or by finding the second derivative of the intensity that has a zero crossing. Some particular examples are:

```
BW = edge(I,’sobel’,thresh,direction)
BW = edge(I,’prewitt’,thresh,direction)
BW = edge(I,’roberts’,thresh)
BW = edge(I,’log’,thresh,sigma)
BW = edge(I,’zerocross’,thresh,h)
BW = edge(I,’canny’,thresh,sigma)
```

`B = medfilt2(A,[m n])` This performs median filtering (using a $m \times n$ neighbourhood) of the matrix **A** in two dimensions. The command `medfilt2` pads the image with 0’s on the edges, so the median values for the points within $[m/2, n/2]$ of the edges might appear distorted.

`B = ordfilt2(A,order,domain,S,padopt)` Here, using this command, two-dimensional order-statistic filtering is performed, where each element in **A** is replaced by the ‘order’th element in the sorted set of neighbours specified by the nonzero elements in the domain.

`J = wiener2(I,[m n],noise)` This command performs two-dimensional pixel-wise adaptive noise-removal filtering (low pass) based on statistics (mean and standard deviation) estimated from a local neighbourhood of each pixel. The additive noise (namely Gaussian white noise) is assumed to be white noise. The MATLAB® command `[J,noise] = wiener2(I,[m n])` also estimates the additive noise, before performing the filtering process. The command `wiener2` returns this estimate of the noise.

Some MATLAB® demonstrations can be obtained using the commands:

`firdemo`, which illustrates spatial and frequency filter design;
`edgedemo`, which uses the ‘edge’ function to apply different edge detection methods to a variety of images;
`nrfiltdemo`, which applies linear and non-linear filtering noise techniques to images corrupted with various source of noise.

Chapter 9

Smoothing, averaging and low pass filters

Smoothing filters are used to reduce noise in an image or for blurring. They result in the removal of small (unnecessary) details from an image, prior to the extraction of a large structure. They can also be used to bridge the gap between lines or curves, missing a few pixels, that are required to be connected.

The output of a smoothing filter is the average of the pixels contained in the neighbourhood of the filter mask. It results in an image with reduced sharp transitions in grey levels. The general formulation of a weighted average filter is:

$$f(x,y) = \frac{\displaystyle\sum_{s=-a}^{a}\sum_{t=-a}^{a} w(s,t)g(x+s,y+t)}{\displaystyle\sum_{s=-a}^{a}\sum_{t=-a}^{a} w(s,t)}, \qquad a = \frac{m-1}{2}, \quad b = \frac{n-1}{2},$$

where $w(s,t)$ corresponds to the weight associated with a particular element of the mask located at position (s,t) and $g(x+s,y+t)$ corresponds to the original image covered by the filter.

9.1 Mean filters

9.1.1 Arithmetic mean filter

The arithmetic mean filter is the average value of the original pixels $g(s,t)$ covered by the filter area denoted by $S_{x,y}$:

$$f(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{x,y}} g(s,t).$$

9.1.2 Geometric mean filter

The geometric mean is the product of all the elements covered by the filter area defined by S_{xy} elevated to the power $1/(mn)$:

$$f(x,y) = \left[\prod_{(s,t)\in S_{x,y}} g(s,t) \right]^{\frac{1}{mn}}.$$

This filter has a similar effect to the arithmetic mean filter, but tends to loose less details in the process. For more information on the geometric mean see: http://www.math.toronto.edu/mathnet/plain/questionCorner/geomean.html

9.1.3 Harmonic mean filter

The harmonic mean is calculated as follows:

$$\frac{mn}{\displaystyle\sum_{(s,t)\in S_{x,y}} g(s,t)}.$$

Such a filter works well with salt noise and Gaussian noise, but fails for pepper noise.

9.1.4 Contraharmonic mean filter

The contraharmonic filter is calculated as follows:

$$\frac{\displaystyle\sum_{(s,t)\in S_{x,y}} (g(s,t))^{Q+1}}{\displaystyle\sum_{(s,t)\in S_{x,y}} (g(s,t))^{Q}},$$

where Q is called the order of the filter. Positive values of Q eliminates pepper noise and negative values of Q eliminate salt noise.

Remark 9.1 *Note that this filter is equivalent to the arithmetic mean filter when $Q = 0$ and the harmonic mean filter when $Q = -1$.*

Exercise 9.1 Using `conv2`, `filter2` and `imfilter`, calculate the convolution of images from the directory W:\EC\STUDENT\m19mse\SampleImages and the following mask:

$$H = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

What do you observe?

Exercise 9.2 Write down the MATLAB® commands and observations to add Gaussian and 'salt' noise, 'pepper' noise and 'salt and pepper' noise and then filter them with an arithmetic mean filter.

Exercise 9.3 Write down the commands and observations to filter the images created in Exercise 7.1 with a geometric mean filter.

Exercise 9.4 Write down the commands and observations to filter the images created in Exercise 7.1 with a harmonic mean filter.

Exercise 9.5 Write down the commands and observations to filter the images created in Exercise 7.1 with a contraharmonic mean filter.

Exercise 9.6 Compare the effects of the filters introduced in this section.

9.2 Order-statistics filters

Order-statistics filters are spatial filters, whose response is based on ordering the pixels contained in the image area encompassed by the filter. The response of the filter at any point is determined by the ranking result.
The MATLAB® command

```
B=ordfilt2(A,NthElement,domain,S,padopt)
```

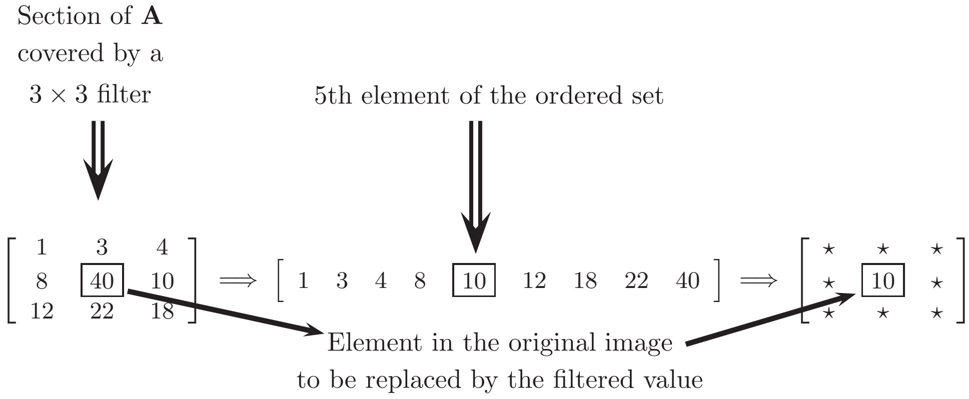
performs two-dimensional order-statistic filtering. This filter replaces each element in **A** by the `NthElement` element in the sorted set of neighbours specified by the nonzero elements in the domain. This means that for a 3 × 3 filter (9 elements in total), `ordfilt2` will replace the element corresponding to the central pixel in the original image, covered by the filter, by the element ranked at the nth position in the original image.

9.2.1 Median filter for bipolar and unipolar impulse noise

The MATLAB® command `B=ordfilt2(A,5,ones(3,3))` implements a 3 × 3 median filter. Consider, for example, the 3 × 3 median filter

$$\begin{bmatrix} 1 & 3 & 4 \\ 8 & 40 & 10 \\ 12 & 22 & 18 \end{bmatrix}.$$

The filtering process gives:



The MATLAB® command `B=medfilt2(A,[m n])` performs median filtering (using a $m \times n$ neighbourhood) of the matrix **A** in two dimensions. The command `medfilt2` pads the image with 0's on the edges, so the median values for the points within $[m/2, n/2]$ of the edges might appear distorted.

Exercise 9.7 Write down the MATLAB® commands which adds 'salt and pepper' noise to an image of your choice and then apply median filters of various sizes, using both `ordfilt2` and `medfilt2` commands. Comment on your results.

9.2.2 Minimum filter for finding the darkest points in the image, with ‘salt’ noise

The MATLAB® commands `B=ordfilt2(A,1,ones(3,3))` implements a 3×3 minimum filter and `B=ordfilt2(A,1,[0 1 0; 1 0 1; 0 1 0])` replaces each element in $\mathbf{A}$ by the minimum of its north, east, south, and west neighbours.

$$\begin{tikzcd}
\begin{bmatrix} 1 & 3 & 4 \\ 8 & 40 & 10 \\ 12 & 22 & 18 \end{bmatrix} \arrow[r, Rightarrow] & \begin{bmatrix} 3 & 8 & 10 & 22 \end{bmatrix} \arrow[r, Rightarrow] & \begin{bmatrix} 1 & 3 & 4 \\ 8 & \boxed{3} & 10 \\ 12 & 22 & 18 \end{bmatrix}. \\
\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \arrow[r] & \begin{array}{c}\text{Mask used to select} \\ \text{elements to be ordered}\end{array} \arrow[u] &
\end{tikzcd}$$

9.2.3 Maximum filter for finding the brightest points in the image, with ‘pepper’ noise

The MATLAB® command `ordfilt2(A,9,ones(3,3))` implements a 3×3 maximum filter, which finds the brightest points in the image, with ‘pepper’ noise.

9.2.4 Midpoint filter for randomly distributed noise

The midpoint filter calculates the midpoint between the maximum and the minimum values in the area encompassed by the filter. It can be defined as follows:

$$f(x, y) = \frac{1}{2}\left[\max_{(s,t)\in S_{x,y}} g(s, t) + \min_{(s,t)\in S_{x,y}} g(s, t)\right],$$

where $f(x, y)$ is the value of the filtered image at the point (x, y), $g(x, y)$ is the original image and $S_{x,y}$ is the area covered by the filter, centred on the set of coordinates (x, y).

9.2.5 Alpha-trimmed mean filter for a combination of noise types

This filter is similar to the arithmetic mean filter, but uses a smaller range of grey levels to calculate the average. This helps in the case where a few pixels have significantly different values than the remainder of the pixels. In particular it uses

$$m \times n - (d/2 \text{ lowest grey level values}) - (d/2 \text{ highest grey level values})$$

pixels, where m and n are the numbers of rows and columns in the image respectively and d is used to specify the number of high and low grey levels to ignore, in order to calculate the mean of the pixels covered by the filter area such that:

$$f(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{x,y}} g_{mn-d}(s,t),$$

where $g_{(mn-d)}(s,t)$ corresponds to the remaining $m \times n - d$ pixels in the original image, excluding the $d/2$ pixels with highest grey level values and the $d/2$ pixels with lowest grey level values.

Remark 9.2 *Note that, when* $d = 1$*, the alpha-trimmed mean filter is equivalent to the arithmetic mean filter.*

Exercise 9.8 Implement the midpoint filter using `ordfilt2` and/or other combinations of appropriate MATLAB® commands.

Exercise 9.9 Implement an alpha-trimmed mean filter using a combination of appropriate MATLAB® commands.

Exercise 9.10 Write a MATLAB® m-file to compare and demonstrate the effectiveness of the smoothing filters introduced in this Chapter.

This section has described the behaviour of number of typical low pass filters implemented in the spatial domain for use in image processing. Since low pass filtering will remove noise, as well as information that may be relevant, it is, therefore, necessary to establish the requirements of a smoothing filter for the particular application. It is common practice to apply a low pass filter before detecting the contour of regions of interest in the image using edge detectors. The next section, namely Section 9.3, describes a number of edge detectors that are widely available; some of which combine smoothing and edge detection. This section will describe a few noise models and highlight the issues associated in filtering images with noise and present means to reduce noise in the image for edge detection purposes.

9.3 Edge detection

An edge is a 'marked' transition between two regions of interest within an image. Ideally an edge exhibits a high degree of discontinuity. To detect this discontinuity it is possible to use filters based on derivatives. Indeed, the stronger a signal discontinuity, the higher will be its derivative. To alleviate this problem, an approximate derivative of the image can be used, which is effectively a low pass filter that reduces the noise. The following two subsections present edge detectors based on a gradient or first derivative, and the Laplacian or second derivative. Issues highlighted with such operators will then be highlighted in three figures.

9.3.1 The gradient (a first order derivative)

The gradient of a function f at coordinates (x, y) is defined as follows:

$$\boldsymbol{\nabla}\mathbf{f} \overset{\text{def}}{=} \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}.$$

In image processing, however, the gradient is often used to refer to the magnitude of the 'mathematical gradient', namely:

$$|\boldsymbol{\nabla}\mathbf{f}| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}.$$

To simplify the implementation of the gradient, it is common practice to approximate it with:

$$|\boldsymbol{\nabla}\mathbf{f}| \approx \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right| = G_x + G_y,$$

where $G_x = \left|\dfrac{\partial f}{\partial x}\right|$ and $G_y = \left|\dfrac{\partial f}{\partial y}\right|$. The direction of the gradient is also an important quantity that is perpendicular to the direction of the edge at that point. It is given by:

$$\tan^{-1}\left(\begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}\right).$$

The gradient can be implemented using masks of order $n \times n$, where n is an odd number. Consider the case when $n = 3$. It is assumed that the element calculated is the central element of the image covered by the mask (denoted z_5).

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

The elements of the mask along the vertical, horizontal or diagonals are approximated by the derivative operator, such that the product of these elements with the image result in an expression similar to:

$$\frac{\partial f}{\partial x}(x, y) \approx f(x+1, y) - f(x, y).$$

In theory, the sum of all the elements of the mask should be zero.

9.3.2 The Laplacian

The Laplacian (namely the Laplacian operator) is based on second order derivatives. It has the same properties in all directions (that is it is rotationally invariant) and is defined by

$$\nabla^2 f(x, y) \overset{\text{def}}{=} \frac{\partial^2 f}{\partial x^2}(x, y) + \frac{\partial^2 f}{\partial y^2}(x, y).$$

The Laplacian can be approximated by a convolution sum.

The following masks, of order 3×3, are often used:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

which return only edge information. The masks:

$$\begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

also return some information on the grey level.
Note the right mask uses the following in the vertical and horizontal direction:

$$\frac{\partial^2 f}{\partial x^2}(x,y) \approx f(x+1,y) + f(x-1,y) - 2f(x,y) \tag{9.1}$$

$$\frac{\partial^2 f}{\partial y^2}(x,y) \approx f(x,y+1) + f(x,y-1) - 2f(x,y). \tag{9.2}$$

where $f(\cdot,\cdot)$ denotes the image, and the coefficients of f, in (9.1)-(9.2), namely 1, 1, -2, are elements of the filter.

Issues with the Laplacian:

- Laplacian masks are rotationally symmetric, which means that edges in all orientation contribute to the result. They are, therefore, unable to detect edge direction.
- The Laplacian gives rise to a double edge.
- The Laplacian is extremely sensitive to noise.

The sign of the Laplacian indicates which side of the edge is the brighter, $(-)$, or the darker $(+)$. Therefore, if a negative sign is followed by a positive sign, this shows a transition from a light to a dark grey level (see [Gonzalez et al., 2009]). Laplacians are generally used on smoothed images (that is after a smoothing filter has been applied to the image). This helps reduce the negative effect of noise on the response of the filter. The response of the Laplacian is positive on one side of an edge and negative on the other side. This means that by adding some percentage of its response to the original image it is possible to sharpen the edge on an image, as shown in Figure 9.1. The top left illustration in Figure 9.1 represents the original image; the negative image of the Laplacian is shown in the top right of the figure; the edge enhanced image resulting from the subtraction of the Laplacian from the original image is shown in the bottom left of the figure, whilst a comparison between the image profiles for the original and the edge enhances image is given in the bottom right plot.

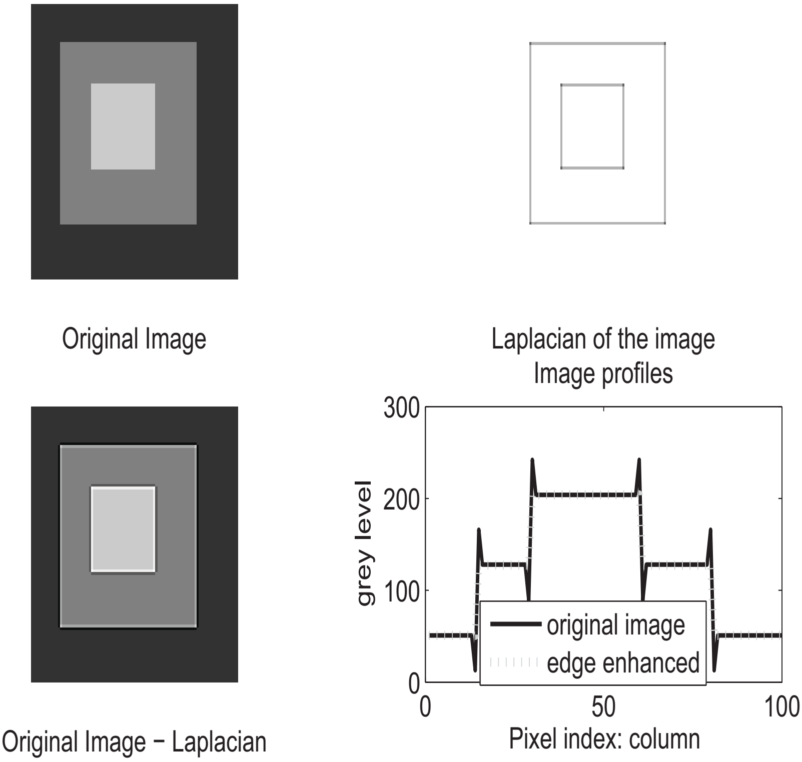


Figure 9.1: Illustrating the use of Laplacian to enhance the edges of regions of interest.

In theory, as mentioned previously, the sum of all the elements of the mask should be zero. When they are not, it means that information relative to the image grey level is also included in the filter.

To illustrate the issues associated with edge detection in the presence of noise an artificial image was created with a series of constant grey levels followed by a slope of slowly varying grey levels up to another plateau within which two step changes in grey level occurred, see Figure 9.2 (top left). The profile along one of the row of the image calculated using the function `improfile` is shown on the top right of Figure 9.2. The bottom two plots in Figure 9.2 show the first and the second derivative. The profile, along the image column, of the first derivative, implemented using the approximation $\frac{\partial f}{\partial x} \approx f(x+1) - f(x)$, and its second derivative, implemented using the expression $\frac{\partial^2 f}{\partial x^2}(x) \approx f(x+1) + f(x-1) - 2f(x)$, are shown in the bottom two figures. To detect the edge it is then necessary to specify a threshold value above which the gradient and/or the Laplacian response would indicate the presence of an edge. In the particular example of Figure 9.2, it is possible to take the absolute value of the gradient and specify a threshold of 10 to indicate strong edges at positions 43 and 47, respectively. The second derivative can be used to detect the start and the end of the slope, as well as the double step change at columns 43 and 47. Note that sharp edges such as the step change are characterised by two consecutive edges. Such feature can be exploited to increase the edge contrast. A good illustration of the effect of noise on edge detection is given in Section 10 of [Gonzalez et al., 2004].

Figure 9.3 illustrates that, with a small amount of Gaussian noise, the edge detection process will be slightly more complex as false edges, characterised by a non-null gradient, are created. This spurious response is emphasised by the Laplacian. In particular, one additional edge may be found half way to the slope of the grey level profile represented in the top right plot.

Figure 9.4 illustrates that, with an order of magnitude more noise, it becomes impossible to detect the actual edges of the different regions using the previously identified threshold, nor with a new set of thresholds, given that the Laplacian, at the actual edge location of the region of interest, is of equivalent or smaller magnitude than that due to noise.

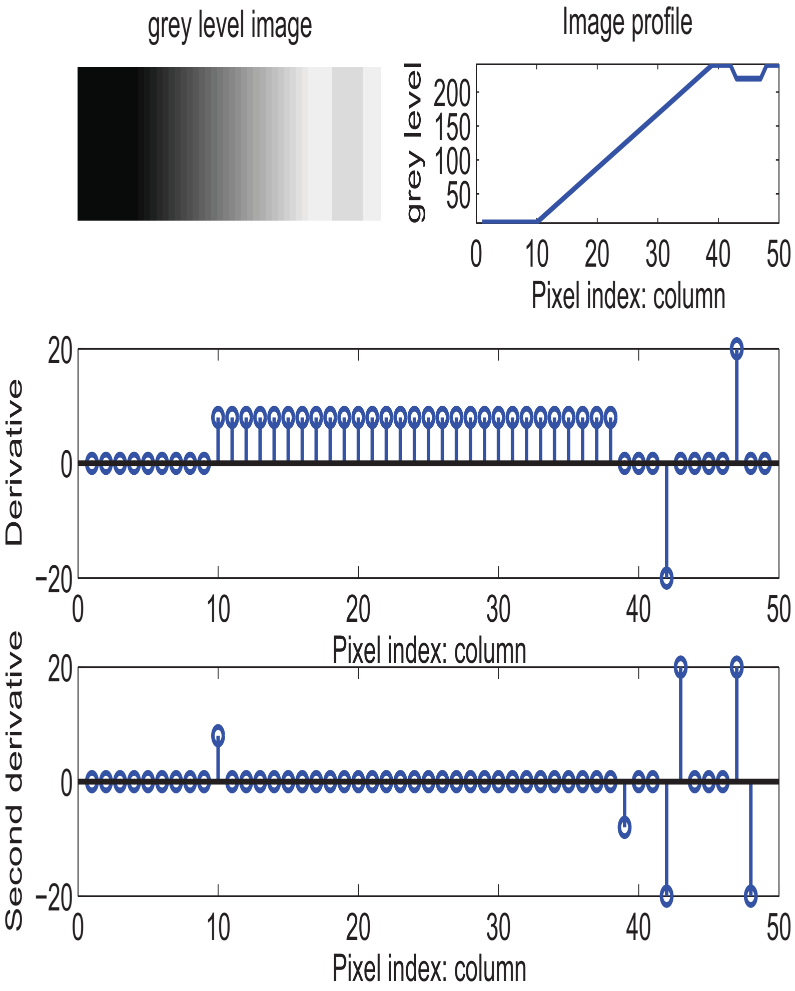


Figure 9.2: Illustrating the output of edge detectors based on the first as well as the second derivative.

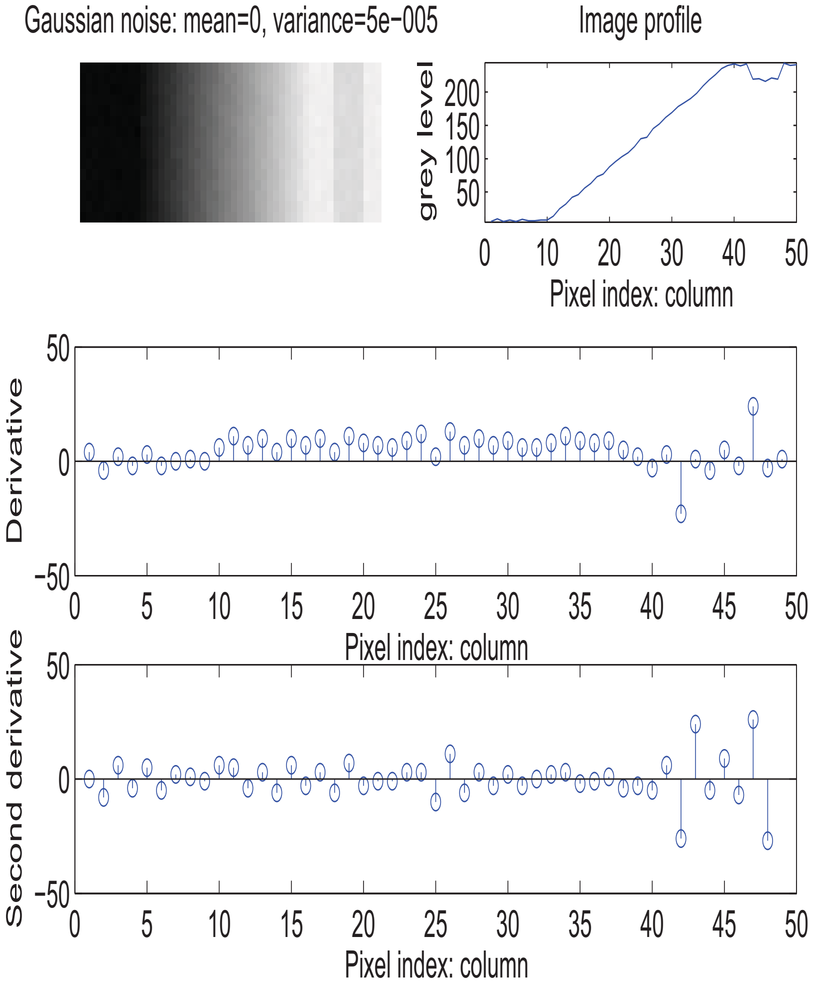


Figure 9.3: Small amount of Gaussian noise on the grey level profile affecting an image and the corresponding first and second derivative of the image.

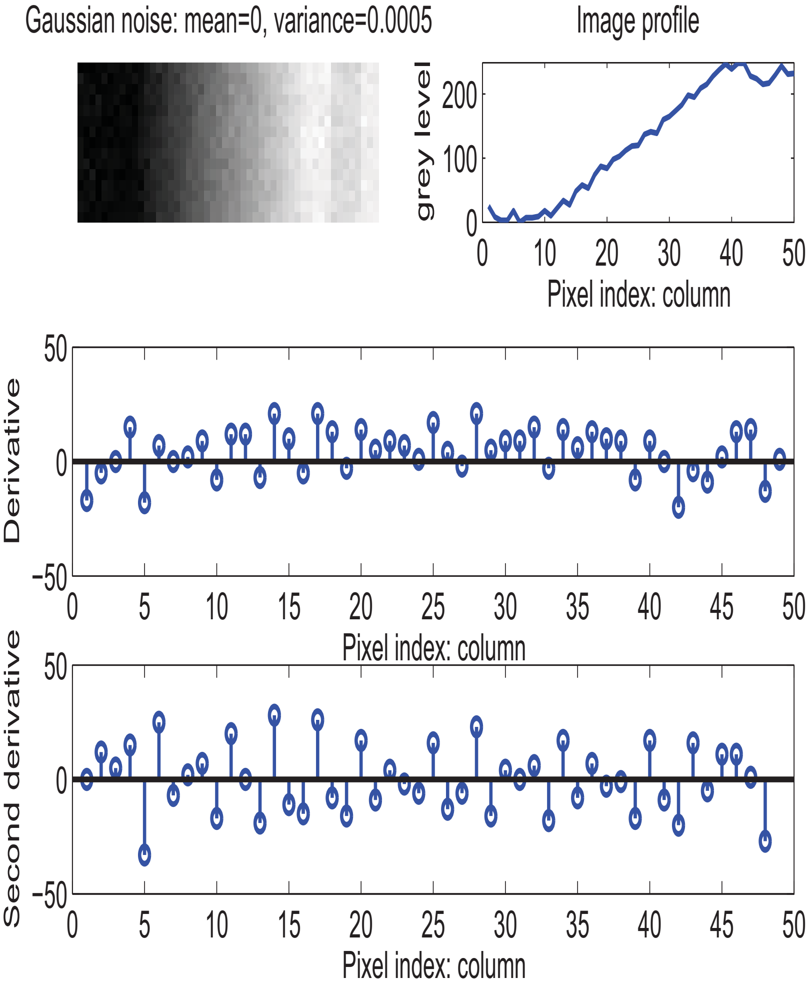


Figure 9.4: The inappropriateness of first and second derivative filters to detect the edge in a noisy image.

9.3.3 Gradient operators and masks

Roberts operator

The Roberts operator is more useful with binary images. It utilises the sum of the magnitude of the difference of the diagonal neighbours.

w_1	w_2	w_3
w_4	-1	0
w_7	0	1

w_1	w_2	w_3
w_4	0	-1
w_7	1	0

$$\boxed{G_x = z_9 - z_5} \qquad \boxed{G_y = z_8 - z_6}$$

The Roberts method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of **I** is maximum. The edge magnitude is calculated using:

$$|\mathbf{A}(r,c) - \mathbf{A}(r-1,c-1)| + |\mathbf{A}(r,c-1) - \mathbf{A}(r-1,c)|$$

instead of the original formulation which is:

$$\sqrt{[\mathbf{A}(r,c) - \mathbf{A}(r-1,c-1)]^2 + [\mathbf{A}(r,c-1) - \mathbf{A}(r-1,c)]^2}\,.$$

Suitable MATLAB® commands are:

```
BW = edge(I,'roberts')
BW = edge(I,'roberts',thresh)
[BW,thresh] = edge(I,'roberts',...)
```

Prewitt operator

The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of **I** is maximum. This method uses

$$\frac{\partial f}{\partial x}(x,y) \approx f(x+1,y) - f(x,y)$$

in both the vertical and the horizontal directions.

Row mask:

-1	-1	-1
0	0	0
1	1	1

Column mask:

-1	0	1
-1	0	1
-1	0	1

$$\boxed{G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)} \quad \boxed{G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)}$$

Convolve the row mask by the image to obtain s_1 and convolve the row mask by the image to obtain s_2. Then calculate the edge magnitude using $\sqrt{s_1^2 + s_2^2}$ and the edge direction, which is perpendicular to the edge itself, using $\tan^{-1}(s_1/s_2)$. Some appropriate MATLAB® commands are:

```
BW = edge(I,'prewitt')
BW = edge(I,'prewitt',thresh)
BW = edge(I,'prewitt',thresh,direction)
[BW,thresh] = edge(I,'prewitt',...)
```

Sobel operator

The Sobel operator provides some smoothing. The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of **I** is maximum.

Row mask:

-1	-2	-1
0	0	0
1	2	1

Column mask:

-1	0	1
-2	0	2
-1	0	1

$$\boxed{G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)} \boxed{G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)}$$

As for the Prewitt operator, convolve the row mask by the image to obtain s_1 and convolve the row mask by the image to obtain s_2. Then calculate the edge magnitude using $\sqrt{s_1^2 + s_2^2}$ and the edge direction using $\tan^{-1}(s_1/s_2)$. Appropriate MATLAB® commands are:

```
BW = edge(I,'sobel')
BW = edge(I,'sobel',thresh)
BW = edge(I,'sobel',thresh,direction)
[BW,thresh] = edge(I,'sobel',...)
```

Kirsch compass mask

Here, only one mask is used and it is rotated to the eight major compass directions:

Vertical edge referred to as 'North' and 'South';
Diagonal edge referred to as 'Northwest', 'Southeast', 'Southwest' and 'Northeast';
Horizontal edge referred to as 'West' and 'East'.

$$\begin{array}{cccc}
\text{North} & \text{Northwest} & \text{West} & \text{Southwest} \\
\uparrow & \nwarrow & \leftarrow & \swarrow \\
k_0 & k_1 & k_2 & k_3 \\
\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} &
\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} &
\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} &
\begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \\
\hdashline
\text{South} & \text{Southeast} & \text{East} & \text{Northeast} \\
\downarrow & \searrow & \rightarrow & \nearrow \\
k_4 & k_5 & k_6 & k_7 \\
\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} &
\begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} &
\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} &
\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}
\end{array}$$

The *edge magnitude* is the maximum value of the convolution of each of the masks with the image. The *edge direction* is defined by the mask that produces the maximum magnitude.

Robonson compass mask

$$\begin{array}{cccc} r_0 & r_1 & r_2 & r_3 \\ \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} & \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix} \end{array}$$

$$\begin{array}{cccc} r_4 & r_5 & r_6 & r_7 \\ \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} & \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} & \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} & \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \end{array}$$

This is a similar principle to the Kirsch compass mask, but simpler to implement as one only needs to convolve the first 4 masks and negate the results from the first four for the other four.

The edge magnitude is the maximum value of the convolution of each of the masks with the image.
The edge direction is defined by the mask that produces the maximum magnitude.

Frei-Chen masks

Frei-Chen masks form a complete set of basis vectors. As such any 3 × 3 subimage can be represent by a weighted sum of the nine Frei-Chen masks.

$$\begin{array}{cccc} f_1 & f_2 & f_3 & f_4 \\ \frac{1}{2\sqrt{2}}\begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} & \frac{1}{2\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix} & \frac{1}{2\sqrt{2}}\begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix} & \frac{1}{2\sqrt{2}}\begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix} \end{array}$$

$$\begin{array}{cccc} f_5 & f_6 & f_7 & f_8 \\ \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} & \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix} & \frac{1}{6}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} & \frac{1}{6}\begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix} \end{array}$$

$$\begin{array}{c} f_9 \\ \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \end{array}$$

The first four masks (f_1 - f_4) comprise the edge subspace. The second four masks (f_5 - f_8) comprise the line subspace. The last mask (f_9) is the average subspace.

To use the Frei-Chen masks for edge detection, select the subspace of interest (i.e. f_1 to f_9) and find the relative projection of the image onto that particular subspace using the following formulae:

$$\cos(\theta) = \sqrt{\frac{M}{S}}, \qquad M = \sum_{k \in \{e\}} (I_s, f_k)^2 = \sum_{k=1}^{9} (I_s, f_k)^2 ,$$

where e consist of the set of masks of interest, (I_s, f_k) refers to the process of overlaying the mask on the sub-image, multiplying the coincident terms and summing the results (for more details see [Umbaugh, 2010]).

Canny operator

The Canny method finds edges by looking for local maxima of the gradient of **I**. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. Therefore, this method is less likely than the other methods to be affected by noise, and more likely to detect true weak edges. Some appropriate MATLAB® commands are:

```
BW = edge(I,'canny')
BW = edge(I,'canny',thresh)
BW = edge(I,'canny',thresh,sigma)
[BW,threshold] = edge(I,'canny',...)
```

Zero-crossing method

The zero-crossing method finds edges by looking for zero crossings after filtering **I** with a specified filter. Some MATLAB® commands are:

```
BW = edge(I,'zerocross',thresh,h)
[BW,threshold] = edge(I,'zerocross',...)
```

Laplacian of Gaussian

The Laplacian of Gaussian method finds edges by looking for zero crossings after filtering **I** with a Laplacian of the Gaussian filter. This filter is defined by

$$\nabla^2 h(r) \overset{\text{def}}{=} -\left[\frac{r^2-\sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}},$$

where $h(r) = -e^{-\frac{r^2}{2\sigma^2}}$, $r^2 = x^2 + y^2$ and σ is the standard deviation with x and y the coordinates of the pixel intensity values $\mathbf{I}(x, y)$.
The filter can be implemented using the MATLAB® commands:

```
BW = edge(I,'log')
BW = edge(I,'log',thresh)
BW = edge(I,'log',thresh,sigma)
[BW,threshold] = edge(I,'log',...)
```

9.4 Noise in imaging

Noise can arise in imaging during the acquisition, the transmission or the conversion to different image formats.

As an example, consider obtaining images of a robot football system. It is clear that the camera needs to warm up to provide a stable output and the amount of light in the room is also very important. To ensure that the robots are recognised, artificial lighting is used. However, if the room used for the demonstration has any window, the daylight will affect the colour seen by the charge-coupled-device (CCD) sensor.

Noise can be independent of image coordinates if the source of the noise is in the process, or can depend on the coordinates. For example, different parts of the robot footballer could be exposed to different light conditions. These lighting conditions may result in a different type or level of noise.

Images can also be corrupted during transmission from the video input to the processing computer. To minimise such noise, coaxial cables can be used to transmit the images to the personal computer, which is used to control the robot footballers.

Noise models can be defined in the spatial domain, which are described fully in [Gonzalez et al., 2004] (see Chapter 5, Table 5.1), or in the frequency domain using various Fourier properties of the noise.

9.5 Adding noise to images using MATLAB®

MATLAB® can generate noise in the image using the function `imnoise`, with a command of the form:

```
NoisyImage=imnoise(OriginalImage,TypeOfNoise,NoiseParameters)
```

Some particular examples are:

`J = imnoise(I,'gaussian',m,v)` adds Gaussian white noise of mean m and variance v to the image **I**. The default is zero mean noise with 0.01 variance.

`J = imnoise(I,'localvar',V)` adds zero-mean, Gaussian white noise of local variance V to the image **I**. Here, V is an array of the same size as **I**.

`J = imnoise(I,'localvar',image_intensity,var)` adds zero-mean, Gaussian noise to an image **I**, where the local variance of the noise, var, is a function of the image intensity values in **I**. The `image_intensity` and the `var` arguments are vectors of the same size, and the command `plot(image_intensity,var)` plots the functional relationship between noise variance and image intensity. The `image_intensity` vector must contain normalized intensity values ranging from 0 to 1.

`J = imnoise(I,'poisson')` generates Poisson noise from the data instead of adding artificial noise to the data. In order to respect Poisson statistics, the intensities of unit8 and uint16 images must correspond to the number of photons (or any other quanta of information). Double-precision images are used when the number of photons per pixel can be much larger than 65535 (but less than 10^{12}); the intensity values vary between 0 and 1 and correspond to the number of photons divided by 10^{12}.

`J = imnoise(I,'salt & pepper',d)` adds 'salt and pepper' noise, also referred to as bipolar impulse noise, to the image **I**, where d is the noise density. This affects approximately `d*prod(size(I))` pixels. The default is 0.05 noise density.

`J = imnoise(I,'speckle',v)` adds multiplicative noise to the image **I**, using the equation $\mathbf{J} = \mathbf{I} + n \times \mathbf{I}$, where n is uniformly distributed random noise with mean 0 and variance v. The default for v is 0.04.

Remark 9.3 *The mean and variance parameters for 'gaussian', 'localvar', and 'speckle' noise types are always specified as if the image were of class double in the range* $[0, 1]$*. If the input image is of class uint8 or uint16, the* `imnoise` *function converts the image to double, adds noise according to the specified type and parameters, and then converts the noisy image back to the same class as the input.*

Figure 9.5 shows the effect of adding noise to an image, using the MATLAB® command `imnoise` and various parameter options, and Figure 9.6 illustrates the effect of noise on the resulting image histogram. It can be observed that the original image contained only three distinct grey levels. The distribution of pixels around the three initial grey levels is characteristic of the distribution of the noise model used. To facilitate the visualisation of the result a semi-log scale was used to display the histogram.

9.6 Summary

Images can be corrupted by noise. In order to minimise the effect of noise on the image, it is useful to determine, approximately, the type of noise that is present. Spatial noise can be described by random numbers characterised by a probability density function (PDF) or by a cumulative distribution function (CDF). Assuming that a CDF is given by $F_z(w)$, where w is a distributed random variable in the interval $(0, 1)$, then the random variable z can be calculated

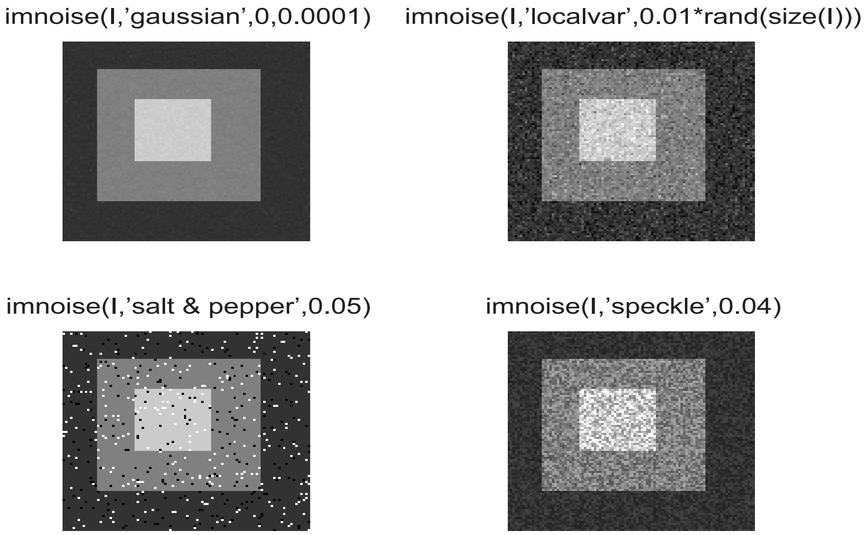

imnoise(I,'gaussian',0,0.0001)
imnoise(I,'localvar',0.01*rand(size(I)))
imnoise(I,'salt & pepper',0.05)
imnoise(I,'speckle',0.04)

Figure 9.5: Images illustrating the use of the MATLAB$^{®}$ command `imnoise` to add noise to images.

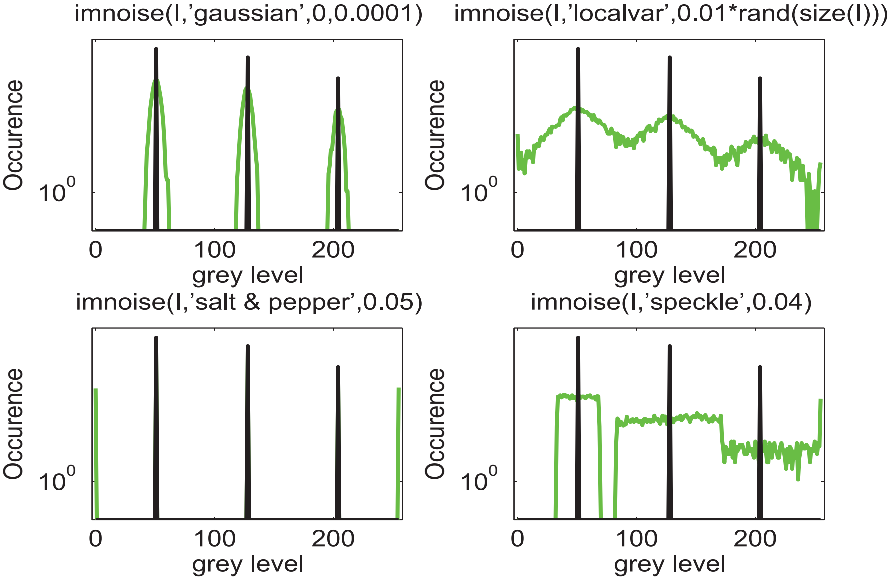


Figure 9.6: Illustrating the effect of noise on image histograms.

from its CDF by solving the equation: $z = F_z^{-1}(w)$, that is find the solution to $F_z(z) = w$.

The MATLAB® command

```
NoisyImage=imnoise(OriginalImage,TypeOfNoise,NoiseParameters)
```

can add various type of noise to an image, including: gaussian, uniform, and salt & pepper. The histogram of a noisy image can help identify the type of noise in the image.

9.7 Removing noise

Certain filters and mathematical morphology can help in removing some noise, see Section 9.1.

- Linear filter, such as Gaussian or average filters.
- Nonlinear filter, such as a Median filter.
- Adaptive filter,where the filter adapts to the statistical information present in the part of the image considered, in order to calculate its output.
- The `wiener2` function applies a Wiener filter (a certain type of linear filter) to an image adaptively, tailoring itself to the local image variance. If the variance is large, `wiener2` performs little smoothing. If the variance is small, `wiener2` performs more smoothing. The filter command `wiener2` performs best when the noise is additive noise with constant-power (for example, a 'white' process), such as Gaussian noise.
- Mathematical morphology can be performed, using the command `imopen`, which removes a small particle or noise pixel. The radius of the structuring element indicates the size of the particle/noise to remove. (Note that this topic is not discussed until the next chapter.)

Exercise 9.11 Using MATLAB®, read the image: ckt-board-orig.tif, located at:
W:\ec\student\m19mse\SampleImages\GW2004\dipum_images_ch03\
and add Gaussian noise with zero mean and a standard deviation of 0.1 (recall that the variance is the square of the standard deviation). Describe the effect of Gaussian noise. Explain what happens when you increase and decrease the variance.

Exercise 9.12 Using MATLAB®, read the image: ckt-board-orig.tif, located at:
W:\ec\student\m19mse\SampleImages\GW2004\dipum_images_ch03\
and add uniform noise with variance of 0.05. Describe the effect of uniform noise. Explain what happens when you increase and decrease the variance.

Exercise 9.13 Using MATLAB®, read the image: ckt-board-orig.tif, located at:
W:\ec\student\m19mse\SampleImages\GW2004\dipum_images_ch03\
and add add bipolar impulse noise with a noise density of 0.07. Describe the effect of bipolar impulse noise. Explain what happens when you increase and decrease the noise density.

Illustrative example

An image representing three regions of interest of distinct grey level has been corrupted with Gaussian noise and, additionally, binomial impulse noise (Salt and Pepper). Figure 9.7 shows the original image, as well as the noise corrupted image together with a comparison of their histograms. The Gaussian noise results in Gaussian distributions centred around the original grey level of each three regions of interest. The binomial impulse noise is characterised by the black and white pixel on the noisy image and by two sets of pixels at 0 and 255.

Figure 9.8 illustrates the application of four different filters to the noisy image. It can be clearly seen that filters, that do not contain any smoothing combined with the edge detection, exhibit false positives. In particular, the binomial impulse noise, which creates pixel grey level significantly different from its neighbours, results in high gradient change and, hence, high magnitude in the resulting filtered image. It is possible to reduce the number of edges created by the Gaussian noise, within each region of interest, however, this also leads to open contours of the regions of interest, with a few pixels missing. Closed contours can be obtained, but at the cost of a large number of false positives. Filters based on the Laplacian operator or Gaussian filter and the Canny filter produce better contours. The Canny filter is the only one able to produce a closed contour as well as no false positive within the regions of interest. This is due to its ability to make use of weak edge responses to link two strong edges together.

Figure 9.9 clearly demonstrate the benefits of making use low pass filtering prior to applying an edge detector. All the edge detectors applied produce

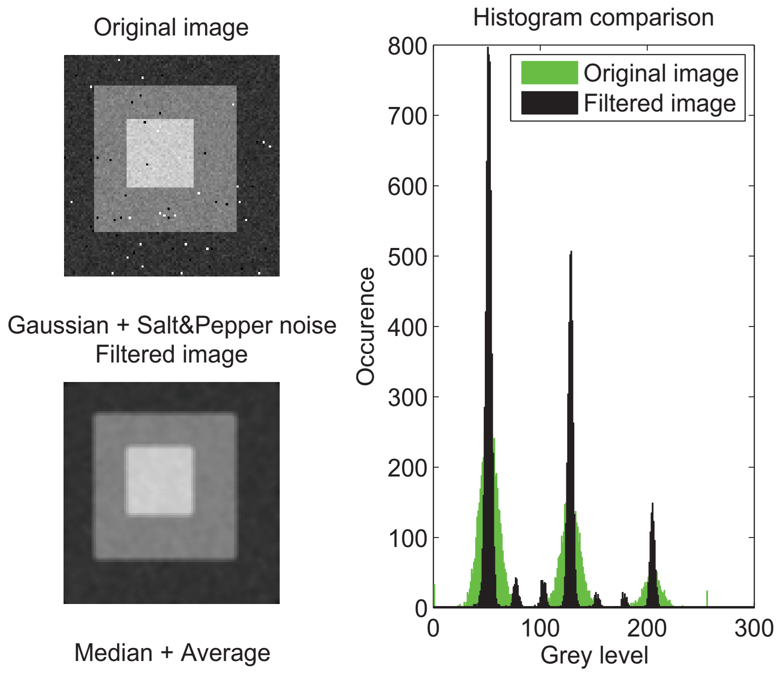
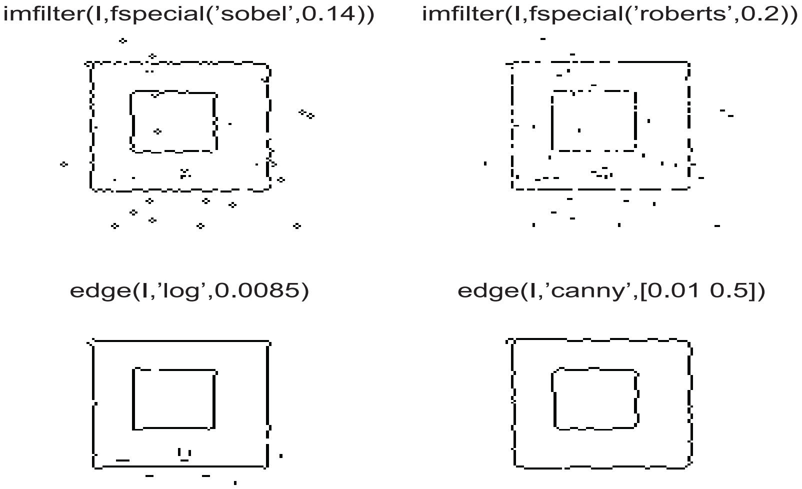


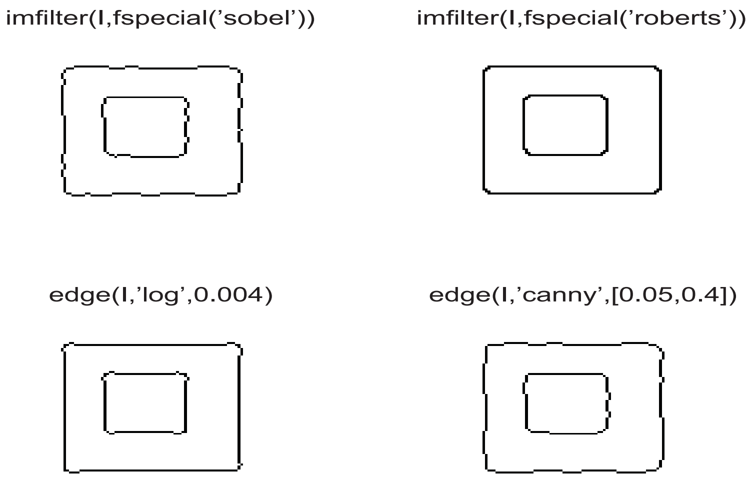
Figure 9.7: Illustrating the effect of noise on images and their histogram.



imfilter(I,fspecial('sobel',0.14))
imfilter(I,fspecial('roberts',0.2))
edge(I,'log',0.0085)
edge(I,'canny',[0.01 0.5])

Figure 9.8: Illustrating the trade off between obtaining a large number of false positives and not being able to obtain a closed contour of the regions of interest.

closed contour. The edges are rounded due to the application of the average filter which reduces the pixel intensity at the edges of each square region. In this particular case, the best combination is to apply a median filter, followed by an average filter, prior to using the Roberts edge detector, which emphasises horizontal and vertical edges.



imfilter(I,fspecial('sobel'))

imfilter(I,fspecial('roberts'))

edge(I,'log',0.004)

edge(I,'canny',[0.05,0.4])

Figure 9.9: Illustrating the benefit of applying low pass filtering to obtained closed contours of regions of interest.

Chapter 10

Mathematical morphology and set notations

Morphology is a broad set of image processing operations that process images based on shapes. A structuring element is applied, through morphological operations to an input image, which creates an output image of the same size. Mathematical morphology is based on set theory. Using set theory terminology, a digital image can be represented by a function $f(x, y)$, where $(x, y) \in \mathbb{Z} \times \mathbb{Z}$ and f is a mapping that assigns a value (namely an integer) to each pair of distinct coordinates (x, y).

The following notation will be utilised in the subsequent work.

A digital image can be represented by a set $A \subset \mathbb{Z} \times \mathbb{Z}$ and $w = (x, y) \in \mathbb{Z} \times \mathbb{Z}$ is an element of that digital image. If a pixel belongs to the image A, then $w \in A$. If a pixel does not belongs to an image, then $w \notin A$. If a set A of pixels satisfies a specified condition, then the set A is expressed as $A = \{w | \text{condition}\}$. If the set A is empty, then this is denoted by $A = \emptyset$.

For example, using the above notation, the reflection of a set A can be expressed, formally, as $\hat{A} = \{w | w = -b, \quad \text{for } b \in A\}$ and the translation of a set A by $z = (z_1, z_2)$ is given by $A_z = \{w | w = a + z, \quad \text{for } a \in A\}$. Other important notations concerning intersection, union, complement and difference of sets (see

Table 10.1) are defined by:

$$\begin{aligned} \text{intersection:} \quad & A \cap B \overset{\text{def}}{=} \{w|w \in A \text{ and } w \in B\} \\ \text{union:} \quad & A \cup B \overset{\text{def}}{=} \{w|w \in A \text{ or } w \in B\} \\ \text{complement:} \quad & A^c \overset{\text{def}}{=} \{w|w \notin A\} \\ \text{difference:} \quad & A - B \overset{\text{def}}{=} A \cap B^c \end{aligned}$$

and are illustrated graphically in Figure 10.1.

Set operation	Description	MATLAB® expressions for binary images	Name
$A \cap B$	Intersection	`A & B`	AND
$A \cup B$	Union	`A|B`	OR
A^c	Complement	`~A`	NOT
$A - B$	Difference	`A & ~B`	DIFFERENCE

Table 10.1: Set operations and associated MATLAB® expressions.

Exercise 10.1 Create two matrices filled with 1 or 0 such that, in one matrix, the elements 1 form a square and, in the second matrix, the elements 1 form a triangle, with the triangle having a point common with the square. Determine the complement, the union, the intersection, the difference, the reflection and the translation by $z = (3, 5)$ of the square and the triangle.

Exercise 10.2 Perform the OR, AND and NOT logical operations with the square and the triangle, introduced in Exercise 10.1.

10.1 Fundamental morphological operators

morphological operators Morphological image processing is based on set theory and the operations of dilation and erosion. A thorough description of these topics can be found in [Gonzalez et al., 2004].

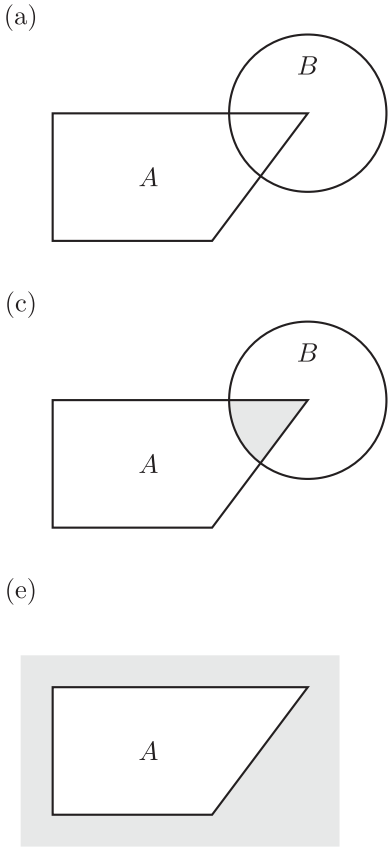


Figure 10.1: (a) Two sets A and B. (b) The union of A and B. (c) The intersection of A and B. (d) The difference: $A - B$. (e) The complement of A.

10.1.1 The dilation operator, $\oplus$, for bridging gaps

The dilation operation, denoted by $\oplus$, grows or thickens objects of a binary image. The definition of dilation (see [Gonzalez et al., 2004]) is:

$$A \oplus B \overset{\text{def}}{=} \{z| \left(\hat{B}\right)_z \cap A \neq \emptyset\},$$

where $\left(\hat{B}\right)_z$ denotes the reflection of all elements of B, about the origin, translating the origin of $\hat{B}$ to the point z.

This definition means that the dilation of A, by the structuring element B, is the set consisting of all the structuring element origin location (?), where the reflected and translated B overlaps at least some portion of A. This means that an element 1 appears in the dilated image at the origin of the structuring element whenever at least one pixel of the original image is covered by the structuring element. Using MATLAB®, the dilation process produces an output pixel whose value is the maximum value of all the pixels in the neighbourhood of the input pixel. Pixels beyond the image border are assigned the minimum value afforded by the data type, for example, when using binary or grey scale, it gives a zero value.

Some properties of dilation are the following.

(D1) Commutativity: $A \oplus B = B \oplus A$. Note that, in image processing, the image is conventionally the first operand of $A \oplus B$ and the structuring element the second.

(D2) Associativity: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$. This property can be used to advantage if a structuring element B can be decomposed into two structuring elements B_1 and B_2. Then dilating A with B is the same as dilating A with B_1 and dilating the result with B_2.

(D3) Dilation can be expressed as a union of shifted point sets:

$$A \oplus B = \bigcup_{b \in B} A_b.$$

(D4) Dilation is invariant to translation: $(A)_z \oplus B = (A \oplus B)_z$.

In MATLAB®, dilation can be performed using a command with the following structure:

```
TransformedImage = imdilate(ImageObject,StructuringElement)
```

To assess the effect of dilation, the following MATLAB® routine can be used:

```
% Set dilated image output to grey (half the max grey level)
% Set elements corresponding to 1 in original image to white
%
Idil = imdilate(ImageObject,StructuringElement);
Index = find(ImageObject==max(max(ImageObject)));
Index1 = find(Idil==max(max(Idil)));
Ioutput=zeros(size(ImageObject));
Ioutput(Index1) = double(max(max(Idil)))/2;
Ioutput (Index)= double(max(max(ImageObject)));
```

Dilation can be used to bridge gaps, as shown in Figure 10.2.

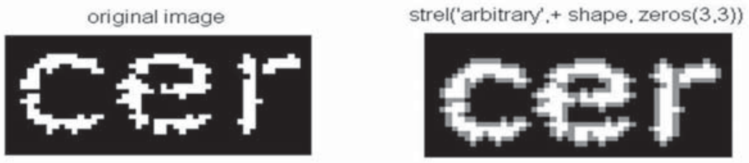


Figure 10.2: Illustrating the ability of bridging gaps in letters using a structuring element `[0 1 0; 1 1 1; 0 1 0]`.

Remark 10.1 *The grey pixels have been obtained by subtracting the dilated image from the original image, resulting in the edge of the pattern being drawn.*

10.1.2 The erosion operator, $\ominus$, for eliminating irrelevant details

The erosion operation, denoted by $\ominus$, shrinks or thins objects in binary images. The manner and extent of the 'shrinking' is controlled by the structuring element. The definition of erosion (see [Gonzalez et al., 2004]) is:

$$A \ominus B \overset{\text{def}}{=} \{z|\,(B)_z \subseteq A.$$

This definition means that the erosion of A by the structuring element B is the set of all points z such that B, translated by z, is contained in A, that is it is a subset of A. This means that a 1 appears in the eroded image, at the place of the origin (or centre) of the structuring element, whenever the whole structuring element fits into A. Using MATLAB$^{\circledR}$, the erosion process produces an output pixel whose value is the minimum value of all the pixels in the neighbourhood of the input pixel. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0. Pixels beyond the image border are assigned the maximum value afforded by the data type.

Some properties of erosion are the following.

(E1) Erosion and dilation are dual of each other with respect to complement and relfection: $(A \ominus B)^c = A^c \oplus \hat{B}$.

(E2) Erosion is translation invariant.

In MATLAB$^{\circledR}$, dilation can be performed using a command with the following structure:

```
TransformedImage = imerode(ImageObject,StructuringElement)
```

The effect of erosion can be assessed using the following MATLAB$^{\circledR}$ routine:

```
% This code set the original pixels in grey
% and the remaining pixels in the eroded image in white
%
Ierode = imerode(ImageObject,StructuringElement) ;
Index1 = find(ImageObject==max(max(ImageObject)));
Index = find(Ierode==max(max(Ierode)));
Ioutput = zeros(size(ImageObject));
Ioutput(Index1) = double(max(max(ImageObject)))/2;
Ioutput(Index) = double(max(max(ImageObject)));
```

The effects of erosion can be seen in Figure 10.3.

Remark 10.2 *There are other methods to display the pixels affected by the morphological operations. For example, the MATLAB$^{\circledR}$ command:*

```
Isub=imsubtract(Image, Ierode);imshow(Isub)
```

will show only the contours.

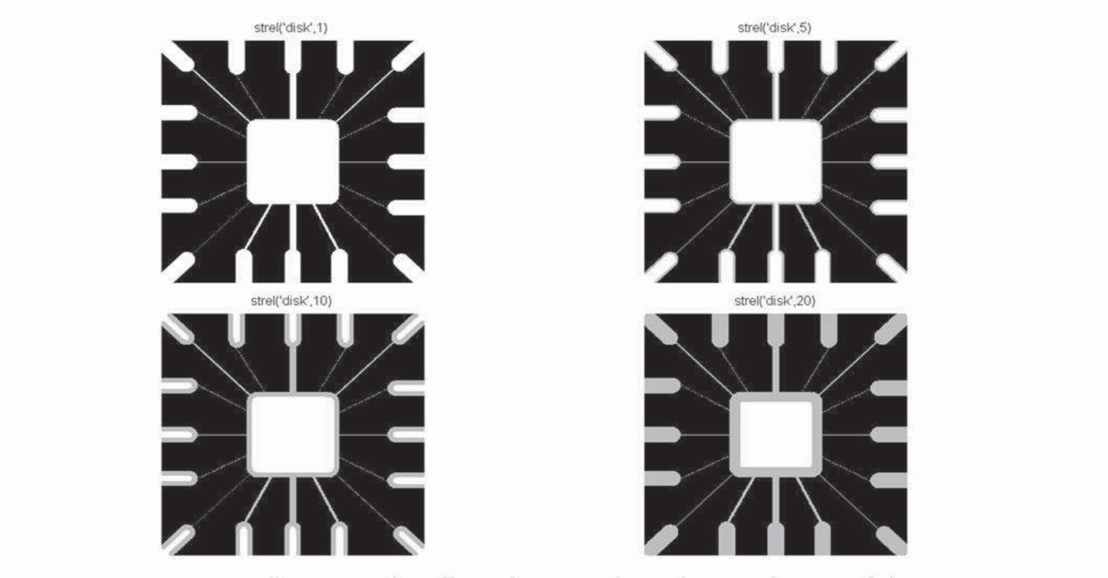


Figure 10.3: Illustrating the effect of erosion depending on the size of the structuring element.

10.2 Structuring elements

The structuring element consists of a pattern specified by the coordinates of a number of discrete points relative to some origin. Normally Cartesian coordinates are used and so a convenient way of representing the element is as a small image on a rectangular grid.

When a morphological operation is carried out, the origin of the structuring element is typically translated to each pixel position in the image in turn, and then the points within the translated structuring element are compared with the underlying image pixel values. The details of this comparison, and the effect of the outcome depend on which morphological operator is being used
(see http://homepages.inf.ed.ac.uk/rbf/HIPR2/strctel.htm).

An important point to note is that although a rectangular grid is used to represent the structuring element, not every point in that grid is part of the structuring element in general. Hence, the elements may contain some blanks. In many texts, these blanks are represented as zeros (which is not strictly correct).

10.2.1 Constructing a structuring element in MATLAB®

The function `strel` is used to construct structural elements that are stored as `strel` objects. These objects are decomposed into a number of structuring elements. These can be used by some of the image processing functions to speed up the calculations. For example, the command

```
SE = strel(shape,parameters)
```

creates a structuring element, `SE`, of the type specified by shape, where each structuring element has parameters to define its formation. By default, the origin of the structuring element is at the centre of the matrix describing the structuring element. To have an origin not at the centre, it is possible to define a matrix where the non-zero elements are not centred around the centre of the matrix; for example,

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Exercise 10.3 Assess the effect of various structuring elements by applying erosion, dilation, opening and closing on binary images provided on the W drive. Comment on your results.

10.2.2 Examples of structuring elements

The MATLAB® command `SE=strel('square',W)` creates a square structuring element whose width is `W` pixels. Note that `W` must be a non-negative integer scalar (see Figure 10.4). The command `SE=strel('rectangle',MN)` creates a

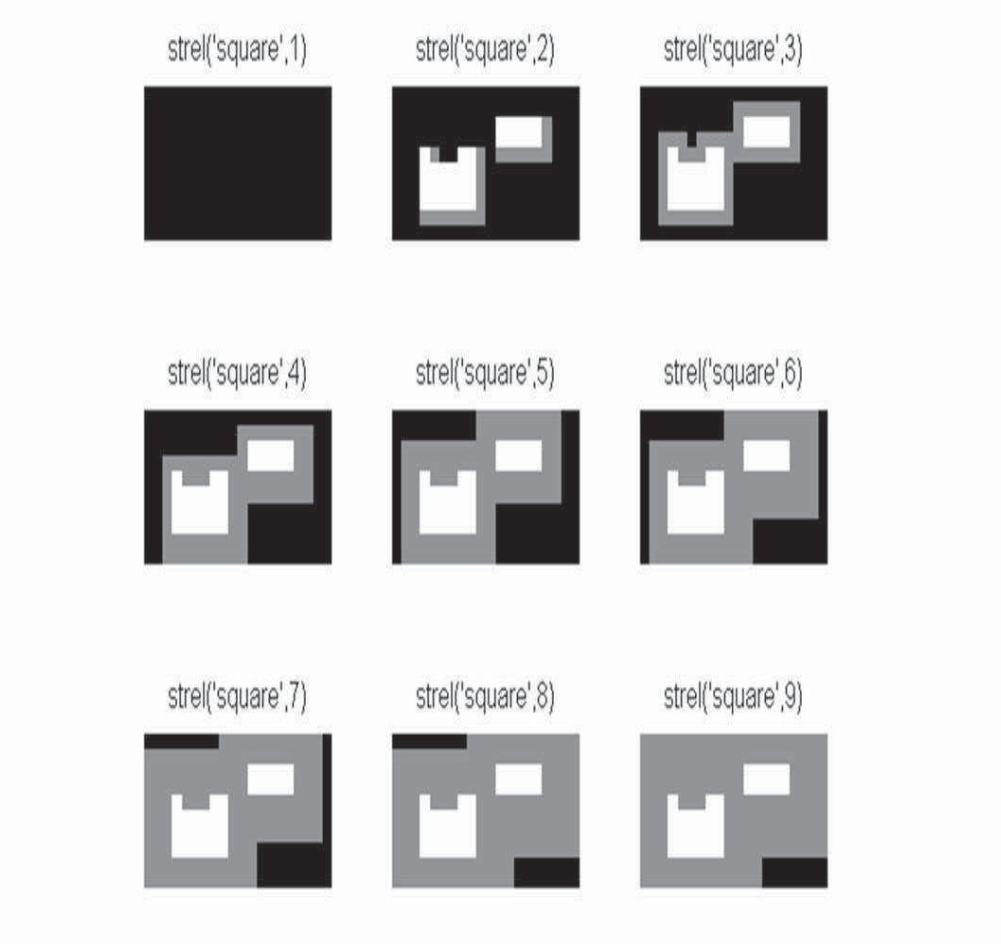


Figure 10.4: Examples using `SE = strel('square',W)`.

flat, rectangle-shaped structuring element, where `MN` specifies the size. Here `MN` must be a two-element vector of non-negative integers. The first element of `MN` is the number of rows in the structuring element neighbourhood; the second element is the number of columns (see Figure 10.5). In addition, the command

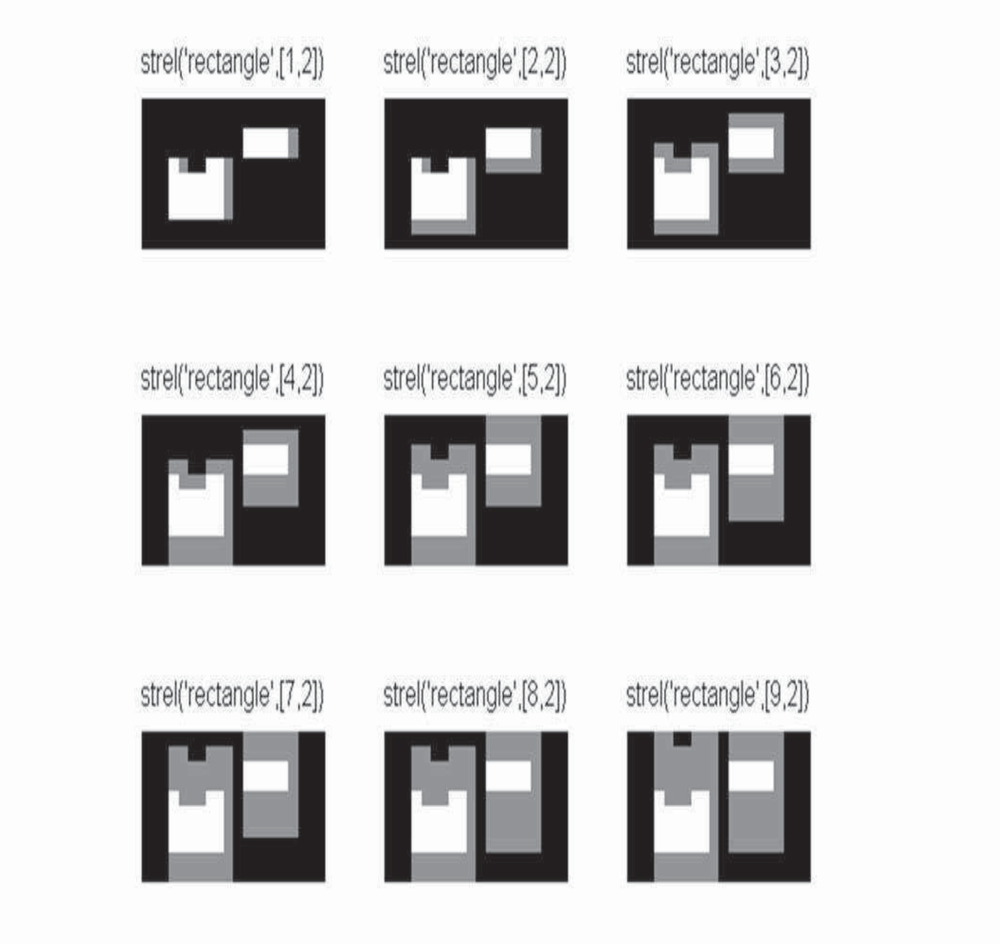


Figure 10.5: Examples using `SE = strel('rectangle',W)`.

`SE=strel('line',LEN,DEG)` creates a flat, linear structuring element, where `LEN` specifies the length, and `DEG` specifies the angle (in degrees) of the line, as measured in a counter-clockwise direction from the horizontal axis. Note that `LEN` is, approximately, the distance between the centres of the structuring element members at opposite ends of the line. The MATLAB® command `SE=strel('diamond',R)` creates a flat, diamond-shaped structuring element, where `R` specifies the distance from the structuring element origin to the points of the diamond. Note that `R` must be a non-negative integer scalar. The command `SE=strel('octagon',R)` creates a flat, octagonal structuring element, where `R` specifies the distance from the structuring element origin to the sides of the octagon, as measured along the horizontal and vertical axes. Here, `R` must be a non-negative multiple of 3 (see Figure 10.6). The MATLAB® command `SE=strel('arbitrary',NHOOD)` creates a flat structuring element where `NHOOD` specifies the neighbourhood. Here, `NHOOD` is a matrix containing 1's and 0's; the location of the 1's defines the neighbourhood for the morphological operation. The centre (or origin) of `NHOOD` is its centre element, given by `floor((size(NHOOD)+1)/2)`. One can omit the 'arbitrary' string and just use

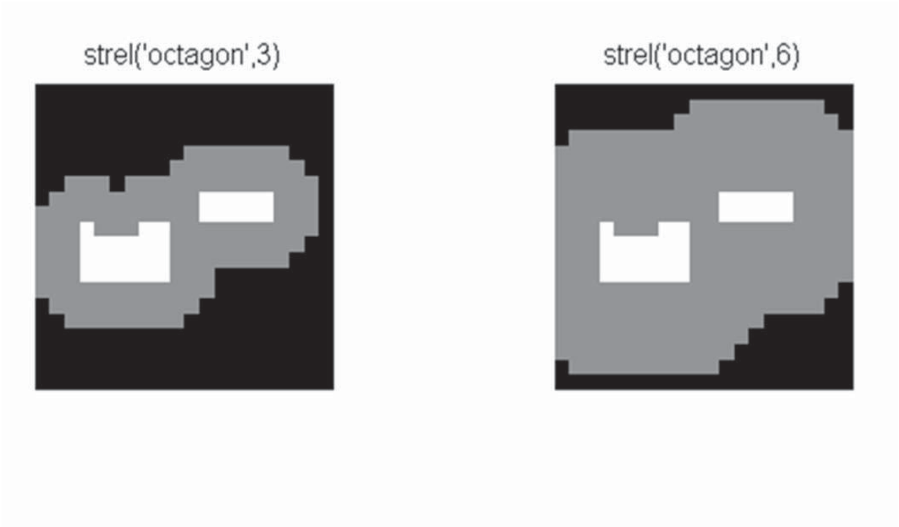


Figure 10.6: Examples using `SE = strel('octagon',W)`.

`strel(NHOOD)`.

The MATLAB® command `SE=strel('arbitrary',NHOOD,HEIGHT)` creates a non-flat structuring element, where `NHOOD` specifies the neighbourhood. The argument `HEIGHT` is a matrix the same size as `NHOOD` containing the height values associated with each non-zero element of `NHOOD`. The `HEIGHT` matrix must be real and finite-valued. One can omit the 'arbitrary' string and just use `strel(NHOOD,HEIGHT)`.

10.3 Combining dilation and erosion

Mathematical morphology uses dilation and erosion operations as a basis. New operators can be formed by combining a number of dilations/erosions and erosions/dilations with different structuring elements. This is possible because dilation and erosion are not inverse transformations, that is if an image is eroded and then dilated, then the resulting image is not the original image.

This section will consider opening, closing operations, and the hit-or-miss transformation, that is also used as a building block to derive more complex transformations.

10.3.1 Opening, denoted by $\circ$

The opening operation is an erosion by a structuring element B followed by a dilation of the result by the same structuring element:

$$A \circ B = (A \ominus B) \oplus B.$$

Opening can also be interpreted as the union of all translations of B that fit entirely within A (see [Gonzalez et al., 2004]). An appropriate MATLAB® command has the structure:

```
TransformedImage = imopen(ImageObject,StructuringElement)
```

10.3.2 Closing, denoted by $\bullet$

The closing operation is a dilation followed by an erosion, using the same structuring element. Closing can be interpreted as the complement of the union of all translations of B that do not overlap A. An appropriate MATLAB® command has the structure:

```
TransformedImage = imclose(ImageObject,StructuringElement)
```

Exercise 10.4 Write a MATLAB® function file that takes as argument the structuring element '`[0 1 0;1 1 1;0 1 0]`' and, in addtion, the original image '`ImageObject`' and returns '`SeeEffectOfDilate`' or '`SeeEffectOfErode`', '`SeeEffectOfOpen`' and '`SeeEffectOfDilate`'. Test your function with various images and comment on your results.

Exercise 10.5 Using black and white images, provided on the W drive, check that you are able to fill gaps using the closing operation. Write down the appropriate MATLAB® commands to achieve this and write down any relevant observations.

Exercise 10.6 Check that the opening operation allows you to separate structures that are linked together by a few pixels.

10.4 Hit-and-Miss transformation

The hit-or-miss operation preserves pixels whose neighbourhoods match the shape of the structuring element `SE1` and do not match the shape of the structuring element `SE2`. The MATLAB® implementation

```
BW2 = bwhitmiss(OriginalImage,SE1,SE2)
```

performs the hit-and-miss operation defined by the structuring elements `SE1` and `SE2`. Here, `SE1` and `SE2` can be flat structuring element objects, created by `strel`, or neighbourhood arrays.

Remark 10.3 *The neighbourhoods of `SE1` and `SE2` should not have any overlapping elements.*

The syntax `bwhitmiss(BW1,SE1,SE2)` is equivalent to

```
imerode(BW1,SE1);
imerode(∼BW1,SE2)
```

The MATLAB® command `BW2 = bwhitmiss(BW1,INTERVAL)` performs the hit-and-miss operation, defined in terms of a single array, called an interval. An interval is an array whose elements can contain 1, 0, or −1. The 1-valued elements make up the domain of SE1, the -1-valued elements make up the domain of SE2, and the 0-valued elements are ignored. The syntax `bwhitmiss(INTERVAL)` is equivalent to `bwhitmiss(BW1,INTERVAL == 1,INTERVAL == -1)`.

Example 10.1 Consider an example of the use of the hit-or-miss transformation to detect the centre of a '+' shape, defined by

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

for an image represented by A, where

A=

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & \textcircled{1} & 1 & \textcircled{1} & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

SE1=

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

SE2=

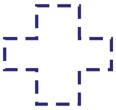
$$\begin{bmatrix} \textcircled{1} & 0 & \textcircled{1} \\ 0 & 0 & 0 \\ \textcircled{1} & 0 & \textcircled{1} \end{bmatrix}$$

If any element circled in A matches elements of the structural element SE2, then the output is zero, indicating that the shape, illustrated in SE1, does not match the pattern.

The MATLAB® command `HM1=bwhitmiss(A,SE1,SE2)` gives

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \textcircled{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

INT2= $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$



Ignoring the circled elements in `A`, which correspond to the zeros in the structural element `INT2`, the hit-or-miss transformation will find in `A` the required shape, represented by the elements in the dashed line, given in `INT2`.

The MATLAB® command `HM3=bwhitmiss(A,INT2)` gives

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \textcircled{1} & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Two '+' are detected if surrounding pixels are ignored.

`INT1=`

$$\begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & 1 \\ -1 & 1 & -1 \end{bmatrix}$$

The structural element `INT1` performs the same operation, combining `SE1` and `SE2` into a single structuring element.

The MATLAB® command `HM2=bwhitmiss(A,INT1)` gives

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

10.5 Skeleton

It has been seen that the open operation (introduced in §10.3.1) forces the shape of the template/structuring element on the convex part of an edge. The close operation (see §10.3.2) forces the shape of the template/structuring element on the concave part of an edge. Erode and open operations can be combined to form a *skeleton* operation. The skeleton operation reduces all objects in an image to lines, without changing the essential structure of the image. This can be achieved in MATLAB® using a command with the following structure:

```
SkeletonImage = bwmorph(OriginalImage,'skel',NoOfTime)
```

The command `SkeletonImage = bwmorph(OriginalImage, 'skel',Inf)`, with `NoOfTime = Inf`, removes pixels on the boundaries of objects, but does not allow objects to break apart. The remaining pixels make up the image skeleton. This option preserves the *Euler number*.

Some images illustrating the effect of skeletonisation are shown in Figure 10.7.

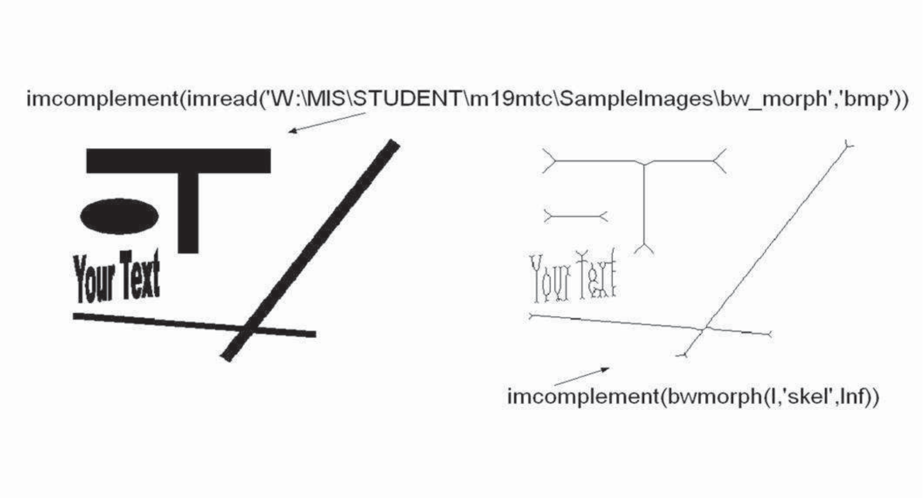

imcomplement(imread('W:\MIS\STUDENT\m19mtc\SampleImages\bw_morph','bmp'))
Your Text
Your Text
imcomplement(bwmorph(I,'skel',Inf))

Figure 10.7: Images illustrating the effect of skeletonisation, where, for display, `imcomplement` is used.

10.5.1 The Euler number determined using MATLAB®

Using MATLAB®, the `bweuler` function returns the Euler number for a binary image. The Euler number is a measure of the topology of an image. For an image containing a number of objects, the Euler number is defined as the number obtained by subtracting the number of holes in those objects from the total number of objects in the image. One can use either 4- or 8-connected neighbourhoods.

Connectivity is a criteria that describe how pixels in an image form a connected group.

Remarks 10.4 *Pixels are connected if their edges or corners touch. This means that if two adjoining pixels are ones, they are part of the same object, regardless of whether they are connected along the horizontal, vertical, or diagonal direction.*

In the example illustrated in Figure 10.7, there are nine 8-connected objects and two holes. The Euler number is calculated using the MATLAB® command: `eul = bweuler(SK,8)`, and gives the result: `eul = 7`.

10.6 Thinning

Thinning removes pixels so that an object, without holes, shrinks to a minimally connected stroke, namely a line segment, and an object with holes shrinks to a connected ring, halfway between each hole and the outer boundary. This operation can be achieved, in MATLAB®, using a command with the following structure:

```
SkeletonImage = bwmorph(OriginalImage,'thin',NoOfTime)
```

with `NoOfTime = Inf`, thins objects to lines. This option preserves the Euler number. Some images illustrating the effect of 'thinning' are shown in Figure 10.8.

10.7 Thicken

This operation thickens objects by adding pixels to the exterior of objects. Using `NoOfTime = Inf`, it adds pixels to the exterior of objects, until doing so would result in previously unconnected objects being 8-connected. This operation can be achieved in MATLAB® using a command with the following structure:

```
SkeletonImage = bwmorph(OriginalImage,'thicken',NoOfTime)
```

Some images illustrating the effect of 'thickening' are shown in Figure 10.9.

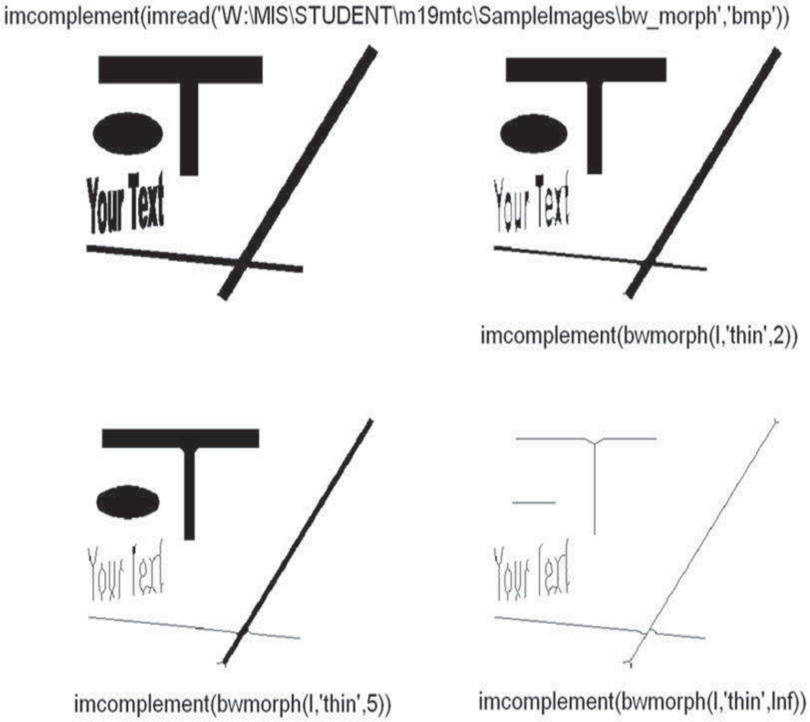

imcomplement(imread('W:\MIS\STUDENT\m19mtc\SampleImages\bw_morph','bmp'))
Your Text
Your Text
imcomplement(bwmorph(I,'thin',2))
Your Text
Your Text
imcomplement(bwmorph(I,'thin',5))
imcomplement(bwmorph(I,'thin',Inf))

Figure 10.8: Images illustrating the effect of thinning, where `imcomplement` is used for display.

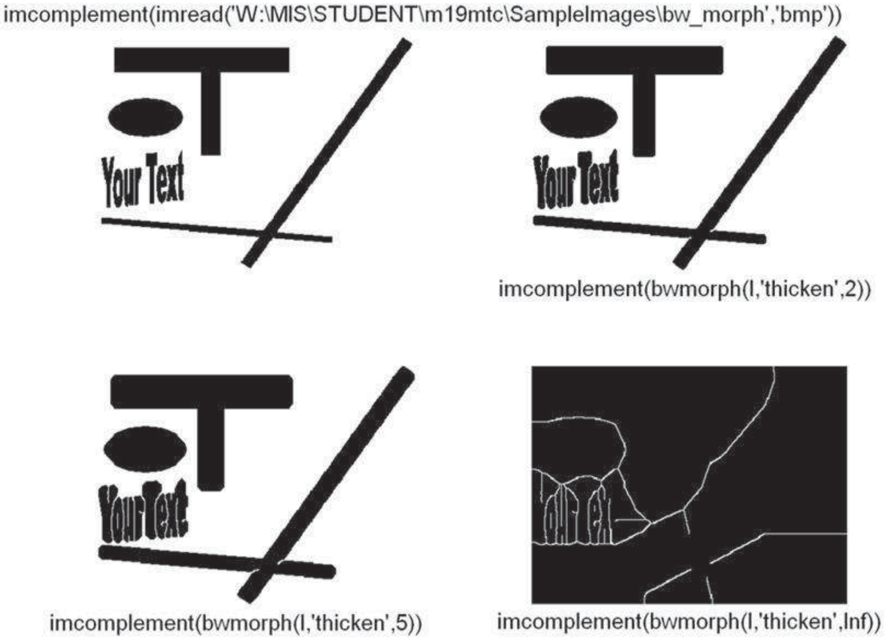

imcomplement(imread('W:\MIS\STUDENT\m19mtc\SampleImages\bw_morph','bmp'))
Your Text
Your Text
imcomplement(bwmorph(I,'thicken',2))
Your Text
imcomplement(bwmorph(I,'thicken',5))
imcomplement(bwmorph(I,'thicken',Inf))

Figure 10.9: Images illustrating the effect of thickening, where `imcomplement` is used for display.

10.8 Top hat

The MATLAB® option `'tophat'`, in the command `bwmorph(OriginalImage, 'tophat',NoOfTime)`, returns the image with the morphological opening of the image subtracted.

10.9 Bottom hat

In the MATLAB® command

```
SkeletonImage = bwmorph(OriginalImage, 'bothat',NoOfTime)
```

the option `'bothat'` (which abbreviates 'bottom hat') performs morphological closing (namely dilation followed by erosion) and subtracts the original image.

10.10 Morphological operations implemented in `bwmorph`

`'bridge'` bridges unconnected pixels, that is it sets 0-valued pixels to 1 if they have two nonzero neighbours that are not connected. For example:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{becomes} \quad \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

`'clean'` removes isolated pixels, that is individual 1's that are surrounded by 0's, such as the centre pixel in this pattern:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

`'fill'` fills isolated interior pixels, that is individual 0's that are surrounded by 1's, such as the centre pixel in this pattern:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

`'hbreak'` removes H-connected pixels. For example:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \text{becomes} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

`'majority'` sets a pixel to 1 if five or more pixels in its 3-by-3 neighbourhood are 1's; otherwise, it sets the pixel to 0.

`'remove'` removes interior pixels. This option sets a pixel to 0 if all its 4-connected neighbours are 1, thus leaving only the boundary pixels on.

`'spur'` removes spur pixels, that is pixels that are not connected. For example:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \qquad \text{becomes} \qquad \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

`'dilate'` performs dilation using the structuring element `ones(3)`.

`'erode'` performs erosion using the structuring element `ones(3)`.

`'open'` implements morphological opening (namely erosion followed by dilation).

`'close'` performs morphological closing (namely dilation followed by erosion).

Exercise 10.7 Modify the matrix **I**, representing the original image, to highlight the differences between the various operators. Justify your modifications.

Exercise 10.8 Write the MATLAB® code and justify the results obtained for the image, represented by **I**, where

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

using the command `bwmorph` with the various options:

```
BR = bwmorph(I,'bridge')  MAJ = bwmorph(I,'majority')
CLE = bwmorph(I,'clean')  RV = bwmorph(I,'remove')
CLO = bwmorph(I,'close')  SH = bwmorph(I,'shrink',inf)
OP = bwmorph(I,'open')    TK = bwmorph(I,'thicken',inf)
DG = bwmorph(I,'diag')    TN = bwmorph(I,'thin',inf)
DL = bwmorph(I,'dilate')  SK = bwmorph(I,'skel',inf)
ER = bwmorph(I,'erode')   SP = bwmorph(I,'spur',inf)
FL = bwmorph(I,'fill')    TH = bwmorph(I,'tophat',inf)
HB = bwmorph(I,'hbreak')  BH = bwmorph(I,'bothat',inf)
```

Example 10.2 Consider an image of some non-overlapping grains of rice. This example will illustrate the use of morphological operators to count the number of rice grains. The particular morphological operators used are indicated in the following MATLAB® commands and their effects are shown in Figure 10.10.

```
I = imread(’rice.png’)
figure(1);
clf
subplot(331)
subimage(I)
%
subplot(332)
background = imopen(I,strel(’disk’,5));
subimage(background)
background1 = imfilter(background ,fspecial(’gaussian’,10));
%
subplot(333)
surf(double(background(1:8:end,1:8:end))),zlim([0 255]);
set(gca,’ydir’,’reverse’);
I2 = imsubtract(I,background);
%
subplot(334)
subimage(I2)
%
subplot(335)
I3 = imadjust(I2);
imshow(I3);
%
subplot(336)
level = graythresh(I3);
bw = im2bw(I3,level);
subimage(bw)
%
subplot(337)
[labeled,numObjects] = bwlabel(bw,4);
subimage(labeled);
%
subplot(338)
pseudo_color= label2rgb(labeled,@spring,’c’,’shuffle’);
subimage(pseudo_color);
%
subplot(339)
pindata=regionprops(labeled,’basic’);
max([pindata.Area]);
hist([pindata.Area],20)
```

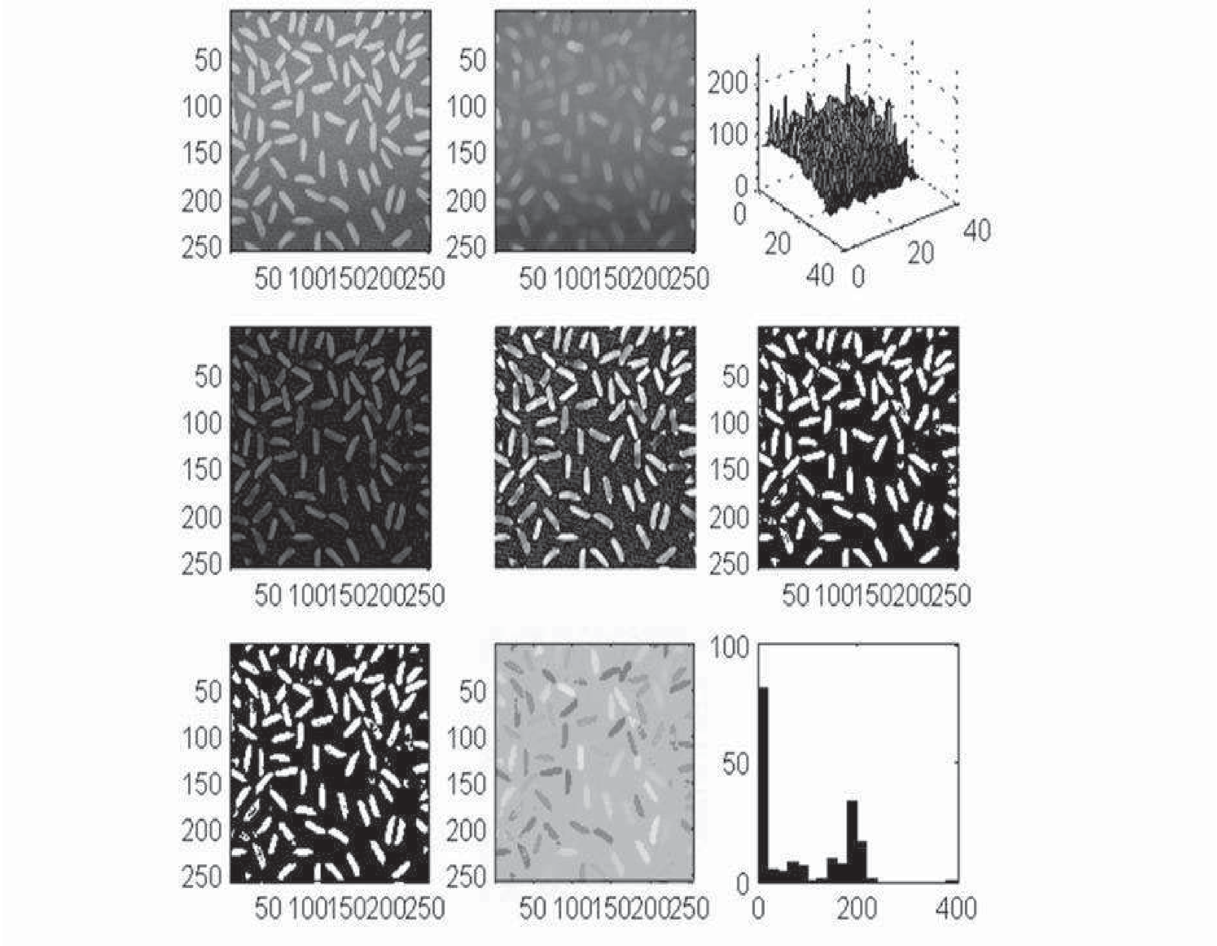


Figure 10.10: Images illustrating the use of morphological operators to count grains of rice.

Appendix A

Summation formulae

Finite summations

(i) $$\sum_{r=1}^{n} r = \frac{n(n+1)}{2}$$

(ii) $$\sum_{r=1}^{n} r^2 = \frac{n(n+1)(2n+1)}{6}$$

(iii) $$\sum_{r=0}^{n} a^r = \frac{1-a^{n+1}}{1-a}, \qquad a \neq 1$$

(iv) $$\sum_{r=0}^{n} ra^r = \frac{a[1-(n+1)a^n + na^{n+1}]}{(1-a)^2}, \qquad a \neq 1$$

Infinite summations

(i) $\displaystyle\sum_{r=0}^{\infty} a^r = \frac{1}{1-a},$ for $|a| < 1$ and diverges for $|a| \geq 1$.

(ii) $\displaystyle\sum_{r=0}^{\infty} ra^r = \frac{a}{(1-a)^2},$ for $|a| < 1$ and diverges otherwise.

Appendix B

Table of bilateral z-transforms and properties

B.1 Table of bilateral z-transforms

B.2 Properties of z-transforms

Let $\mathbb{C}$ denote the set of complex numbers.
Suppose $\mathcal{Z}[\{x(k)\}] = X(z)$ denotes the bilateral z-transform of $\{x(k)\}$, with region of convergence $R_x = \{z \in \mathbb{C} \ : \ \alpha_1 < |z| < \alpha_2\}$, and $\mathcal{Z}[\{y(k)\}] = Y(z)$ denotes the bilateral z-transform of $\{y(k)\}$, with region of convergence $R_y = \{z \in \mathbb{C} \ : \ \beta_1 < |z| < \beta_2\}$. Let $X_+(z) = \mathcal{Z}[\{x(k)\zeta(k)\}]$ denote the unilateral z-transform of $\{x(k)\}$.

a) **Linearity**: For any constants a and b,

$$\mathcal{Z}[\{ax(k) + by(k)\}] = aX(z) + bY(z)$$

with region of convergence $R_x \cap R_y$.

b) **Delay/Advance property**:

$$\mathcal{Z}[\{x(k \pm m)\}] = z^{\pm m}X(z), \quad \text{with region of convergence } R_x.$$

Consider the sequence $\{x(k)\zeta(k)\} = \{x(0),\ x(1),\ x(2), \ldots\}$, where $\{\zeta(k)\}$

Sequence	Transform	Region of convergence
$\delta(k)$	1	all z
$a^k\zeta(k)$	$\dfrac{z}{z-a}$	$\lvert z\rvert > \lvert a\rvert$
$-b^k\zeta(-k-1)$	$\dfrac{z}{z-b}$	$\lvert z\rvert < \lvert b\rvert$
$\dfrac{k(k-1)\dots(k-(n-2))}{(n-1)!}a^{k-n+1}\zeta(k)$	$\dfrac{z}{(z-a)^n}$	$\lvert z\rvert > \lvert a\rvert$
$\sin(\omega k)\zeta(k)$	$\dfrac{z\sin(\omega)}{z^2-2z\cos(\omega)+1}$	$\lvert z\rvert > 1$
$\cos(\omega k)\zeta(k)$	$\dfrac{z(z-\cos(\omega))}{z^2-2z\cos(\omega)+1}$	$\lvert z\rvert > 1$
$\dfrac{-a^k}{k}\zeta(k-1)$	$\log_e(1-az^{-1})$	$\lvert z\rvert > \lvert a\rvert$
$\dfrac{a^{-k}}{k}\zeta(-k-1)$	$\log_e(1-az)$	$\lvert z\rvert < \lvert a\rvert^{-1}$

Table B.1: Table of bilateral z-transforms.

denotes the unit step sequence. Then

$$\mathcal{Z}\left[\{x(k+m)\zeta(k)\}\right] = z^m X_+(z) - z^m \sum_{j=0}^{m-1} x(j)z^{-j}.$$

c) **Multiplication by an exponential** a^k:

$$\mathcal{Z}\left[\left\{a^k x(k)\right\}\right] = X\left(\frac{z}{a}\right),$$

with region of convergence $\{z \in \mathbb{C}\ :\ |a|\alpha_1 < |z| < |a|\alpha_2\}$.

d) **Multiplication by** k:

$$\mathcal{Z}\left[\{kx(k)\}\right] = -z\frac{\mathrm{d}X}{\mathrm{d}z}(z), \quad \text{with region of convergence } R_x.$$

e) **Convolution**: For $x(k) * y(k) = \sum_{m=-\infty}^{\infty} x(m)y(k-m)$,

$$\mathcal{Z}\left[\{x(k) * y(k)\}\right] = X(z)Y(z)$$

with region of convergence $R_x \cap R_y$.

f) **Initial value theorem**: For the sequence $\{x(k)\}$ that is zero for $k < k_0$, then $x(k_0)$ can be evaluated using the result

$$x(k_0) = \lim_{z\to\infty} z^{k_0} X(z).$$

g) **Final value theorem**:
If $\lim_{k\to\infty} x(k)$ exists, then $\lim_{k\to\infty} x(k) = \lim_{z\to1}[z^{-1}(z-1)X(z)]$.

If $\{x(k)\}$ is a causal sequence, then $\lim_{k\to\infty} x(k) = \lim_{z\to1}[(z-1)X_+(z)]$.

h) **Inversion**: If $\{x(k)\} = \mathcal{Z}^{-1}\left[X(z)\right]$ and the singularities of $X(z)$ are poles of finite order, then

$$\{x(k)\} = \begin{cases} \displaystyle\sum_{\substack{\text{all poles of } X(z)z^{k-1}\\ \text{inside } C}} \left[\text{residue of } X(z)z^{k-1}\right], & k \geq 0, \\ \displaystyle -\sum_{\substack{\text{all poles of } X(z)z^{k-1}\\ \text{outside } C}} \left[\text{residue of } X(z)z^{k-1}\right], & k < 0, \end{cases}$$

where C is a closed contour which lies within the region of convergence.

Appendix C

Tables of Fourier transforms and properties

All constants, for example $A,\ a,\ \omega_0$, are assumed to be real and positive. For finite energy signals, see Table C.1, whilst for power signals , see Table C.2.

C.1 Tables of Fourier transforms

C.2 Properties of the Fourier transform

Suppose $\mathcal{F}\left\{f(t)\right\} = F(i\omega)$ and $\mathcal{F}\left\{g(t)\right\} = G(i\omega)$, where $\mathcal{F}\left\{\cdot\right\}$ denotes Fourier transform.

a) **Linearity:** For any constants a and b,

$$\mathcal{F}\left\{af(t) + bg(t)\right\} = aF(i\omega) + bG(i\omega).$$

b) **Symmetry:**

$$\mathcal{F}\left\{F(it)\right\} = 2\pi f(-\omega).$$

If $t \mapsto f(t)$ is an even function, then

$$\mathcal{F}\left\{F(it)\right\} = 2\pi f(\omega).$$

Function	Transform
$e^{-at}\zeta(t)$	$1/(a+i\omega)$
$\begin{cases} 1, & \vert t\vert < a \\ 0, & \text{otherwise} \end{cases}$	$2a\text{sinc}(\omega a)$
$\begin{cases} A\left[1-\frac{\vert t\vert}{a}\right], & \vert t\vert < a \\ 0, & \text{otherwise} \end{cases}$	$Aa\text{sinc}^2\left(\frac{\omega a}{2}\right)$
$e^{-a\vert t\vert}$	$2a/(a^2+\omega^2)$
e^{-at^2}	$(\pi/a)^{\frac{1}{2}}\exp\left(-\omega^2/(4a)\right)$
$1/(a^2+t^2)$	$(\pi/a)\exp(-a\vert\omega\vert)$

Table C.1: Table of Fourier transforms for finite energy signals.

Function	Transform
1	$2\pi\delta(\omega)$
$\delta(t)$	1
$\zeta(t)$	$\pi\delta(\omega)+1/(i\omega)$
$\text{sgn}(t)$	$2/(i\omega)$
$\exp(i\omega_0 t)$	$2\pi\delta(\omega-\omega_0)$
$\vert t\vert$	$-2/(\omega^2)$

Table C.2: Table of Fourier transforms for power signals.

c) **Change of scale:** For a real constant $\alpha \neq 0$,

$$\mathcal{F}\left\{f(\alpha t)\right\} = \frac{1}{|\alpha|} F\left(i\frac{\omega}{\alpha}\right).$$

d) **Time shift (Delay):**

$$\mathcal{F}\left\{f(t-t_0)\right\} = F(i\omega)\exp(-i\omega t_0).$$

e) **Frequency shift (Modulation):**

$$\mathcal{F}\left\{f(t)\exp(i\omega_0 t\right\} = F(i(\omega-\omega_0)).$$

f) **Frequency differentiation and integration:**

(i) $\mathcal{F}\left\{-itf(t)\right\} = \dfrac{\mathrm{d}F(i\omega)}{\mathrm{d}\omega}$

(ii) $\mathcal{F}\left\{\dfrac{f(t)}{-it}\right\} = \displaystyle\int F(i\omega)\,\mathrm{d}\omega.$

g) **Time differentiation and integration:**

(i) $\mathcal{F}\left\{f'(t)\right\} = i\omega F(i\omega)$, assuming $f \to 0$ as $t \to \pm\infty$.
In general, assuming $f,\ f', \ldots f^{(n-1)} \to 0$ as $t \to \pm\infty$,

$$\mathcal{F}\left\{f^{(n)}(t)\right\} = (i\omega)^n F(i\omega).$$

(ii) $\mathcal{F}\left\{\displaystyle\int_{-\infty}^{t} f(u)\,\mathrm{d}u\right\} = \dfrac{1}{i\omega}F(i\omega) + \pi F(0)\delta(\omega).$

h) **Convolution:**

$$\mathcal{F}\left\{f(t)g(t)\right\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(iu)G(i(\omega-u))\,\mathrm{d}u = \frac{1}{2\pi}F(i\omega) * G(i\omega)$$

and

$$\mathcal{F}^{-1}\left\{F(i\omega)G(i\omega)\right\} = \int_{-\infty}^{\infty} f(u)g(t-u)\,\mathrm{d}u = f(t) * g(t).$$

Appendix D

Proof of Proposition 4.1

Proof:

$$\begin{aligned}\mu_{\tilde{y}} = \mathcal{E}\left[\tilde{y}(k)\right] = \mathcal{E}\left[\sum_{m=-\infty}^{\infty} w(m)\tilde{x}(m,k)\right] &= \sum_{m=-\infty}^{\infty} w(m)\mathcal{E}\left[\tilde{x}(m,k)\right] \\ &= \sum_{m=-\infty}^{\infty} w(m)\mu_{\tilde{x}}, \qquad \text{since } x(k) \text{ is wide-sense stationary,} \\ &= \mu_{\tilde{x}} \sum_{m=-\infty}^{\infty} w(m).\end{aligned}$$

and, hence, $\mu_{\tilde{y}}$ is independent of k.

$$\begin{aligned}R_{\tilde{y}\tilde{y}}(m_1, m_2) &= \mathcal{E}\left[\tilde{y}(m_1)\tilde{y}(m_2)\right] \\ &= \sum_{r_1}\sum_{r_2} w(r_1)w(r_2)\mathcal{E}\left[\tilde{x}(r_1, m_1)\tilde{x}(r_2, m_2)\right] \\ &= \sum_{r_1}\sum_{r_2} w(r_1)w(r_2)R_{\tilde{x}\tilde{x}}(r_1, r_2, m_2 - m_1),\end{aligned}$$

since $x(k)$ is wide-sense stationary.
Clearly, $R_{\tilde{y}\tilde{y}}(m_1, m_2)$ is a function of $m_2 - m_1$, only.

□

Appendix E

Answers to the exercises

E.1 Chapter 1

1. a) linear; time-varying; not memoryless; causal;

   b) nonlinear; time-invariant; memoryless; causal;

   c) linear; time-invariant; not memoryless; causal;

   d) linear; time-varying; not memoryless; non-causal.

2. $\zeta(k) - \zeta(k-11) + \zeta(k-16) - \zeta(k-21)$

3. $z(0) = 2,\ \ z(1) = 5,\ \ z(2) = 6.5,\ \ z(3) = 3.25,\ \ z(4) = 1.625,$
   $z(5) = 0.8125,\ \ z(6) = 0.40625$

4. $\dfrac{(1-a^{k+1})}{1-a}\zeta(k)$

5. a) $\{5,\ -7,\ 7,\ 5,\ 16,\ 2,\ 20\}$;

   b) $\{(x * y)(k); k = -1:1:6\} = \{6,\ -6,\ 0,\ -10,\ 2,\ 2,\ 4,\ 2\}$

6. a) $\zeta(k) - \zeta(k-M)$;

   b) $(k+1)\zeta(k) + (M-k-1)\zeta(k-M)$ or, alternatively, $\displaystyle\sum_{m=0}^{M-1} \zeta(k-m)$.

7. $(f * g)(t) = \begin{cases} 0, & t \le T_1, \\ \frac{1}{a}\left(1 - e^{-a(t-T_1)}\right), & T_1 < t < T_2, \\ \frac{1}{a}\left(e^{-a(t-T_2)} - e^{-a(t-T_1)}\right), & t \ge T_2. \end{cases}$

E.2 Chapter 2

1. a) AR, time-invariant; b) ARMA, time-varying; c) MA, time-invariant.

2. a) $1+az^{-1}+a^2z^{-2}+a^3z^{-3}+a^4z^{-4}+a^5z^{-5}$, converges for all z except $z=0$;

   b) $2z\left(\dfrac{3}{2z+1}+\dfrac{1}{z-3}\right)$, $\frac{1}{2}<|z|<3$;

   c) $\dfrac{1}{2}+\dfrac{1}{z}-\dfrac{1}{3z^2}$, converges for all z except $z=0$;

   d) $\dfrac{z}{(z-1)^2}$, $|z|>1$.

3. $x(k) = a^k\zeta(k)+\zeta(k-11)$; $X(z) = \dfrac{z}{z-a}+\dfrac{z^{-10}}{z-1}$, $|z| > \max\{1, |a|\}$

4. (i) $(x*y)(k) = \delta(k)-2\delta(k-2)+\frac{1}{2}\zeta(k)-\zeta(k-2)$ and so $\mathcal{Z}\left[\{(x*y)(k)\}\right] = 1-2z^{-2}+\dfrac{0.5z}{z-1}-\dfrac{z^{-1}}{z-1}$.

   (ii) $X(z) = \mathcal{Z}\left[\{x(k)\}\right] = 1+\dfrac{0.5z}{z-1}$ and $Y(z) = \mathcal{Z}\left[\{y(k)\}\right] = 1-2z^{-2}$ and so $\mathcal{Z}\left[\{(x*y)(k)\}\right] = X(z)Y(z) = 1-2z^{-2}+\dfrac{0.5z}{z-1}-\dfrac{z^{-1}}{z-1}$.

5. $X(z) = \dfrac{2z^3}{2z-1}-z^2-\dfrac{1}{2}z = \dfrac{1}{4}\left(\dfrac{z}{z-\frac{1}{2}}\right)$, $|z| > \frac{1}{2}$

6. a) $\left\{(1+2^k)\zeta(k)\right\}$; b) $\left\{[3(4)^{k-1}-(3+k)(2)^{k-2}]\zeta(k)\right\}$;
   c) $\left\{(2-2^{k-1})\zeta(k)-\frac{3}{2}\delta(k)\right\}$ or $\left\{(2-2^{k-1})\zeta(k-1)\right\}$

7. $y(k) = 1+\frac{1}{7}(-1)^{k+1}-\frac{1}{7}(\frac{1}{6})^{k-1}$

8. a) $(1-2^k)\zeta(k)-3^k\zeta(-k-1)$; b) $(1+3^k-2^k)\zeta(k)$;
   c) $(2^k-3^k-1)\zeta(-k-1)$

9. a) $y(k) - \frac{1}{4}y(k-2) = x(k)$

   b) $y_\zeta(k) = \left[\frac{1}{6}\left(-\frac{1}{2}\right)^k - \left(\frac{1}{2}\right)^{k+1} + \frac{4}{3}\right]\zeta(k).$
   Since $y_\delta(k) = y_\zeta(k) - y_\zeta(k-1)$, it follows that

$$\begin{aligned} y_\delta(k) = &\left[\frac{1}{6}\left(-\frac{1}{2}\right)^k - \left(\frac{1}{2}\right)^{k+1} + \frac{4}{3}\right]\zeta(k) \\ &- \left[\frac{1}{6}\left(-\frac{1}{2}\right)^{k-1} - \left(\frac{1}{2}\right)^{k} + \frac{4}{3}\right]\zeta(k-1). \end{aligned}$$

   Alternatively, since $y_\delta(k) = \mathscr{Z}^{-1}\left[H(z)\right]$,

$$y_\delta(k) = \frac{1}{2}\left[\left(\frac{1}{2}\right)^k + \left(-\frac{1}{2}\right)^k\right]\zeta(k).$$

   The AR process is an IIR process since there are an infinite number of non-zero terms in $\{y_\delta(k)\}$.

10. $x(k) = 2^k\left[2\sin\left(\frac{\pi}{2}k\right) - 3\cos\left(\frac{\pi}{2}k\right)\right]\zeta(k)$

11. $y(k) = \frac{1}{1-a}(1 - a^{k+1})\zeta(k)$;
    Stable system since $H(z)$ has a pole at $z = a$, which lies in the unit circle in the z-plane.

12. a) $y(k) - \frac{1}{2}y(k-1) = x(k)$

    b) Using `x=[2 4 4 0 0 0 0]; a=[1 -0.5]; b=[1]; y=filter(b,a,x)`, gives $y(0) = 2,\ \ y(1) = 5,\ \ y(2) = 6.5,\ \ y(3) = 3.25,\ \ y(4) = 1.625,\ \ y(5) = 0.8125,\ \ y(6) = 0.4063$.

    c) (i) Yes (ii) Yes

13. a) Yes, since $h(k) = 0$ for $k < 0$.

    b) Yes, since $H(z)$ has a pole at $z = -\frac{1}{4}$, which lies in $\{z \in \mathbb{C} : |z| < 1\}$.

    c) IIR, since $h(k)$ has an infinite number of non-zero terms.

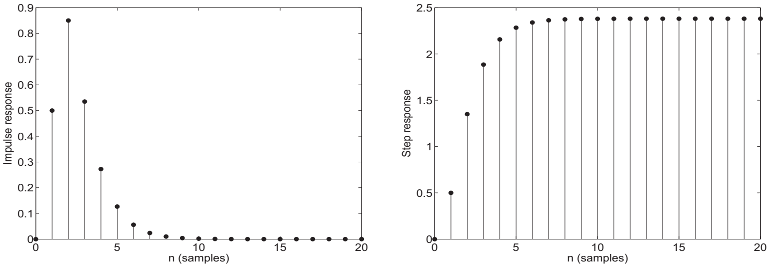
14. a) $y_\delta(k) = \left[7\left(\frac{2}{5}\right)^{k-1} - \frac{13}{2}\left(\frac{3}{10}\right)^{k-1}\right]\zeta(k-1);$

    $y_\zeta(k) = \left[\frac{50}{21} - \frac{35}{3}\left(\frac{2}{5}\right)^k + \frac{65}{7}\left(\frac{3}{10}\right)^k\right]\zeta(k)$

    b) The MATLAB® commands:

```
a=[1 -0.7 0.12]; b=[0 0.5 0.5];
impz(b,a,21);
stepz(b,a,21)
```

give the following:



15. $\left[\frac{13}{6} - \frac{2}{3}\left(\frac{2}{5}\right)^k\right]\zeta(k) + \frac{1}{2}(3)^{k+1}\zeta(-k-1)$

16. a) $y(k+2) = x(k+2) - x(k+1) + \frac{1}{4}y(k+1) + \frac{1}{8}y(k)$;

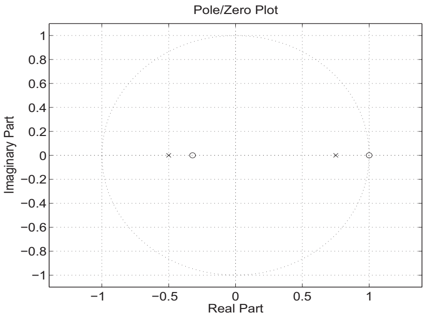
b) $\frac{1}{3}\left[5(-\frac{1}{4})^k - 2(\frac{1}{2})^k\right]\zeta(k)$;

c) No, since $H(z)$ has a zero at $z = 1$.

17. a) $\dfrac{(z-1)(7z+\frac{9}{4})}{(z+\frac{1}{2})(z-\frac{3}{4})}$; The region of convergence is $\{z \in \mathbb{C} : |z| > \frac{3}{4}\}$.

The following plot was obtained from MATLAB® using the commands:

```
a=[1 -0.25 -0.375]; b=[7 -4.75 -2.25]; fvtool(b,a):
```



b) The system function has poles at $z = -\frac{1}{2},\ \frac{3}{4}$, which lie in the unit circle $|z| = 1$.

c) No, since the zero $z = 1$ does not lie inside the unit circle.

d) $6\delta(k) + \left[3\left(-\frac{1}{2}\right)^k - 2\left(\frac{3}{4}\right)^k\right]\zeta(k)$

e) $y(k) = \frac{1}{4}y(k-1) + \frac{3}{8}y(k-2) + 7x(k) - \frac{19}{4}x(k-1) - \frac{9}{4}x(k-2)$

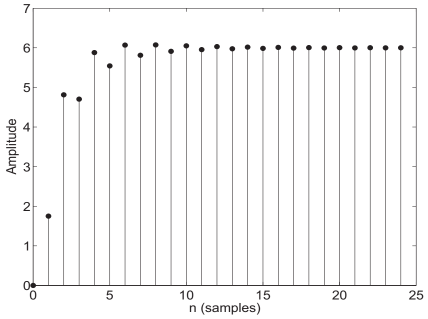
18. a) $H(z) = \dfrac{7(z+2)}{4(z-\frac{1}{2})(z+\frac{3}{4})}; \quad \{z \in \mathbb{C} : |z| > \frac{3}{4}\}$

b) Since the filter is stable, the region of convergence is of the form $\{z \in \mathbb{C} : |z| > \frac{3}{4}\}$.
Hence, the filter is causal.

c) $\left[6 - 7\left(\frac{1}{2}\right)^k + \left(-\frac{3}{4}\right)^k\right]\zeta(k)$
The MATLAB® commands:

```
a=[1 0.25 -0.375]; b=[0 1.75 3.5];
stepz(b,a,25)
```

produce the plot:



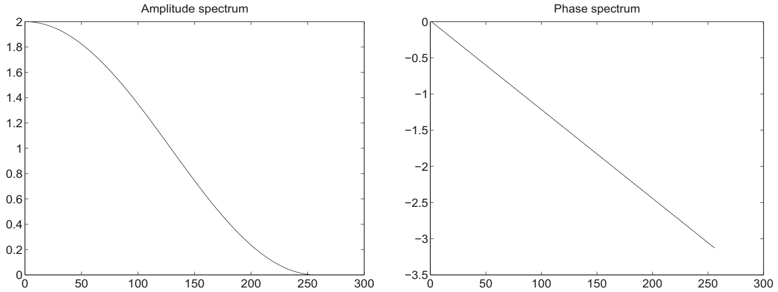
19. BIBO stable: $-1 < \alpha < \frac{1}{2}$

E.3 Chapter 3

1. a) $H(e^{-i\theta}) = e^{-i\theta}(1 + \cos(\theta))$;
Since $y(k) = \frac{1}{2}x(k) + x(k-1) + \frac{1}{2}x(k-2)$, then, the MATLAB® commands:

```
a=[1]; b=[0.5 1 0.5];
[h,w]=freqz(b,a,256);
mag=abs(h); phase=angle(h);
plot(mag), title(’Amplitude spectrum’);
plot(phase),title(’Phase spectrum’)
```

give the following plots:



b) $\frac{1}{2}\zeta(k) + \zeta(k-1) + \frac{1}{2}\zeta(k-2)$

2. a) $h(k) = \begin{cases} 1, & k=-1, \\ 1, & k=1, \\ 0, & \text{otherwise;} \end{cases}$ b) no; c) yes;

d) FIR; e) $2\cos(\theta)$; f) yes.

3. a) $|H(e^{i\theta})| = \dfrac{12}{\sqrt{17+8\cos(\theta)}}$ and $\arg H(e^{i\theta}) = \tan^{-1}\left(\dfrac{\sin(\theta)}{4+\cos(\theta)}\right)$

b) $|H(e^{i0})| = \dfrac{12}{5}, \ |H(e^{i\frac{\pi}{3}})| = \dfrac{12}{\sqrt{21}}, \ |H(e^{i\pi})| = 4;$

$$\arg H(e^{i0}) = \arg H(e^{i\pi}) = 0, \ \arg H(e^{i\frac{\pi}{3}}) = \Theta \overset{\text{def}}{=} \tan^{-1}\left(\frac{\sqrt{3}}{9}\right);$$

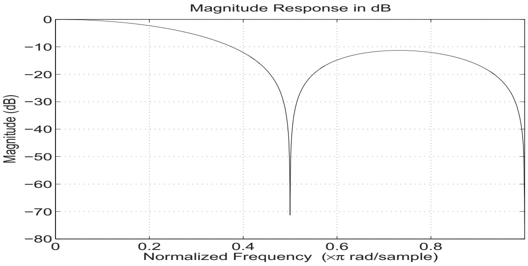
$$y(k) = 12 + \frac{4}{\sqrt{21}}\cos\left(\tfrac{1}{3}\pi + \Theta\right) - 4\sin\left(\pi k + \tfrac{1}{4}\pi\right)$$

4. $|H(e^{i\theta})| = \dfrac{1}{M}\left|\dfrac{\sin\left(\frac{\theta M}{2}\right)}{\sin\left(\frac{\theta}{2}\right)}\right| = \left|\dfrac{\operatorname{sinc}\left(\frac{\theta M}{2}\right)}{\operatorname{sinc}\left(\frac{\theta}{2}\right)}\right|;$

Using the MATLAB® commands:

```
a=[1]; b=[0.25 0.25 0.25 0.25];
fvtool(b,a)
```

the following graph was obtained:



5. a) $H(z) = az/\left(z - \frac{1}{2}\right)$. Since there is a simple pole at $z = \frac{1}{2}$, the filter is BIBO stable.

   b) (i) Since $H(e^{i\theta}) = a/\left(1 - \frac{1}{2}e^{-i\theta}\right)$, the condition $|H(e^{i0})| = 1$ gives $a = \frac{1}{2}$.

   (ii) Consider $x(k) = 2e^{i0} - 3\Re\{e^{i\frac{\pi}{3}k}\} + 6\Im\{e^{i\pi k}\}$.

θ	$\lvert H(e^{i\theta})\rvert$	$\arg H(e^{i\theta})$
0	1	0
$\frac{\pi}{3}$	$\frac{1}{\sqrt{3}}$	$-\frac{\pi}{6}$
π	$\frac{1}{3}$	0

   Therefore, the steady-state output is

$$y(k) = 2 - \sqrt{3}\cos\left((2k-1)\tfrac{\pi}{6}\right) + 2\sin(\pi k)\,.$$

6. a) For $\theta \in \left[-\frac{3\pi}{4}, \frac{3\pi}{4}\right]$, $|H(e^{i\theta})| = 4$ and $\arg H(e^{i\theta}) = -\theta$.

   Therefore, $H(e^{i\theta}) = \begin{cases} 4e^{-i\theta}, & \theta \in \left[-\frac{3\pi}{4}, \frac{3\pi}{4}\right] \\ 0, & \text{otherwise.} \end{cases}$

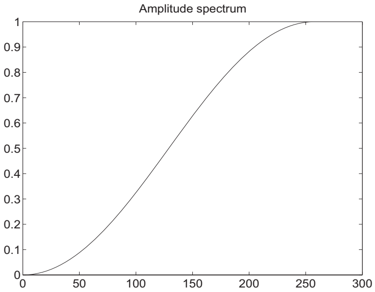
   The result $h(k) = \dfrac{1}{2\pi}\displaystyle\int_{-\pi}^{\pi} H(e^{i\theta})e^{ik\theta}\, \mathrm{d}\theta$ gives $h(k) = 3\,\mathrm{sinc}\left(\frac{3}{4}\pi(k-1)\right)$

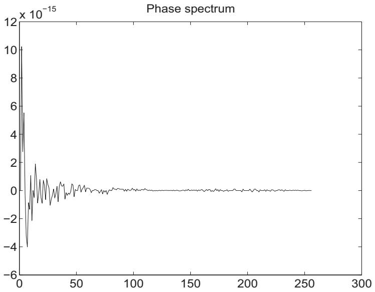
   b) IIR

   c) noncausal

7. a) yes; b) no; c) $\frac{1}{2}(1 - \cos(\theta))$;

```
a=[0 1]; b=[-0.25 0.5 -0.25];
[h,w]=freqz(b,a,256);
mag=abs(h); phase=angle(h);
plot(mag), title('Amplitude spectrum');
plot(phase),title('Phase spectrum')
```





8. a) $y(k) = x(k-1) + 2x(k-2) + x(k-3)$

b) $y_\delta(k) = \delta(k-1) + 2\delta(k-2) + \delta(k-3)$;
$y_\zeta(k) = \zeta(k-1) + 2\zeta(k-2) + \zeta(k-3)$

c) $2|1 + \cos(\theta)|$

9. $L(z)$ has a single simple pole $z = -\ell_2^{-1}$. Since $\ell_2 > 1$, the pole lies in the open unit disk in the z-plane;

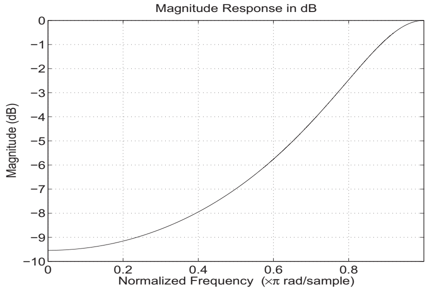
$$|L(e^{i\theta})| = \ell_1 \sqrt{\frac{2(1 - \cos(\theta))}{1 + \ell_2^2 + 2\ell_2 \cos(\theta)}}$$

10. a) $\dfrac{e^{i\theta}}{e^{i\theta} - \alpha} - \dfrac{e^{i\theta}}{e^{i\theta} - 0.8}$; c) $\dfrac{91}{109} = 0.8349$ (4D)

11. $H(e^{i\theta}) = \dfrac{1}{2e^{i\theta} + 1} = \dfrac{2e^{-i\theta} + 1}{5 + 4\cos(\theta)}$;
The commands:

```
a=[2 1]; b=[0 1];
fvtool(b,a)
```

give:

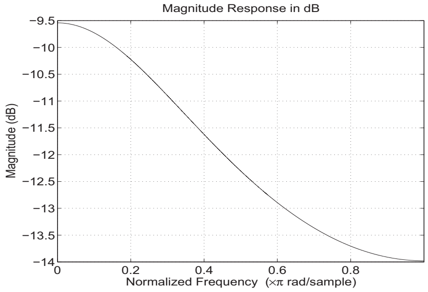


12. $H(e^{i\theta}) = \dfrac{1 - a\cos(\theta) - ia\sin(\theta)}{1 - 2a\cos(\theta) + a^2}$; $\quad y(k) = \dfrac{\cos(k\theta) - a\cos((k+1)\theta)}{1 - 2a\cos(\theta) + a^2}$

13. $H(z) = \dfrac{2z+1}{(4z-1)(2z+1)}$;
The system is stable since the poles $z = \frac{1}{4},\ -\frac{1}{2}$ lie in the open unit disk. The MATLAB® commands:

```
a=[8 2 -1]; b=[0 2 1];
fvtool(b,a)
```

give:



14. $|F(i\omega)| = 2A\omega/(\alpha^2 + \omega^2)$; $\qquad \arg F(i\omega) = -\frac{\pi}{2}$.

15. $\dfrac{T^2}{2\pi}\mathrm{sinc}^4\left(\frac{1}{2}\omega T\right)$

16. a) $E(x) = \frac{\pi}{2}$

b) Show that $X(i\omega) = \pi f(\omega)$, where $f(\omega) = \begin{cases} 1, & |t| < 1 \\ 0, & \text{otherwise} \end{cases}$,

and so $E(x) = \pi$

17. $g(t) = \dfrac{a}{2\pi}\,[\operatorname{sinc}(a(t+1)) + \operatorname{sinc}(a(t-1))]$; At least 0.5 Hz.

18. b) Not exactly, since $X(i\omega)$ is not bandlimited.

c) Since $X(i\omega)$ is not bandlimited, there is no Nyquist rate.

19. (i) $6\pi[\delta(\omega - 10\pi) + \delta(\omega + 10\pi)]$

(ii) $f_s = 7$,

$$X_s(i\omega) = 42\pi \sum_{n=-\infty}^{\infty} [\delta(\omega - 10\pi - 14\pi n) + \delta(\omega + 10\pi - 14\pi n)];$$

$f_s = 14$,

$$X_s(i\omega) = 84\pi \sum_{n=-\infty}^{\infty} [\delta(\omega - 10\pi - 28\pi n) + \delta(\omega + 10\pi - 28\pi n)];$$

$$X_s(i\omega)H(i\omega) = \begin{cases} 6\pi[\delta(\omega - 4\pi) + \delta(\omega + 4\pi)], & f_s = 7, \\ 6\pi[\delta(\omega - 10\pi) + \delta(\omega + 10\pi)], & f_s = 14. \end{cases}$$

20. 112; 0.0002 s

21. a) 20 s; b) 0.002π rad; c) 0.05 Hz; d) 50π rad./sec.;

e) $P(10) = P(990) = 0.01$, $P(11) = P(989) = 0.002$, $P(n) = 0$ otherwise.

22. a) Spacing: $\frac{25}{6}\pi$

Frequencies: $\Re[X(n)]:\ 0,\ \frac{25}{2}\pi,\ \frac{125}{6}\pi,\ \frac{100}{3}\pi,\ \frac{275}{6}\pi,\ \frac{325}{6}\pi$ rad/sec

$\Im[X(n)]:\ \frac{25}{2}\pi,\ \frac{325}{6}\pi$ rad/sec

b) $P(0) = \frac{1}{256}$, $P(3) = P(13) = \frac{41}{256}$, $P(5) = P(11) = \frac{9}{256}$, $P(8) = \frac{1}{64}$, $P(n) = 0$ otherwise.

23. The MATLAB® commands:

```
x=[j 1+j 2 1-j -j];
sx=sum(x.*conj(x))
y=fft(x);
syn=sum(y.*conj(y))/5
```

produce the results: `sx=10` and `syn=10`.

24. $P(0) = 0, \qquad P(1) = \frac{9}{32} = 0.28125, \qquad P(2) = \frac{1}{16} = 0.0625,$
$P(3) = \frac{9}{32} = 0.28125$
These results can be obtained using MATLAB® with the commands:

```
x=[0 1.5*(1+j) 1 1.5*(1-j)];
pspec=x.*conj(x)/(length(x)∧2)
```

25. The following MATLAB® commands:

```
a=[1.0000 1.1619 0.6959 0.1378];
b=[0.0495 -0.1486 0.1486 -0.0495];
[h,w]=freqz(b,a,256);
mag=abs(h); phase=angle(h);
figure(1); plot(w/pi,mag), grid,
title('Amplitude spectrum'),
xlabel('Frequency in multiples of pi');
figure(2); plot(w/pi,phase), grid,
title('Phase spectrum'),
xlabel('Frequency in multiples of pi');
k=[0:100];
x=cos(0.3*pi*k)-2*cos(0.6*pi*k)+cos(0.7*pi*k)...
...+4*sin(0.8*pi*k)-3*sin(0.9*pi*k);
y=filter(b,a,x);
w=2*pi/length(y)*k; as=abs(fft(y));
figure(3); plot(w/pi,as), grid,
title('Amplitude spectrum'),
xlabel('Frequency in multiples of pi')
```

produce the graphs illustrated in Figures E.1 and E.2.
Figure E.1 indicates that the filter is high-pass and this is confirmed in Figure E.2, which shows that the components of $\{x(k)\}$ with frequencies 0.3π, 0.6π and 0.7π have been attenuated by the high-pass filter and the components with frequencies 0.8π and 0.9π have been passed through the filter with little attenuation.

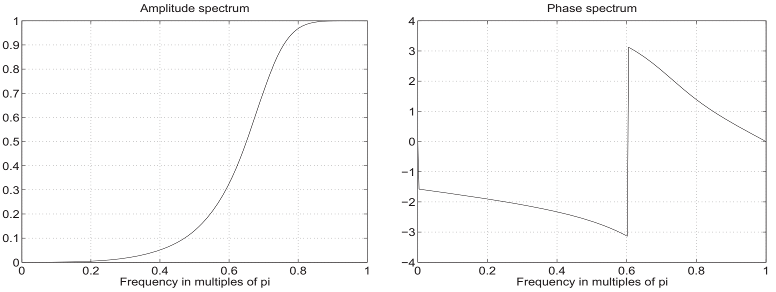


Figure E.1: Amplitude and phase spectra for the filter.

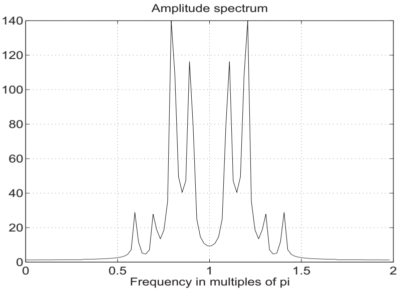


Figure E.2: Amplitude spectrum for the filtered signal.

E.4 Chapter 4

1. $R_{yy}(k) = \sigma_x^2 h(k) * h(-k)$ and $P_{av}(y(k)) = \sigma_x^2 h(0) * h(0) = \sigma_x^2 \sum_{k=-\infty}^{\infty} [h(k)]^2$.

2. $R_{nm}(k) = (h * R_{nn})(k) = \frac{1}{4}\delta(k) + \frac{1}{8}\delta(k-3)$;
   [Hint: To find $R_{mm}(k)$, you may use: $\mathcal{Z}\left[\{h(k) * h(-k)\}\right] = H(z)H(z^{-1})$ and then take inverse z-transforms to find $h(k) * h(-k)$.]

3. c) $\frac{448}{405}\sigma_x^2$

4. $\mu_x^2 + \frac{1}{2}\sigma_x^2$

5. a) $R_{ww}(k) = 2\delta(k) - \delta(k-3) - \delta(k+3)$ and $R_{nw}(k) = \delta(k) - \delta(k-3)$
   b) $\overline{w^2(k)} = 2$; $g(k) = k\zeta(k) - (k-3)\zeta(k-3)$
   or $g(k) = \delta(k-1) + 2\delta(k-2) + 3\zeta(k-3)$
   c) $(S/N)_{out} = \frac{9}{4}$.

9. $\frac{1}{2}e^{-|\tau|}$

10. a) $4 - \cos(\theta)$; b) 4

11. $\dfrac{c(a - a^{-1})z}{(z-a)(z-a^{-1})}$, $|a| < |z| < |a|^{-1}$

12. $R_{xx}(k) = \frac{1}{2}\delta(k+1) + \delta(k) + \frac{1}{2}\delta(k-1)$, $P_{av}(x) = 1$, $\sigma_x^2 = \frac{3}{4}$

13. a) $\dfrac{z(2z - 0.25)(2 - 0.25z)}{(z-0.5)(z+0.25)(1-0.5z)(1+0.25z)}$, $0.5 < |z| < 2$
    b) $\overline{y^2(k)} = \dfrac{376}{45}$

14. $H(z) = \dfrac{2.5z}{(z-0.2)(z-0.6)}$; $h(k) = 6.25[(0.6)^k - (0.2)^k]\zeta(k)$

15. a) Now $S_{nv}(z) = H(z)S_{nn}(z)$ and, therefore,

$$H(z) = S_{nv}(z)/S_{nn}(z) = \frac{2z+1}{4z-3}.$$

    c) $\overline{v^2(k)} = R_{vv}(0) = \mathcal{Z}^{-1}\left[\dfrac{12z}{(4z-3)(4-3z)}\right]\Bigg|_{k=0} = \dfrac{12}{7}$.

16. $S_{yy}(\theta) = \sigma_x^2 \cos^2\left(\frac{1}{2}\theta\right)$; $\qquad S_{xy}(z) = \frac{1}{2}\sigma_x^2\left(1 + z^{-1}\right)$

17. a) (i) $R_{xx}(k) = 2\delta(k)$

(ii) $S_{xx}(\theta) = 2$

(iii) $S_{yy}(\theta) = \begin{cases} 8, & 0 < |\theta| < \frac{\pi}{2}, \\ 0, & \frac{\pi}{2} < |\theta| < \pi, \end{cases}$

$S_{yy}(\theta) = S_{yy}(\theta + 2\pi)$;

b) (i) The same as that in Exercise 17. a).

(ii) The same as that in Exercise 17. b).

(iii) $S_{yy}(\theta) = \begin{cases} 8, & \frac{\pi}{4} < |\theta| < \frac{\pi}{2}, \\ 0, & \text{otherwise}, \end{cases}$

$S_{yy}(\theta) = S_{yy}(\theta + 2\pi)$;

$\overline{y^2(k)} = \begin{cases} 4, & \text{for Exercise 17. a)} \\ 2, & \text{for Exercise 17. b)} \end{cases}$

18. 1.695×10^{-6}

Bibliography

[Balmer, 1997] Balmer, L. (1997). *Signals and systems - an introduction.* Prentice Hall, Harlow, Essex, U.K., 2nd edition.

[Bellanger, 1990] Bellanger, M. (1990). *Digital processing of signals - theory and practice.* John Wiley & Sons, Chichester, West Sussex, U.K.

[Bernd, 2004] Bernd, J. (2004). *Practical Handbook on image processing for scientific and technical applications.* CRC Press, Boca Raton, Florida, U.S.A., 2nd edition.

[Bovik, 2005] Bovik, A. (2005). *Handbook of Image and Video Processing.* Elsevier Academic Press, San Diego, California, U.S.A.

[Chapman et al., 1997] Chapman, M., Goodall, D., and Steele, N. (1997). *Signal processing in electronic communications.* Horwood Publishing, Chichester, West Sussex, U.K.

[Commowick et al., 2008] Commowick, O., Arsigny, V., Isambert, A., Costa, J., Dhermain, F., Bidault, F., Bondiau, P.-Y., Ayache, N., and Malandain, G. (2008). An efficient locally affine framework for the smooth registration of anatomical structures. *Medical Image Analysis*, 12:427–441.

[Deller et al., 2000] Deller, J., Hansen, J., and Proakis, J. (2000). *Discrete time processing of speech signals.* Wiley-IEEE Press, New York, U.S.A., 2nd edition.

[Flanagan, 1972] Flanagan, J. (1972). *Speech analysis, synthesis and perception.* Springer-Verlag, New York, U.S.A.

[Forsyth and Ponce, 2003] Forsyth, D. and Ponce, J. (2003). *Computer Vision A Modern Approach, Int. Ed.* Pearson Education, Upper Saddle River, New Jersey, U.S.A.

[Gabel and Roberts, 1987] Gabel, R. and Roberts, R. (1987). *Signals and linear systems.* John Wiley & Sons, New York, U.S.A., 3rd edition.

[Goldberg and Riek, 2000] Goldberg, R. and Riek, L. (2000). *A practical handbook of speech coders.* CRC Press, Boca Raton, Florida, U.S.A.

[Gonzalez et al., 2004] Gonzalez, R., Woods, R., and Eddins, S. (2004). *Digital image processing using MATLAB®.* Prentice Hall (Pearson Education), Upper Saddle River, New Jersey, U.S.A.

[Gonzalez et al., 2009] Gonzalez, R., Woods, R., and Eddins, S. (2009). *Digital image processing using MATLAB®.* Gatesmark Publishing, Knoxville, Tennessee, U.S.A., 2nd edition.

[Haas and Burnham, 2008] Haas, O. and Burnham, K., editors (2008). *Intelligent and adaptive systems in medicine (Part II Artificial Intelligence Applied to Medicine).* CRC Press (Taylor & Francis), New York, U.S.A.

[Haddad and Parsons, 1991] Haddad, R. and Parsons, T. (1991). *Digital Signal Processing - theory, applications and hardware*, volume I. Computer Science Press, New York, U.S.A.

[Ingle and Proakis, 1997] Ingle, V. K. and Proakis, J. (1997). *Digital signal processing - using MATLAB® V.4.* PWS Publishing Company, Boston, MA, U.S.A.

[Jackson, 1996] Jackson, L. (1996). *Digital filters and signal processing.* Kluwer Academic Publishers, Norwell, MA, U.S.A., 3rd edition.

[Jong, 1982] Jong, M. (1982). *Methods of discrete signal and system analysis.* McGraw-Hill, New York, U.S.A.

[Kay, 1987] Kay, S. (1987). *Modern Spectral Estimation: Theory and Application.* Prentice-Hall, Englewood Cliffs, New Jersey, U.S.A.

[Makhoul, 1975] Makhoul, J. (1975). Linear prediction: A tutorial review. *Proc. IEEE*, 63:561–580.

[Marple, 1987] Marple, S. (1987). *Digital spectral analysis with applications.* Prentice Hall, New Jersey, U.S.A.

[Nixon and Alberto, 2002] Nixon, M. and Alberto, A. (2002). *Feature extraction and image processing.* Newnes (Elsevier), Burlington, Massachusetts, U.S.A.

[Oppenheim et al., 1999] Oppenheim, A., Schafer, R., and Buck, J. (1999). *Discrete-time signal processing.* Prentice-Hall International, London, U.K., 2nd edition.

[Otsu, 1979] Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:62–66.

[Pan et al., 2004] Pan, T., Lee, T., Rietzel, E., and Chen, G. (2004). 4D-CT imaging of a volume influenced by respiratory motion using multi slice CT. *Med. Phys.*, 34:333–340.

[Papamichalis, 1987] Papamichalis, P. (1987). *Practical approaches to speech coding.* Prentice Hall, Upper Saddle River, New Jersey, U.S.A.

[Proakis and Manolakis, 1989] Proakis, J. and Manolakis, D. (1989). *Introduction to digital signal processing.* Macmillan Publishing Company, New York, U.S.A.

[Quatieri, 2001] Quatieri, T. (2001). *Discrete-time speech signal processing - principles and practice.* Prentice-Hall, Upper Saddle River, New Jersey, U.S.A.

[Rabiner and Schafer, 1978] Rabiner, L. and Schafer, R. (1978). *Digital processing of speech signals.* Prentice-Hall, Upper Saddle River, New Jersey, U.S.A.

[Ramamurthy and Spanias, 2009] Ramamurthy, K. and Spanias, A. (2009). *Matlab software for the code excited linear prediction algorithm: the federal standard - 1016.* Morgan and Claypool publishers, San Rafael, California, U.S.A.

[Sims et al., 2009] Sims, R., Isambert, A., Gregoire, V., Bidault, F., Fresco, L., Sage, J., Mills, J., Bourhis, J., Lefkopoulos, D., Commowick, O., Benkebil, M., and Malandain, G. (2009). A pre-clinical assessment of an atlas-based automatic segmentation tool for the head and neck. *Radiotherapy and Oncology*, 93:474–478.

[Sonka et al., 1999] Sonka, M., Hlavac, V., and Boyle, R. (1999). *Image processing, analysis, and machine vision.* PWS Publishing, Pacific Grove, California, U.S.A., 2nd edition.

[Sonke et al., 2005] Sonke, J., Zijp, L., Remeijer, P., and van Herk, M. (2005). Respiratory correlated cone beam CT. *Med. Phys.*, 32:1176–1186.

[Stearns and Hush, 1990] Stearns, S. and Hush, D. (1990). *Digital signal analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, U.S.A., 2nd edition.

[Umbaugh, 2010] Umbaugh, S. (2010). *Digital image processing and analysis: human and computer vision applications with CVIPtools.* CRC Press (Taylor & Francis), Boca Raton, Florida, U.S.A., 2nd edition.

[Ziemer et al., 1998] Ziemer, R., Tranter, W., and Fannin, D. (1998). *Signals and systems - continuous and discrete.* Prentice Hall, Upper Saddle River, New Jersey, U.S.A.

[Zuiderveld, 1994] Zuiderveld, K. (1994). *Contrast limited adaptive histograph equalization.* Graphic Gems IV (Academic Press Professional), San Diego, California, U.S.A.

Useful web addresses for image processing

Some useful web sites can be found at:

```
http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm
http://www.fftw.org/links.html
http://www.imageprocessingbook.com
http://www.fftw.org/links.html
http://www.fftw.org/links.html
http://www.efg2.com/Lab/Library/ImageProcessing/Algorithms.htm#GeneralAlgorithms
http://www-2.cs.cmu.edu/~cil/vision.html
http://www-video.eecs.berkeley.edu/
```

Signal processing index

Image processing index