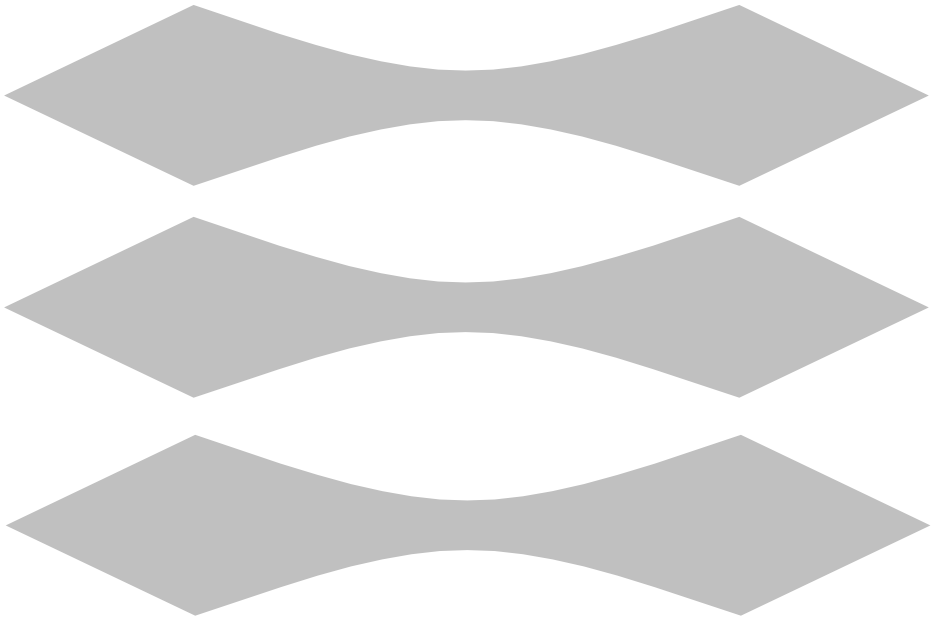

DIGITAL COMMUNICATION

34230, Lecture Notes, 2017

Knud J. Larsen & Jørn Justesen

DTU Fotonik
Department of Photonics Engineering

Section for Communication Technology
Technical University of Denmark
Building 343, DK-2800 Kgs. Lyngby

DTU

CONTENTS

LIST OF EXAMPLES

PREFACE

Many colleagues have contributed to the development of these notes over the years and comments and remarks from the students have also been included. We are thankful for all these contributions. It is not possible to mention each and everyone, but the present issue contains contributions from Søren Forchhammer, Lars Staalhagen, Jari Korhonen, and Metodi Yankov.

A very important part of the course is the exercises, which in a number of sessions take the reader through all subjects including many hands-on simulations and calculations with MATLAB in order to provide a more deep understanding of the theory. The text for each exercise is presented at the DTU Inside site for the course, and for most exercises there will also be solutions published after the exercise.

Many of the examples in the notes involve direct use of MATLAB, and we assume version 6.0 or later. Readers not familiar with MATLAB are advised to consult a book on this powerful programming language or at least the short notes available at the course. The examples are in general made as MATLAB scripts ("m-files"), but the program listings in the examples may be mixed with printouts of results, the results may be slightly reformatted, and not all details for plots and prints are shown in the program listing. It is intended to present all MATLAB programs and references to functions in the `courier` font. Variables are partly in this font or written as vectors/matrices in **bold**.

Literature references are marked as [k.n], which refers to a list at the end of each chapter. References internally in the notes are given as a chapter or section number, e.g. Section 3.2.3. References to equations in the notes are given as (k.n), where k is the number the chapter. To give the Danish students an account of the Danish technical terminology the notes indicate this as (*dansk terminologi*), i.e. italic font.

These notes are the eighteenth edition of notes for the course. In this edition some sections have been moved around and it may cause errors in addition to old errors not found yet. The authors will appreciate all comments and descriptions of errors found.

Knud J. Larsen and Jørn Justesen

August 2017

1. INTRODUCTION TO TELECOMMUNICATION AND COURSE

1.1 Introduction

Telecommunication(s) (*telekommunikation* or more traditionally *teleteknik*) is the heading for all activities involved in **transmission of information between users usually separated by more than normal visual or hearing distance**, ("tele" means distant).

In modern times, the transmission is physically done as electromagnetic signals, but often the physical appearance of the signals will be quite hidden behind mathematical and symbolic descriptions as it will be seen from this course. Note, that telecommunication is concerned with transport of **information** in contrast to more physical forms of transport, e.g. traditional mail. Information may appear in many ways as it will be seen from the next section and especially from Chapter 4: speech, music, images, data etc. The goal of the transmission from a technical and economical point of view is to provide this transmission as reliable as possible taking into account the costs of doing so. In modern times these considerations always result in **digital communication**, i.e. the transmitted signals are selected (randomly) from a finite set of known signals. In Section 1.4 we describe fundamental problems of communication and why digital communication is the only possible solution.

Since telecommunication is a very broad subject many different facets of the subject may be treated in courses like this. We shall narrow our point of view in the description of basic problem areas of telecommunication below. DTU offers relevant courses in almost all these areas.

- Transfer of signals over a certain, sometimes very large, distance. The media used for the transmission may be cables, optical fibers, or radio (e.g. via satellites). This subject, **transmission** (*transmissionsteknik*) is the main content of this course and it is further specialized in digital transmission as the only modern technique for communication.

- Connection of users who want to be connected. In principle all connections could be made between all users but a little thought will show that this is not feasible. Thus there is a need for special functions allowing users to share common parts of the communication path and then switch these parts in a way that each user could be connected to any other user. This subject is called **switching** (*koblingsteknik*) and for traditional telecommunication such as telephony, the subject has been much separated from the transmission subject. In more modern applications, switching is partly inherent in the transmission system, e.g. in data networks.
- **Information sources.** The information to be transmitted appears in many ways, e.g. speech, music, images, data etc. and traditionally separate means of transmission have been established for these such as the telephone network, broadcasting radio stations etc. In modern times all information sources are assumed to be converted to a digital format and the transmission systems become more universal. We shall address such conversion in this course (Chapter 4).
- **Interfaces** (*grænseflader*) between users and the telecommunication systems and internally in the systems. A technical description of an interface involves partly electrical characteristics such as voltages, impedances etc. and partly the logical formats and time behavior of the interface. The last part may be called a **protocol** (*protokol*) for the interface. As an example you may think of a telephone connection where initialization of a call has to be done following a certain procedure. In general, this is outside the scope of this course, but examples and details may be found in [1.1]. However, we shall briefly introduce the subject in Section 1.4.1. The protocol may also include some kind of safeguarding messages against un-allowed reading or imitation of e.g. signatures. This is the subject of various courses in **cryptology** (*kryptologi*) and **data security** (*datasikkerhed*). Another aspect of the interface concept is the so-called man-machine interface.
- **Implementation** of the components used in telecommunication networks. The technologies reach from optoelectronics, optical fiber technology through plain electronics to software technologies.
- **Planning and maintenance**. Technical aspects of this problem area are such as teletraffic engineering, network planning, and network management.

- **Market and organization.** We shall briefly introduce some organizational issues in Section 1.3, but we shall not go into any details of general planning and commercial issues.

A telecommunication system provides a **service** (*(tele)tjeneste*) to the user. The system may be public or private. Most examples in this course will be drawn from public systems since these are assumed to be best known but it should be emphasized that the techniques to be described are used in private systems as well. Some examples of private systems are: Corporate data networks, systems for rail or road traffic control, and transmission systems for scientific satellites. In the next section we shall provide examples of public services. In general, a service may be characterized by

- type of information exchanged
- the character of each connection: **direct**, also known as **circuit switched** (*direkte (kredsløbs)koblet*) or **indirect**, **packet switched** (*pakkekoblet*). Subgroups of these character groups exist, but we shall not go into further details.
- number of communicating parties and their role in the communication (transmitter and/or receiver, (*sender, modtager*)) as illustrated in Figure 1.1.

When a number of users is using the same service we describe the connections between them as a **telecommunication network** (*telenet*). A network consists of a number of connected nodes where communication flows along the connections. In most networks, nodes of two types exist: **terminal nodes** placed at the users and **switching nodes** establishing the connections through a shared part of the network (*opkobling*) since direct physical connections for all possible logical connections are very seldom found. From time to time we encompass all public telecommunication networks in one name, the (world-wide) telecommunication network (*telenettet*) which has at least 10^9 terminal nodes. Very comprehensive descriptions of telecommunication networks are given in [1.1, 1.2].

In the next section we give some examples of public services and we continue with a description of the organizational structure of public telecommunication (primarily in Denmark), and organizations for standardization which is very important due to the very large extent of the world-wide telecommunication network.

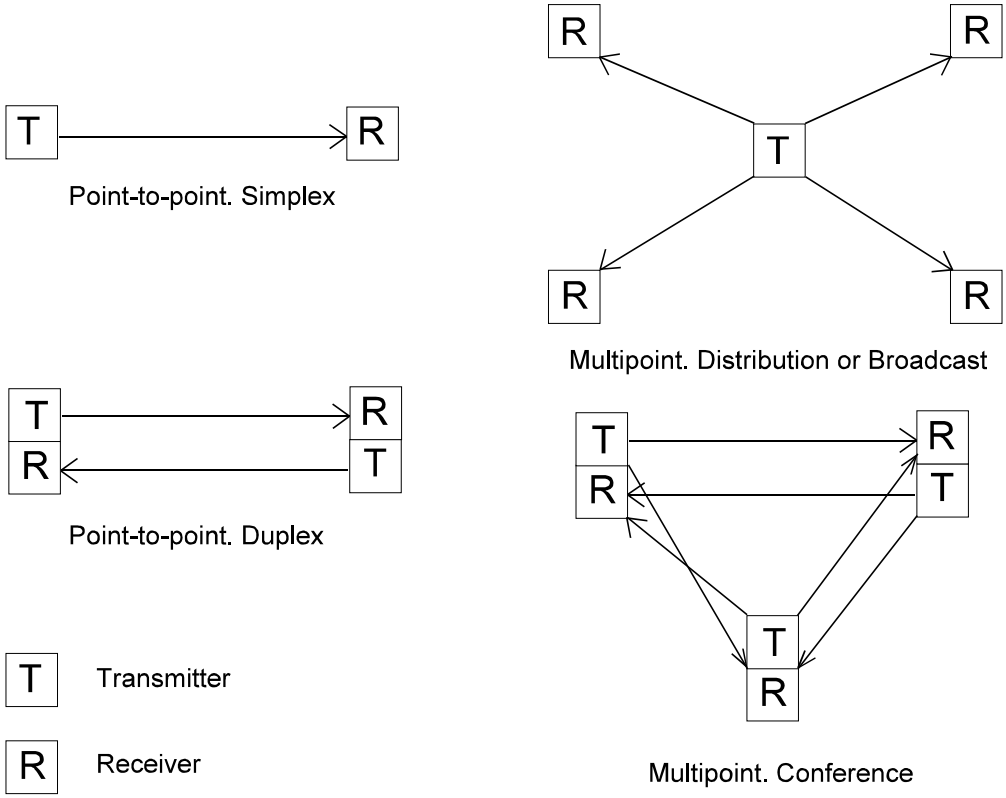

Figure 1.1
Communication structures.

After this description we introduce, in Section 1.4, the fundamental problems involved in transmission of information from one point to another and thereby we give an introduction to the present course which is almost solely devoted to that issue. Throughout the course we shall use some implementation examples and two of these are introduced in Section 1.5.

1.2 Public telecommunication services

The telecommunication network offers many different **services** (*(tele)tjenester)* for the users, and only a very brief description of each is possible. Most of the services are offered by a number of **operators** (*operatører*) to the users, the **subscribers** (*abonnenter*), that subscribe to the service. We have chosen to describe some services offered in Denmark, but the services in comparable countries are equal although the explicit names may be different. The list is not complete, but it shows the plentitude.

Telephony (*telefoni*), often denoted **PSTN**, the Public Switched Telephone Network. A slightly more humorous name is POTS, Plain Old Telephone Service. The service provides a switched connection (a circuit) between two users capable of duplex transmission of speech, i.e. analog signals in the frequency range 300 - 3400 Hz. The users are identified with telephone numbers and after the connection is established, the so-called circuit switched connection is found. Although the main application is for speech, other applications have also appeared: **facsimile, fax** (*faksimile, fax*) for transmission of documents, "telecopying", and **data transmission** through speech connections with use of modems. The switching is performed in a number of **telephone exchanges** which are arranged in a hierarchical structure covering the entire world. The hierarchical structure is further explained in [1.1].

As it will be known, the access equipment to the PSTN service is normally a **(tele)phone** (*telefon(apparat)*), but another possibility is **wireless** or **cordless telephony** (*trådløs telefoni*). By means of base stations, i.e. small radio stations, handheld terminals are coupled to the telephony service when rather close to the base station (< 0.5 km). The most recent system is the digital standard from 1992, **DECT**, Digital Enhanced Cordless Telecommunications, which allows more terminals for each base station and may be used for data transmission as well. Since DECT will serve as an example several times in this course, a more comprehensive treatment is given in Section 1.5.

ISDN **Integrated Services Digital Network**. The basic service provides access to other services and some special ISDN facilities [1.1]. The users are connected to the network via a digital connection providing 2·64 kbit/s, which may be used for telephony or data (circuit switched) together with 16 kbit/s for packet switched data. Since we shall use this connection as an example several times in this course we give a more comprehensive description in Section 1.5. In view of the limited data rate, **xDSL** (mostly **ADSL**, see Chapter 5) has replaced ISDN as a data access method, e.g. in connection to the Internet.

Telegraphy (*telegrafi*), often known as **TELEX** or teleprinter service. Exchange of telegrams in a special network. There is a circuit switched connection between the two terminals and the transmitted information consists of characters (alphabet of 59 characters). TELEX is now mostly superseded by other communication means, e.g. e-mail, but still used in maritime applications (via radio).

X.25 Packet switched data transmission [1.3 (X.25), 1.1]. The data are transmitted as telegrams (packets) without direct connection between transmitter and receiver. The interface is defined in the recommendation X.28. The service is now mostly replaced with internet protocols.

Internet Communication with information providers from a computer, e.g. a PC. At the first stage, the connection is plain data transmission between the user and an **Internet Service Provider** of which a large number is found, [1.4]. This first stage uses a modem on an ordinary telephone line, a dedicated line, xDSL (mostly ADSL), a leased line, or a modem for a CATV system (introduced later). From the connection point for the service provider to the information provider the transmission uses a purely digital network, the **Internet**, which is packet switched in much (but not exactly, see [1.1]) the same way as X.25 mentioned above. The users are addressed by internet addresses which are 32-bit numbers, e.g. 10.51.32.133, which may be a translation of a more user friendly name, e.g. ftnk0427.win.dtu.dk. Internet access also allows **electronic mail**, **e-mail**, (*elektronisk post*) as it will be known.

Leased lines (*faste kredsløb*). Leased lines are defined as telecommunications facilities providing transparent transmission capacity between network termination points. Leased lines are found both as analog connections (corresponding to telephony connections described above) and digital connections with 2048 kbit/s or even greater capacity. We shall return to bit rates and formats in Chapter 8.

Video conference (*videomøde*) Video conferences with sound and video can be arranged between more than two parties (conference, cf. Figure 1.1). Standardized by ITU-T in H.32x-series. As transmission service, the above mentioned ISDN, digital leased lines or packet networks may be used.

Radio and video distribution (*radio*- and *TV-distribution*) Connections between information providers (**broadcasters**) and terrestial transmitters or directly to the users via cable, **CATV, Community Antenna TeleVision** (*fællesantenneanlæg*). Traditionally the transmission technique here has been analog, but new digital systems **Digital Audio Broadcast, DAB** and **Digital Video Broadcast, DVB** are now in operation. DVB is actually also the name of the system used in Europe and other continents. I North America, South America, Japan, and China similar, but different systems are found. Most countries have terminated the terrestrial analog TV distribution network, but analog TV is still found in some CATV installations.

All the services described above are using the **fixed (telecommunication) network** (*fastnettet*) but services with more mobility are also found as described below. All these services work on **wireless networks** (*trådløse net*) by nature, but each of these networks has a considerable fixed infrastructure so it is mostly the access part, i.e. the last link to the users, which is wireless.

Mobile telephony (*Mobiltelefoni*). Telephony using radio between mobile terminals and **base stations** (*basisstationer*). Each base station communicates with the mobile terminals within a so-called **cell** (*celle)*. Together the cells cover the area within which communication is possible. The location of the mobile terminals (i.e. which cell) is detected automatically by the system. Sometimes mobile telephony of this type is named **cellular telephony** to distinguish it from old systems without automatic location detection and no automatic transfer between the cells, the so-called **handover**. Terrestrial and satellite (see below) based networks are found.

Examples of terrestrial mobile telephone networks are: The nordic mobile telephone network, **NMT** which provides analog telephony, and the digital Global System for Mobile Communication, **GSM**. NMT was the first mobile network to span more countries, but it is now closed in Denmark. GSM spans at least 80 countries around the world. It also offers data transmission services resulting in a quite complete **mobile communication system**.

GSM is found in three versions: 900 MHz, the original GSM, GSM1800 around 1800 MHz, and an American version around 1900 MHz. A 450 MHz version may also be introduced. Much more details about GSM systems are found in [1.1]. A special version, **GSM-R**, is intended for communication between trains and train traffic centers replacing among other things the visual communication between train drivers and signals along the track, [1.5]. GSM-R is now installed in Denmark as part of a new railway signaling system. The standard frequency range is 876-880 MHz (uplink) and 921-925 MHz for the downlink. A third generation of mobile communication systems, **UMTS**, the Universal Mobile Telecommunication Systems, is also in operation offering much higher data rates for the users (HSDPA) which has also let some subscribers to use this as connection to the Internet instead of the fixed access methods described above, [1.1]. The frequency range is around 2 GHz. Recently, a fourth generation, **LTE**, Long Term Evolution, has been installed, [1.1]. Several frequency bands are in use, from 700 MHz to 2600 MHz.

In 2006 the total number of radio cells in Denmark was 24819 [1.4] so a significant fixed part is also required for a mobile communication network.

Satellite telephony

Mobile telephony with satellite base stations is offered by several operators (for overviews, consult [1.6]). An example is **Inmarsat** operating 11 satellites in geo-stationary orbit. The main cells are very large, e.g. the Atlantic Ocean including the eastern part of North America, South America, and Europe. In addition to the large cells, there are also 19 more regional cells which are less demanding for the terminals with respect to antenna and power and around 200 spot beam cells (few hundred km wide). Several services are offered, the most advanced being the BGAN - Broadband Global Area

Network. Many of the older services are for speech or data communication with a moderate rate. BGAN is for more universal usage with data rates up to 492 kbit/s. An even more capable service, BGAN HDR, provides up to 800 kbit/s (one terminal, two may be bonded giving more than 1 Mbit/s).

The services are offered to land mobile, maritime and aeronautical markets - partly with different names. In 2014, about 340 000 terminals for Inmarsat were found [1.7] and the number is increasing. Another example of a satellite system is **Iridium** which uses 66 low-earth orbiting satellites and smaller cells (48·66 = 3168 cells) than the Inmarsat systems, [1.8]

Paging (*personsøgning*). Delivery of short messages to portable receivers (number codes or even text messages for the more advanced receivers). Paging systems are now ousted as a public service by the much more popular mobile communication which by its nature involves location of the subscribers and even provides a text message service, **SMS**.

Localization There is a considerable demand for services that give the user a position information, and maybe offer him or her some information dependent on the position. The first application was navigation, and the **GPS**, Global Positioning System, is the most recent system for providing information on the exact position. It was introduced by the United States originally for military application, but it has found many civil applications as well, e.g. navigation instrument for leisure boats or cars. The positioning is based on measuring the time delay for radio signals to a number of satellites whose positions are well known. Currently a competing European system is being developed. Positioning may also be done using mobile phones which normally are within reach of several base stations at the same time allowing the position to be determined.

Hotspots Wireless networks (**WLAN**) have become very popular for connection of computers instead of wired local area networks, LAN. The term “hotspot” is a popular name for the wireless network access point. Basicly, most standards in this field originate in the American Institute of Electrical and Electronic Engineers, IEEE, whose wireless standards are known as 802.11x, where x is a letter indicating different rates of communication and frequency bands

used. Equipment used for these standards are often said to be **WiFi** certified. In recent years, hotspots have developed into a service provided at places where many people meet, e.g. restaurants, gas stations, airports, campings, or in public transportation. The service is not very universal since it is offered by various companies, and you subscribe to the service of the specific company which then provides access to the Internet. However, the subscription is often free of charge.

An overview of these and many other wireless networks is given in [1.9].

The size of the different services may be stated in different ways which may need very many definitions and complicate the presentation here. A simple way is to use the number of subscriber lines (*abonnentlinieantallet*), and [1.4] gives such numbers. Telephony (fixed or mobile) is still a dominating application counting the number of subscriber lines. This is also the case world-wide where more than 10^9 subscribers are found. The number of internet subscribers is believed to be of the same order. Estimating the traffic, e.g. in bits/s, is more difficult.

1.3 Organization and standardization

A service is supplied for the users by an operator who sees to getting paid for use of the service. In most countries, e.g. Denmark, a **license** (*tilladelse*) from a public authority is required for an operator. In Denmark the control of operators is performed by the **Danish Business Authority** (*Erhvervsstyrelsen*) which also grants the licenses. In former times, a license was a monopoly, a so-called **concession** (*koncession*), but now competing operators are found for many services. In Denmark, all concessions were terminated in July 1996, and a free opportunity to apply for a license was introduced by the so-called **liberalization** (*liberaliseringen*) and now many operators are found, some with their own network and some using already established networks. The process of controlling the telecommunication operator market is named **regulation** (*regulering*).

An important aspect of regulation is to distribute resources that are common and limited. Several such resources exist in telecommunication, e.g. telephone numbers and radio frequencies. The Danish Business Authority is responsible for the administration of the

numbering area in Denmark, i.e. allocation of numbering resources, e.g. allocation of the national numbering plan (mostly 8-digit numbers) and other numbering plans of interest to Danish operators. With respect to frequencies, the Danish Business Authority is responsible for the administration of the Danish frequency allocation table and provision of licenses to use a frequency (band).

Since telecommunication involves a great number of participants there is a very large need for standardization, both nationally and internationally. In hardly no other subarea within the electrotechnical area, the use and importance of standards are so pronounced which will also be seen in the present course. Standards for Denmark follow common international standards.

Internationally the telecommunication standardization is performed in a United Nations organization, **ITU**, **International Telecommunication Union**. Before 1993 the work here was divided between two organizations: **CCIR** for radio communication and **CCITT** (French acronym for "Comité Consultatif International Télégraphique et Téléphonique") for cable based communication. Every fourth year, CCITT and CCIR published a complete set of standards, named **recommendations**, and each new edition was characterized with a new color of the binding. The last issue of the CCITT recommendations has a blue binding, [1.3]. The two organizations are now merged into one, **TSS, Telecommunication Standardization Sector** which performs its work in 15 **Study Groups**. Each of these groups is authorized to publish recommendations when they are ready, i.e. without the previous periodical publication times. Even if the name recommendation suggests that it is only something being recommended they will in practice act as standards. In this course we shall refer to the TSS-recommendations as **ITU-T** no matter what the age of the recommendation is. All ITU-T recommendations are named as "letter.number", e.g. G.703 or V.90, and thus they are easily recognized.

In the eighties there were much criticism of the very slow ITU-T/CCITT procedure, and in Europe this lead to the formation (1988) of **ETSI**, **European Telecommunications Standards Institute**. The purpose of ETSI is to provide technical standards for use in a common European telecommunication market. ETSI has produced such important standards as GSM and DECT. The institute is authorized by the EU-commission as the

European standardization organization for telecommunication. To act as such, the national standardization organizations (NSO) such as the Danish Business Authority participate in the organization and especially in the decisions about standards. In addition to the national telecom agencies, the members of ETSI are: Operators of public services, manufacturers of telecommunication equipment, user organizations, and research institutions within the area. The work in ETSI results in documents of two types:

- **ETS**, **European Telecommunication Standard**. In these standards, functions in a given system are described, e.g. ETS 300 142: "ISDN and other digital telecommunications networks; Line transmission of non-telephone signals; Video codec for audiovisual services at p x 64 kbits [ITU-T Recommendation H.261 (1993), modified]".
- **TBR**, **Technical Basis for Regulation**. A TBR provides a basis for the decision by the EU-commission about a **CTR**, **Common Technical Regulation**, which regulates equipment that is allowed to be used within EU (slightly extended), e.g. CTR 3: "ISDN Basic Access". When equipment is allowed in one of the countries with respect to a CTR it is marked with **CE** (*CE-mærke*) (e.g. CE00170) and it may then be used in all EU-countries.

Another interesting organization is **ISO**, **International Standards Organisation** which also performs standardization work of relevance to telecommunication. The standardization in ISO is mostly concerned with equipment and reaches from very simple issues such as a modem connector to rather overall issues in connection with software for communication between computers. A famous contribution from ISO is the ISO-OSI protocol reference model treated in literature, e.g. [1.1]. A short introduction is given below in Section 1.4.1.

1.4 Fundamental problems in transmission of information

Figure 1.2
Basic model for transmission

In Figure 1.2 we have shown a very basic model for transmission of information in order to pinpoint where the fundamental problems are found. The model consists of the **transmitter** (*sender*), the **transmission channel** (*transmissionskanal*), and the **receiver** (*modtager*).

First, it is important to describe the transmission channel used. We shall give detailed descriptions of four kinds of channels in Chapter 7: Metallic transmission lines, optical transmission lines, radio channels, and storage media. The last item is not a transmission system as such, but since there are close relations to the techniques used for storing information in magnetic or optical media we shall include this in the channel descriptions as well.

The channel descriptions use a number of models, and the first description as a linear, time-invariant system uses the methods of deterministic signal analysis introduced in courses about signals and linear systems [1.10]. The deterministic signal analysis is also important for a number of components in the transmitter and the receiver. In the present course we have devoted Chapter 2 to an introduction to notation and to the main tools used from the prerequisite courses. The main tool is of course transforms between time and frequency domains, and we have focused on discrete time signals which are often used in implementation of transmission systems.

Deterministic descriptions are not sufficient for telecommunication systems for two reasons:

- Information is characterized by surprise: If we know in advance what the information will be, there is no need to transmit it.

- The physical transmission channels are not deterministic. They may change in an unpredictable way (as some radio channels do) and most important: signals from outside the system interfere with the transmitted signal. Although some interference is the result of a purely deterministic mechanism, the exact result of the interference may be dependent on random information in other transmission systems or other systems with a random behavior. We name these foreign signals **noise** (*støj*), and the **main problem of communication is that of doing it in the presence of noise**. We shall characterize different forms of noise later in the course.

Thus we need methods from probability theory, i.e. the so-called **stochastic processes**, to describe the behavior of transmission systems, and these are introduced in Chapter 3. Like the deterministic description we shall focus on time discrete signals. These chapters also deal with processing of stochastic signals in linear time invariant systems. Chapter 3 ends with a description of sampling that converts time-continuous signals into time-discrete signals.

Although the course is mostly focused on description of the transmission system shown in Figure 1.2, a more thorough description of various information sources is also needed. In Chapter 4 we shall describe various signals such as speech, audio and video signals. These are all inherently **analog signals**, i.e. time-continuous signals which may assume any value in a continuous signal range and we shall describe the processing (sampling and quantization) that converts such analog information signals into **digital signals**, i.e. time-discrete signals with a limited number of values. We shall also treat ways of improving on the immediate conversion methods. The main purpose of the improvement is reduction of the number of digital symbols produced in order to cope with capacity limits of the transmission systems. The main tool for this improvement is use of predictive methods which based on statistical properties give a fair prediction of the next sample of the process. The tools introduced may also have other usage than just prediction of signal samples.

The first use of electromagnetic signals for communication was the optical telegraphs around the beginning of the 19th century and then later the electrical telegraph. The electrical telegraph and telegraphy was presented in 1838 by S. Morse [1.11], and in modern terms, the signals were digital. From the invention of the telephone in 1876 (by A.

G. Bell) and for about the next 100 years, the information was expressed directly as analog signals and they were transmitted as such. All signals are of course always "analog" during the transmission as we shall demonstrate later, but the main difference is that digital signals have a finite number of basic waveforms, and the receiver estimates the most likely signal in this finite set. This may possibly result in an error which is a clearly defined event in contrast to analog signals where the transmission may give small or large deviations, and the quality of the received signal is not that precisely defined. In almost any kind of system today, the transmission is based on digital signals. In the present course we shall deal almost exclusively with transmission and reception of digital signals (corrupted by noise). We start out, in Chapter 5, by treating signals that are transmitted with only minor changes in the frequency band occupied, the so-called **baseband transmission**. In Chapter 6 we shall treat transmission where the character of the transmission channel dictates that a shift of the frequency band is necessary. This shift, **modulation**, is needed in radio transmission and some cable based systems as well. In the present course we shall not deal very much with theoretical **limitations of the capacity** of a transmission channel, but we would like to point out that such a limitation exists, and it is dependent on the bandwidth available and the ratio between the transmitted signal power and the noise power, the signal-to-noise ratio, Section 5.4 and [1.12]. More bandwidth and a higher signal-to-noise ratio allows more information to be transmitted, but these resources are restricted for many reasons.

This course is mostly concerned with the realization of communication links at the physical level including all the necessary signal processing. In the practical realization many physical links are used to create a **telecommunication network** (*telenet*). One interesting issue in a network is how to share transmission systems between different information sources, and in Chapter 8 we shall discuss this. Basically two main concepts may be identified:

- **multiplexing** (*multipleksning*) where the transmission system is controlled by some central entry point gathering the information from the different contributors and formatting them in a way to allow transmission of these independently through the shared system.
- **multiple access** where the information contributors independently control their access to a shared transmission media following some algorithm also known as an

access protocol. We shall deal with such procedures, protocols, in a more general sense below. In Chapter 8 we shall provide some examples of access methods.

It is not always possible to put a given system into one of these categories since some mixture may be applied, but the categorization is important for the understanding of the access mechanisms. The usage of the terms is also not very stringent.

1.4.1 Protocols and reference models

In the basic model of transmission shown in Figure 1.2 and in the stating of fundamental problems following that, we have mostly focused on the signals exchanged, i.e. physical signals representing some information, and the present course shall deal almost exclusively with this aspect with small excursions to e.g. how these signals may be formatted into blocks (frames) of data. This model overlooks another important aspect of communication, the timely behavior of the communicating parties or processes. The main model for description of communicating processes is that of defining a **protocol** (*protokol*) which describes how to proceed when exchanging information, i.e. what is communicated, how it is communicated and when it is communicated. A protocol could be said to consist of three basic elements: **Syntax** (structure of the information, data format, coding, physical signal representation), **Semantics** (meaning of information exchanged including control information for coordination and error handling) and **Timings** (times at which data should be transmitted or looked for by the receiver, sequencing of information, e.g. how to establish a connection and so on). In Chapter 8 we shall provide some examples of access methods, but as seen e.g. in [1.1, 1.13], protocols are also used for description of communication at levels above the access level. The concept of a protocol is rather general, and in fact most communication uses a protocol, e.g. buying a soft drink from a vending machine requires a protocol: selecting the specific bottle, inserting coins, taking the bottle, taking coins returned, and maybe much more.

In the description of communicating systems it is often used to divide the functionality of the systems into a number of independent subfunctions that may be specified and analyzed independently. ISO and ITU-T have standardized a model for this: the **OSI reference model** which means Open Systems Interconnection Reference Model. In this model the functionality is distributed into 7 **layers** (*lag*) as shown in Figure 1.3. If we consider a particular pair of receiver and transmitter, each of them with its functionality divided into

layers, then each layer in one of them communicates with the corresponding layer in the other − **peer-to-peer communication** − using a specific protocol.

The physical layer (*fysisk lag*) is concerned with transmission of an (unstructured) bit stream over some physical medium. It deals with the mechanical, electrical, functional and procedural characteristics to access the physical medium. The present course deals almost exclusively with the physical layer. The **data link layer** (*datalænkelag*) provides for the reliable transfer of information across the physical link. It sends blocks of data (frames) with the necessary synchronization, error control and flow control. The **network layer** (*netværkslag*) is responsible for establishing, maintaining, and terminating connections which may consist of a number of data links. Thus is makes the upper layers independent of the transmission and switching technologies used in the network. The **transport layer** (*transportlag*) provides reliable, transparent transfer of data between end-points. It provides end-to-end error recovery and flow control. The **session layer** (*sessionslag*) provides the control structure for communication between applications. It establishes, manages, and terminates connections (sessions) between cooperating applications. The **presentation layer** (*præsentationslag*) provides independence to the application processes from differences in data representation (syntax), i.e. the functionality for conversion of information between different data formats is contained here. The last layer is the **application layer** (*applikationslag*) which provides the users' access to the transfer of information using all the six layers below. Other ways of structuring the functionalities may also be found, but there are many similarities to e.g. the **TCP/IP model** used for the Internet. A detailed description of the OSI reference model and the TCP/IP model is given in [1.13]. Not all implementations follow the layering exactly since the performance may in some cases be impeded by the layering.

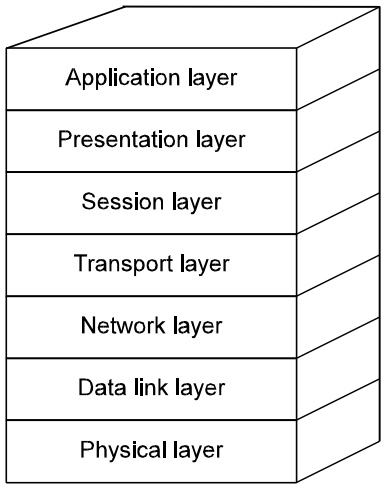

Figure 1.3
OSI reference model.

1.5 Introduction to the examples ISDN and DECT

Formerly, each of the traditional public services used a separate physical network containing switching equipment and separate transmission connections. Thus it is clear that it will be more economical to have an integrated network at least for the fixed part. This consideration leads to the idea of **ISDN, Integrated Services Digital Network**. The original idea from around 1980 is partly outdated by the development of various components allowing easy sharing of high speed transmission lines (to be described further in Chapter 8). However there is still some interest in the transmission capacity offered to the user which is about 4 times (Basic Rate Interface, BRI) the capacity offered with traditional modems, and certain special facilities of the ISDN service are also of interest, [1.1]. Here we shall present the basic system which offers 2·64 kbit/s circuit switched (the so-called **B-channels** (*B-kanaler*)) and 16 kbit/s packet switched (**D-channel** (*D-kanal*)). This increased capacity is obtained without any new cabling to the user, the original subscriber line for analog telephony is used. The only component that has to be installed is a so-called **NT, net termination unit** containing a transmitter and receiver.

The NT transmits 160 kbit/s in full duplex using the original **subscriber line** (*abonnentledning*) which may be up to 7 km long. These subscriber lines were previously used for the traditional telephony, i.e. analog signals in the frequency range 300 - 3400 Hz. The other end of the transmission line has a termination unit similar to the NT, and the communication channels are passed on to ISDN exchanges where the switching is performed. In this course, especially in the end of Chapter 5, we shall return to the particular transmission used at the subscriber line where most of the techniques introduced in this course are applied.

At the user's premises, the NT is connected to a transmission line that passes all the user's terminals, e.g. telephone, fax machine, or PCs. This line acts as a bus since it is shared by all the terminals and it is named the **S-bus**. In Chapter 8 we shall return to the explicit format of the transmission on the S-bus, but in principle it distributes the two B-channels and the D-channel to all the terminals. Which terminal that actually uses the channels to transmit and/or receive data is determined by the setup of the connection between the terminal and the ISDN service. This setup is controlled by sending special data messages on the D-channel and if a connection is established, one or both of the B-channels are

typically reserved for that communication. The D-channel is also reserved for a short moment, but an algorithm exists to ensure that all terminals are served by the shared D-channel. This is also the case when a terminal uses the D-channel for other transmissions (data messages) than the setup messages.

Larger transmission capacity is provided on subscriber lines using the **ADSL** and related techniques (**xDSL** in general, briefly described in Chapters 5 and 6), and in the future, optical subscriber lines may be installed giving much higher data rates. However, the ISDN service works well to demonstrate a lot of concepts in this course.

DECT, **Digital Enhanced Cordless Telecommunications**, will serve as an example several times in this course, and we shall give a short introduction in this section. The introduction is based on the ETSI Standard ETS 300 175. DECT was developed by ETSI in the period from 1988 to 1992 to provide a common European standard for cordless telephony and more general, cordless telecommunication. Like mobile telephony described in Section 1.2, the area covered is divided into a number of cells, characterized as picocells since they are very small (indoors radius up to 60 m, outdoors maybe 500 m). In each picocell the base station may communicate with up to 120 mobile terminals, handsets. The mobile terminals can communicate with each other, but the main purpose of DECT is to serve as an access technology to other services, mainly PSTN or ISDN.

In Europe, DECT employs 10 frequency bands with the center frequencies (1897344 − k·1728) kHz, k = 0, ..., 9. In other parts of the world, the frequency range may be a little different. The transmission is digital, and in Chapter 6 we shall discuss how a digital signal is transmitted over radio, but for the present purpose it is enough to think of a stream of bits following each other. For each frequency the transmitted stream is divided into a number of timeslots of which 24 are found each 10 ms. Each timeslot contains 480 bits of which 320 bits are user data. 12 timeslots are used for transmission down to the handsets and 12 are used in the opposite direction yielding the total of 120 duplex communication channels each with a bit rate of 32 kbit/s. In Section 4.6.3 we shall describe how speech is converted to a digital signal with 32 kbit/s. The quality of the speech signal is superior to e.g. GSM speech quality.

The power emitted from a portable handset is very small, 10 mW, which may be compared with the 2 W possible from a GSM handset. This means that the cells are very small, picocells, as described above. If more cells are covering a given area, the handsets are able to detect the best cell and force the base station system to move the communication to another cell, a handover. In the same way, different DECT systems may co-exist in overlapping areas, since the handsets are able to recognize their own base station and leave out communication at frequencies that are occupied by other systems or too noisy for other reasons. This facility completely removes the task of planning the use of frequencies, which is a hard task in e.g. GSM. The planning of a DECT system merely consists of setting up enough base stations to cover the area completely. It is not always that easy since the high frequencies employed are very influenced by building structures etc. Recently, ETSI has published the technical specifications (TS 102 939) for a low power consumption version of DECT (DECT ULE (Ultra Low Energy)), intended for home automation and other Machine-to-Machine (M2M) applications.

DECT may be employed as an ordinary cordless telephone, i.e. as a replacement for the usual phone connected to PSTN via a cord. If more DECT handsets are found they can communicate with each other and they may operate several usual telephone lines from one base station. This application is the dominant application for **domestic use**, and small DECT systems with one base station able to use 6 to 8 handsets are found in almost any telecommunication shop. For **professional use**, a company or organization normally operates a larger system with many base stations and the DECT system will be coupled to or integrated in the company telephone exchange, the PABX, allowing calls to reach the staff at any place and eliminating the need for wiring etc. in the case of moving around. In addition to the staff, guests may also be served by the system, if they invoke redirection of calls to the hosting company and there obtain a local extension number. In contrast to a mobile telephone system, a DECT system does not locate the terminals. These have to do their own action to be recognized by the system.

The DECT technology may also be used as replacement for the normal telephone cables connecting the subscribers in an area to the telephone exchange. This so-called **wireless local loop** has been used in some countries for a fast development of a network.

1.6 References for Chapter 1

1.1 Lars Staalhagen, ”Introduction to Communication Networks and Protocols”, August 2014.
(Used in the DTU Course 34311 and the previous course 34310).

1.2 Roger. L. Freeman: “Telecommunication System Engineering, Third Edition”, John Wiley and Sons 1996, ISBN 0-471-13302-7.

1.3 CCITT (ITU-T): "Recommendations from IXth Plenary Assembly, Melbourne 1988, Blue Book", International Telecommunication Union, Geneva 1989, ISBN 92-61-03xxx-x.

1.4 Energistyrelsen: "Telestatistik", July 2017.
Accessible via https://ens.dk/ansvarsomraader/telepolitik/tal-paa-teleomraaadet.
(mostly in Danish with some English translation). Some older versions are also used in these notes.

1.5 https://en.wikipedia.org/wiki/GSM-R, July 2015

1.6 http://en.wikipedia.org/wiki/satellite_phone, July 2015
http://personal.ee.surrey.ac.uk/Personal/L.Wood/constellations/overview, July 2015. (The website seems not to have been updated since 2006).

1.7 http://www.inmarsat.com, July 2015.

1.8 http://www.iridium.com, July 2015.

1.9 http://en.wikipedia.org/wiki/Comparison_of_wireless_data_standards, July 2015.

1.10 B.P. Lathi: “Signal Processing and Linear Systems”. ISBN 0-19-521917-1, Oxford University Press N.Y., 1998.
(Used in the DTU courses 31605 and 31606: Signals and Linear Systems)

1.11 "Telecom at 150", IEEE Communications Magazine, Vol. 27 No. 8, August 1989, pp. 6-18.

1.12 J.G. Proakis & M. Salehi: “Communication Systems Engineering. (2nd Edition)”, Prentice Hall, 2002, ISBN 0-13-095007-6.

1.13 W. Stallings: “Data and Computer Communications, Sixth Edition”, Prentice Hall International Inc. 2000, ISBN 0-13-086388-2.

2. DETERMINISTIC SIGNALS

In this chapter we shall state some basic results from signals, linear systems, and the Fourier transform [2.1]. We shall always emphasize a communication perspective for the treatment of signals as it is done in [2.2].

2.1 Basic concepts

In this section we shall summarize some basic properties of signals known from the prerequisites for this course, [2.1]. The listing is only to be seen as a summary - precise definitions and examples have to be found in the prerequisite literature.

- Continuous time / **discrete time signals** (*tidsdiskrete signaler*)
  To be discussed further below.
- Real / complex
  Until Chapter 6, all signals in this course have real values
- Deterministic / random signals
  A **deterministic signal** has defined values at all times whereas a random signal (also known as a **stochastic signal**) may be seen as a collection of random variables, i.e. each value is defined by a probability density function. In Chapter 3 we discuss stochastic signals.
- Periodic / Nonperiodic
  A periodic signal repeats itself after some time, the period.
- Energy and power type
  To be discussed in Section 2.4.

Signals are often sampled and processed in real time whenever that is possible. In other cases discrete time processing is used in simulations of continuous time systems. We shall not discuss continuous time signals in detail in these notes, but rather give a set of relations for signals and systems that relate explicitly to discrete time. Thus a signal, f, may always

be assumed to be discrete time unless otherwise stated. Sometimes we emphasize the discrete time by writing

$$f_k, \mathbf{f}, \text{ or } \{f_k\}$$

which all mean a signal with values

$$..., f_{-2}, f_{-1}, f_0, f_1, f_2, ...$$

occuring at times

$$..., -2T, -T, 0, T, 2T, ...$$

where T is the **sampling time**[1], We shall always include the sampling time T in the expressions, partly to relate the results to absolute frequencies (rather than frequencies normalized by the **sampling frequency** (*samplingfrekvens*, also known as sampling rate, *samplinghastighed*), 1/T, as it is often done in literature), partly to keep the proper physical dimensions. The physical dimension of f may be anything, but it is most common to think of f as being a **voltage** (*spænding*) or some function of a voltage (e.g. squaring).

In this course, we shall use MATLAB to simulate processing and generation of signals. Signals are represented in MATLAB as vectors, e.g. `f = [1, 2, 3, 3, 2, 1]`, and the time dependence is implicit such that the elements are understood to have time distance T. The exact time must be given somewhere in the context. Vectors are of course of finite length so the length must be chosen to encompass all relevant parts of the signal. On the other hand, MATLAB does not have problems with handling millions of elements in a vector. For a periodic signal, one period is usually enough. To simulate the progress of time found in most communication systems where signals are flowing through a system, one must move the indices of the vectors instead.

We shall assume that the properties of continuous time signals are sufficiently accurately represented by sampled signals with a high enough sampling rate. We shall make this more precise below in our treatment of continuous time signals.

---

[1] Denoting the sampling time *T* follows common practice in the literature (e.g.[2.1]), but sometimes T_s and ΔT are used. However, later in the course we shall use T for another purpose and then denote the sampling time T_s.

2.2 Transforms

As it may be known, a signal may be characterized in two "domains", the **time domain** and the **frequency domain** and the choice of these representations depends on the actual problem to be solved as it will be seen throughout this course. The important concept is that of a frequency. In many cases we shall express the frequency in radians per second, $\omega = 2\pi f$, where f is the frequency in hertz (Hz). In applications and in some of the problems, the unit is often Hz.

It is important to be able to relate the two representations and this is done via transforms and these are treated in this section. We shall assume some familiarity with **Fourier series** for **periodic signals**:

$$f(t) = \sum_{m=-\infty}^{\infty} F_m e^{jm\frac{2\pi}{P}t} \Leftrightarrow F_m = \frac{1}{P}\int_{-P/2}^{P/2} f(t) e^{-jm\frac{2\pi}{P}t} dt \qquad (2.1)$$

which gives the relation between a continuous time signal f(t) with period P and an infinite set of contributions at discrete frequencies $m \cdot 2\pi/P$. Changing the bounds for the integral does not change F_m as long as the time interval has length P. We refer to the complex form of the Fourier coefficients, F_m, even though we usually assume the time signals f(t) to be real. Note that the usual notation in these notes is that values and functions in the frequency domain are written with **capitalized** letters, e.g. F_m or $F(\omega)$, in contrast to values and functions in the time domain, e.g. f(t) or f_n. We also assume some knowledge of **Fourier transforms** for **energy type signals** (a short summary is given in Section 2.2.2).

2.2.1 Discrete time signals

We may see the Fourier transform of discrete time signals as obtained by interchanging time and frequency in the Fourier series relations, (2.1). We now define a **spectrum** $F(\omega)$ (*spektrum*), usually complex, by the **Fourier transform** of **discrete time signals**:

$$F(\omega) = \sum_{n=-\infty}^{\infty} T f_n e^{-jn\omega T} \Leftrightarrow f_n = \frac{1}{2\pi}\int_{-\pi/T}^{\pi/T} F(\omega) e^{jn\omega T} d\omega \qquad (2.2)$$

where we assume that the infinite sum exists. It is very important to note that the **spectrum of a discrete time signal is always periodic** with the period $2\pi/T$ and that the spectrum is defined for a continuous range of frequencies. The time domain signal f_n is expressed in the second part of the relation as a function of the basic period ($-\pi/T \le \omega \le \pi/T$) of the

spectrum $F(\omega)$, and this is often the part that is interpreted as physically significant, but any period will do in the calculation. We shall discuss the interpretation in more detail when we analyze the sampling process below in Section 2.2.2. Note that f_n being real implies that $F(-\omega) = F^*(\omega)$ where * means complex conjugation. Thus $F(\omega)$ only has to be known in half of the basic period for real signals. The physical dimension of $F(\omega)$ is a density, e.g. V/Hz for a simple voltage signal. Note that the scaling with 2π before the integral in (2.2) takes care of the integration variable, ω, being measured in rad/s instead of Hz. Thus the size of $F(\omega)$ is independent of the choice of frequency measure.

Example 2.1 Fourier transform of discrete time signals

Assume that a sequence f_n has $f_n = 0$ for $n < 0$ we can write the values for $n \geq 0$ as the vector

$$\mathbf{f} = (1,2,1,0,0,...)$$

and find the spectrum as

$$F(\omega) = T(1 + 2e^{-j\omega T} + e^{-j2\omega T}) = 2T(1 + \cos(\omega T))e^{-j\omega T}$$

where the exponential indicates a shift of one sample period, and the remaining factor, $2T(1+\cos(\omega T))$, is known as the **amplitude spectrum** (*amplitude-spektrum*). The argument of the complex exponential, $-\omega T$, is known as the **phase spectrum** (*fasespektrum*). Note that if the sequence started at $n = -1$, it would become symmetrical around 0 (also known as an **even function**) and the spectrum would become **real**, i.e. the amplitude spectrum calculated above.

If you have any difficulties in understanding the equation above, please remember the definition of complex exponentials and the so-called Euler's relation. It is very often used in this course.

The time function can be recovered by using the right part of (2.2), however, in similar problems we **strongly** encourage the reader to work back to the sum of exponentials and use the left side.

Discrete time harmonic sequences, either real or complex, $e^{jn\omega T}$, play a role similar to that of sinusoidal (trigonometric) functions and complex exponentials, $e^{j\omega t}$, in continuous time. The frequency $\omega = \pi/T$ (1/(2T) Hz) is special in the way that, just as for the frequency 0, it is a real signal without a phase (or phase 0). We shall discuss general time discrete periodic signals below.

Example 2.2 Complex exponentials

Complex exponentials for a specific frequency (4/NT Hz)

$$e^{jk\omega T} = e^{jk\cdot 4\frac{2\pi}{NT}T} = e^{j2\pi\frac{4k}{N}}, \quad k = 0...N$$

may be generated and displayed in MATLAB with a program like the following:

```
N=20;
k=0:1:N;
T=1;
omega=2*pi*4/N/T;
f=exp(j*k*omega*T);
stem(k,real(f),'--o');
hold on;
stem(k,imag(f),'--*');
```

The result is shown in Figure 2.1. The sinusoidal functions for the imaginary and real parts are very apparent, but it is with purpose that we have not connected the

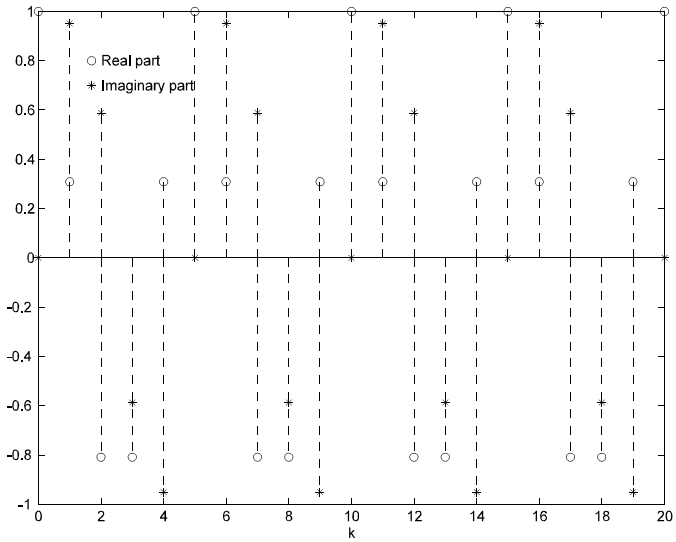

Figure 2.1
Real and imaginary parts of complex exponential

points, but instead shown them as discrete time signals with the `stem` function. Note that a discrete time complex exponential does not necessarily become periodic as it is always the case for time continuous complex exponentials. Periodicity only occurs if ωT is 2π times a rational number. The period in Figure 2.1 is 5 = N/4.

Note also that the exponentials for $k\omega_0$ and $k(\omega_0+m2\pi/T)$ are identical for all m.

For a **discrete and periodic signal** with period N, the transform (2.2) cannot be applied directly since the infinite sum does not exist in the usual sense. Instead, these signals are described by a **Fourier series** as we know from the time continuous case. We shall relate the two versions later in this section, and we shall not go into further details, but simply define the Fourier series and the relation of the coefficients as

$$F_m = \frac{1}{N}\sum_{k=0}^{N-1} f_k e^{-j2\pi\frac{km}{N}} \quad \Leftrightarrow \quad f_k = \sum_{m=0}^{N-1} F_m e^{j2\pi\frac{km}{N}} \tag{2.3}$$

Note the similarity to the Fourier series shown in (2.1). Actually the discrete transform may be seen as an approximation to the integral for the continuous time signal with period NT. The Fourier coefficient F_m is related to the frequency $m\cdot 2\pi/NT$. Since the spectrum only exists for certain frequencies it is known as a **line spectrum** (*liniespektrum*) in contrast to the frequency continuous spectrum from (2.2). As for (2.2), a real f_k implies $F_{-m} = F^*_m$, but since the frequencies $-m\cdot 2\pi/NT$ and $-m\cdot 2\pi/NT + 2\pi/T = (N - m)\cdot 2\pi/NT$ are identical (cf. Example 2.2), the negative frequencies are found at the high indices as well ($-m \sim N - m$). Note that the physical dimensions of f and F are the same in contrast to (2.2).

The transform from (2.3)

$$F_m = \frac{1}{N}\sum_{k=0}^{N-1} f_k e^{-j2\pi\frac{km}{N}} \quad \Leftrightarrow \quad f_k = \sum_{m=0}^{N-1} F_m e^{j2\pi\frac{km}{N}}$$

or in matrix formulation for $\mathbf{f} = \{f_k\}$ and $\mathbf{F} = \{F_m\}$ as column vectors

$$\mathbf{F} = \mathbf{A}\,\mathbf{f} \quad \Leftrightarrow \quad \mathbf{f} = \mathbf{A}^{-1}\mathbf{F}, \text{ where } \mathbf{A} = \frac{1}{N}\left\{e^{-j2\pi\frac{km}{N}}\right\}$$

is known as **Discrete Fourier Transform (DFT)**, and it is particularly important, since this is really the only version of the transform that can be calculated numerically. There are

many efficient computational procedures for calculating this transform, collectively referred to as **Fast Fourier Transform** (**FFT**), but it is not our purpose here to discuss the details of the computation. Note that some authors define DFT as the relation (2.2), whereas we use this term about the transform (2.3). As we saw above, the first component in the transform vector **F** represents the frequency 0. If N is even, the term m = N/2 is the frequency π/T. If N is odd, the contribution at this frequency is not calculated. The last $\lfloor (N-1)/2 \rfloor$ terms represent negative frequencies, and for real signals they are the complex conjugates of the first half. The negative frequencies appear in reverse order, since $-m \sim N - m$. Thus one has to **be careful with the placement of the values for the frequencies** when using predefined functions for the calculation as e.g. in MATLAB. We shall return to this matter in the example below.

The forward transform matrix **A** and the inverse transform matrix $\mathbf{A}^{-1}$ differ only by the factor 1/N and the order of the coefficients: Row k (k = 0,..,N − 1) of $\mathbf{A}^{-1}$ is row (N − k) mod N from **A**, which means that the first row is preserved and the order of the remaining rows are reversed compared to the forward transform. Another way to express the relation between **A** and $\mathbf{A}^{-1}$ is that $\mathbf{A}^{-1} = N\mathbf{A}^{*'}$ where ' denotes matrix transposing[1]. In these equations we have chosen to place a factor 1/N in the forward transform to provide the Fourier series for a discrete and periodic signal with period N. These scaling factors may be placed arbitrarily, but their product has to be 1/N to give the original back after a forward and an inverse transform. The exact placement affects certain relations between the time and the frequency domain as we shall see in Section 2.4. In programs for computing the DFT the factor 1/N is often not included, and there may be only one version of the transform. In **MATLAB** the function **`fft`** takes an input vector of any length and returns the DFT in a vector of the same length. The `fft`-function does **not** include 1/N. The inverse transform (with 1/N) is named **`ifft`**. If the length is a power of two, 2^n, the `fft`-function uses the fast version, FFT. For other lengths, the DFT is calculated directly.

---

[1] We have used ' as the transposing operator just as in MATLAB, but there is small difference: The operator ' (function `ctranspose`) in MATLAB also gives conjugation of complex numbers in the matrix, the so-called Hermitian transpose, but here we have shown the conjugation explicitly. Thus $^{*'}$ in our notation corresponds to ' in MATLAB where transposing without conjugation is .' (function `transpose`). In later sections ' may be used for other purposes also, but the correct interpretation should not be difficult.

Example 2.3 DFT transform of sampled square wave

The immediate representation of a sampled square wave (of period 8T) would be to assume it to start at time 0 like before

$$\mathbf{f} = (1,1,1,1,1,0,0,0)$$

It does not look very much like a square wave, but remember that if the sampling is done exactly at the rising and falling edges of a square that is high at 4T and low at 4T, the values at 0 and 4T are at the edges and somewhat undetermined. We have assumed both of them to be 1.

Like in the Example 2.1 the sequence has to be symmetric around 0 to give only real values in the transform. If you remember that the implicit periodicity of all sequences that are transformed with DFT, the proper way to represent **f** is

$$\mathbf{f} = (1,1,1,0,0,0,1,1)$$

where f(5) - f(8) represent negative indices $-4, \ldots, -1$, and the function is now even:

```
f=[1 1 1 0 0 0 1 1];
N=length(f);
F=fft(f)/N     % Equation (2.3)
F =  0.6250    0.3018   -0.1250   -0.0518
     0.1250   -0.0518   -0.1250    0.3018
```

Note that the last part of the vector **F** may be interpreted as negative index values. If you want the first term (n = 0) centered, there is a MATLAB function `fftshift` which despite its name is not a transform, but just a way of moving the elements to a more appropriate place as shown in the figure. The even function **f** is seen to give a real spectrum. However, in many cases MATLAB also produces a very small imaginary part due to the limited numerical precision, so we may in general have to take the real value before displaying the result, Figure 2.2.

```
stem(-4:3,fftshift(F))
```

The spectrum may be transformed back to the time domain by

```
fback=ifft(F)*N      % Equation (2.3)
fback =    1.0000    1.0000    1.0000         0
           0.0000         0    1.0000    1.0000
```

If the frequencies are placed wrongly in F before transforming, the result becomes wrong. E.g. `ifft(fftshift(F))*N` gives the `fback` sequence multiplied by alternating ±1. Why?

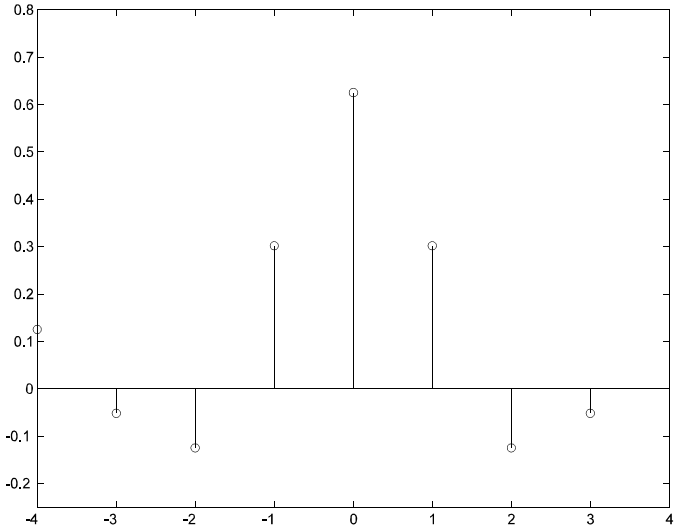

Figure 2.2
Amplitude spectrum of sampled square
(shifted such 0 is in the middle).

If the DFT is to serve as an **approximation to (2.2) for a finite signal**, we may first note that summation is restricted to N terms. Thus we get the correct result if the remaining signal values are 0 or at least small enough to be neglected, i.e. the signal is (approximately) finite in time. However the spectrum is computed only for N equally spaced value within the period $2\pi/T$:

$$F\left(m\frac{2\pi}{NT}\right) \approx \sum_{k=0}^{N-1} T f_k e^{-j2\pi\frac{km}{N}}$$

whereas (2.2) gives a continuous function. This is similar to (2.3), but to give the correct dimensions using DFT in this case, T and N should be placed correctly, so the relation between the frequency domain and the time domain becomes:

$$\left(F\left(m\frac{2\pi}{NT}\right) \approx\right) F_m = \sum_{k=0}^{N-1} T f_k e^{-j2\pi\frac{km}{N}} \quad \Leftrightarrow \quad f_k = \frac{1}{NT}\sum_{m=0}^{N-1} F_m e^{j2\pi\frac{km}{N}} \qquad (2.4)$$

From the discussion of MATLAB function `fft`, it may be seen that it uses the placement of N as given in (2.4), but does not include T (or 1/T in the inverse transform `ifft`). Example 2.4 below illustrates (2.4). The scaling 1/(NT) in the righthand expression of (2.4) is the frequency resolution of the approximation of the integral of (2.2), i.e. $d\omega = 2\pi/(NT)$.

Example 2.4 DFT as an approximation to the Fourier transform

In Example 2.1 we calculated the spectrum of

$$\mathbf{f} = (1,2,1,0,0,...)$$

The following MATLAB program compares this spectrum with the spectrum calculated by (2.4) with the proper scaling of `fft` and `ifft`. We have not shown the graphical output, but note the easy way of making comparisons with `stem` for one result and `plot` for other:

```
N=16;                    % Chosen to get enough information
T=2;                     % Chosen to demonstrate the effect of T
f=[1 2 1 zeros(1,N-3)]; % From Example 2.1
F=T*fft(f);              % Equation (2.4)
FIndex=-N/2:N/2-1;
stem(FIndex,real(fftshift(F)))
hold on
% Analytical expression from Example 2.1
omega=2*pi*FIndex/N/T;
plot(FIndex,real(2*T*(1+cos(omega*T)).*exp(-j*omega*T)))
hold off
pause
stem(FIndex,imag(fftshift(F)))
hold on
plot(FIndex,imag(2*T*(1+cos(omega*T)).*exp(-j*omega*T)))
hold off
% Inverse transform
f=ifft(F)/T
f = 1.0000    2.0000    1.0000         0    0.0000         0
    0.0000         0         0         0         0         0
         0         0    0.0000         0
```

2.2.2 Continuous time signals

In this course we deal almost exclusively with discrete time signals, but in a few cases we have to consider continuous time signals. To assist previous knowledge we presented the Fourier series for a periodic continuous time signal in (2.1), and here we state the definition of the Fourier transform for a **non-periodic continuous time** signal f(t).

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-j\omega t} dt \Leftrightarrow f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega \qquad (2.5)$$

Also here $|F(\omega)|$ is known as the amplitude spectrum and the argument of $F(\omega)$ is known as the phase spectrum. Note that $F(\omega)$ from (2.2) may be seen as an approximation for the continuous time transform if the signal is sampled closely enough, $f_n = f(nT)$, but $F(\omega)$ from (2.2) is periodic as all spectra from discrete time systems in contrast to $F(\omega)$ from (2.5).

The discrete time signals can be mixed with continuous time representations by means of delta functions, but we shall avoid this notation whenever possible.

Example 2.5 Fourier transforms of some non-periodic continuous time signals

Some known results are shown in the table below. Please remember that transforms occur in pairs. An example is the first two rows. The transform of a transform of a function brings us nearly back to the original function.

f(t)	F(ω)
$e^{j\omega_0 t}$	$\delta(\omega-\omega_0)$
$\delta(t-t_0)$	$e^{-j\omega t_0}$
$u_{-1}(t)$ (unit step at t = 0)	$\frac{1}{2}\delta(\omega) + \frac{1}{j\omega}$
1 for $-L/2 \le t \le L/2$ and 0 otherwise	$L\frac{\sin\frac{\omega L}{2}}{\frac{\omega L}{2}}$

In the last row, an important property is seen. The time function that is limited to a finite interval (i.e. zero outside) results in a frequency function that is not limited to any interval. Using the "pair-property" from above, the opposite statement may also be made: A frequency function limited to some interval ("band-limited") must always correspond to a time function that is infinitely long.

When a signal f(t) with spectrum $F(\omega)$ is **sampled** (*aftastet* or *samplet*) with a sampling frequency of 1/T, the spectrum of the sampled signal, f(nT), becomes periodic with period $2\pi/T$ as for all discrete time signals, and it may be calculated, [2.1, 2.2], from $F(\omega)$ as:

$$\tilde{F}(\omega) = \sum_{n=-\infty}^{\infty} F(\omega - n\cdot\frac{2\pi}{T}) \qquad (2.6)$$

From this spectrum, the **sampling theorem** (*samplingsætningen*) may be seen. If the signal is to be reconstructed from the sampled values, the spectrum in the basic period ($-\pi/T \le \omega \le \pi/T$) must be equal to $F(\omega)$ such that this frequency range could be filtered out by a lowpass filter (frequencies up to $1/(2T)$ passed), and $f(t)$ is then found at the output of that filter. It is clear that for this to be possible, the sampling rate has to be higher than twice the highest frequency in the signal spectrum. If this is not the case, components in the above sum overlap and the spectrum in the basic period is modified. The phenomenon is known as **aliasing** (*aliasering*) since the copies ("aliases") of $F(\omega)$ disturb the reconstruction.

2.2.3 Overview of Fourier transforms

Signal type	Time: Continuous	Time: Discrete
Periodic (period P ~ NT)	Fourier series $f(t) = \sum_{m=-\infty}^{\infty} F_m e^{jm\frac{2\pi}{P}t}$ $\Leftrightarrow F_m = \frac{1}{P}\int_{-P/2}^{P/2} f(t) e^{-jm\frac{2\pi}{P}t} dt$ MATLAB: `fft(f)/N, ifft(F)*N`	DFT $f_k = \sum_{m=0}^{N-1} F_m e^{j2\pi\frac{km}{N}}$ $\Leftrightarrow F_m = \frac{1}{N}\sum_{k=0}^{N-1} f_k e^{-j2\pi\frac{km}{N}}$ MATLAB: `fft(f)/N, ifft(F)*N`
Pulse (energy type signal)	Fourier transform $f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega$ $\Leftrightarrow F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$ MATLAB: `T*fft(f), ifft(F)/T`	Discrete Fourier transform $f_n = \frac{1}{2\pi}\int_{-\pi/T}^{\pi/T} F(\omega) e^{jn\omega T} d\omega$ $\Leftrightarrow F(\omega) = \sum_{n=-\infty}^{\infty} T f_n e^{-jn\omega T}$ MATLAB: `T*fft(f), ifft(F)/T`

Table 2.1
Fourier transforms and MATLAB expressions.

2.2.4 Convolution of signals

The **convolution** (*foldning*) of two discrete time sequences, f_m and g_m, is defined as

$$c_n = (f*g)_n = \sum_{m=-\infty}^{\infty} T f_m g_{n-m} \quad \Leftrightarrow \quad C(\omega) = F(\omega)G(\omega) \tag{2.7}$$

The factor T is included in order to give the same physical dimension as obtained by integration with respect to t in the continuous time case, and this factor is of course consistent with the use of T in (2.2) as it seen in the proof below. Note that f and g may be swapped, so $(f*g)_n = (g*f)_n$. In MATLAB, the function `conv` performs a convolution of two vectors of length N and M and provides the result as a vector of length N+M−1. The `conv` function does not include T so it has to be multiplied onto the result. The `conv` function may also be interpreted as a multiplication of two polynomials (coefficients stored in f and g) as it may be seen from the proof of (2.7):

$$\begin{aligned}
F(\omega)G(\omega) &= \sum_{m=-\infty}^{\infty} T f_m e^{-jm\omega T} \sum_{k=-\infty}^{\infty} T g_k e^{-jk\omega T} \\
&= \sum_{m=-\infty}^{\infty} T \sum_{n-m=-\infty}^{\infty} T f_m g_{n-m} e^{-j(m+n-m)\omega T} \\
&= \sum_{m=-\infty}^{\infty} T \sum_{n=-\infty}^{\infty} T f_m g_{n-m} e^{-jn\omega T} = \sum_{n=-\infty}^{\infty} T \left(\sum_{m=-\infty}^{\infty} T f_m g_{n-m} \right) e^{-jn\omega T} \\
&= \sum_{n=-\infty}^{\infty} T (f*g)_n e^{-jn\omega T} = C(\omega)
\end{aligned}$$

The convolution rule (2.7) may be reformulated in several different situations depending on whether the sequences are finite or periodic. In the equations below, f_k is always a discrete time periodic signal (period NT). For g a discrete time signal with finite energy (also known as a pulse, energy is introduced in Section 2.4) we define:

$$\begin{aligned}
&c_\ell = (f*g)_\ell = \sum_{m=-\infty}^{\infty} T f_m g_{\ell-m}, \ \ell = 0...N-1 \\
&\Leftrightarrow \quad C_n = F_n G(n\tfrac{2\pi}{NT}), \ n = 0...N-1
\end{aligned} \tag{2.8}$$

It is easy to prove that the convolution c_ℓ becomes periodic with period N such that only the index intervals indicated are relevant. For g also a discrete time periodic signal (with the same period NT), we define:

$$c_\ell = (f*g)_\ell = \frac{1}{N}\sum_{m=0}^{N-1} f_m\, g_{\ell-m}, \quad \ell = 0...N-1$$
$$\Leftrightarrow \quad C_n = F_n\, G_n, \quad n = 0...N-1 \tag{2.9}$$

which of course is periodic with period N.

Example 2.6 Convolution of sequences

```
T=1;
f=[1 1 2 1 3];
g=[1 1 -1];
c=T*conv(f,g)
c =     1     2     2     2     2     2    -3
c=T*conv(g,f)
c =     1     2     2     2     2     2    -3
```

As expected, the result does not depend on the order of the operands. T is of no importance in this example, but used to follow the definitions above. More illustrations of convolution are found in Example 2.7, especially the property in the frequency domain.

For the periodic convolutions of (2.8) and (2.9), it is a problem that older versions of MATLAB did not provide a direct function for the calculation. This may be solved by adding some extra periods of **f** in front of the principal period. The number of periods to add is determined by the length of **g** since all coefficients of **g** except the first should be covered by the added periods. The result is then found in the principal interval starting in the convolution after the added periods. Newer versions have the function `cconv` which may be used as shown below. A small example will show the operation of both methods where **f** now is periodic with the period from above:

```
N=length(f);
c=T*cconv(f,g,N)
c =  3    -1     2     2     2
% One extra period of f needed for f(m) to cover g(l-m)
c=T*conv(g,[f f])
c =  1     2     2     2     2
     3    -1     2     2     2     2    -3
% Relevant portion of periodic convolution starts at N+1:
c=c(N+1:2*N)
c =  3    -1     2     2     2
% Long pulse gl
gl=[1 1 -1 -2 -3 -4 -5];
```

```
c=T*cconv(f,gl,N)
c =  -23   -18   -20   -23   -20
% Without cconv: More periods of f needed to cover gl
nperiod=ceil((length(gl)-1)/N);
c=T*conv(gl,repmat(f,1,nperiod+1))
c =  1    2    2    0   -3   -8  -18  -20  -23  -20
   -23  -18  -20  -23  -20  -24  -20  -22  -23  -17  -15
% Relevant portion of periodic convolution starts after nperiod:
c=c(nperiod*N+1:nperiod*N+N)
c =  -23   -18   -20   -23   -20
```

A similar trick may be used to calculate the periodic convolution in (2.9) without cconv:

```
% Take a gper with period N as f
gper=[1 1 -1 -1 1];
c=cconv(gper,f,N)/N
c = 0.4000         0         0    0.8000    0.4000
% Without cconv: Two periods of f needed to cover gper
c=conv(gper,[f f])/N;
% Relevant portion of periodic convolution starts at N+1:
c=c(N+1:2*N)
c = 0.4000         0         0    0.8000    0.4000
```

2.2.5 Z-transform

In the literature on discrete time systems (e.g. [2.1, 2.3]), the exponentials in the Fourier transform are often replaced by z defined as

$$z = e^{j\omega T} \quad \Rightarrow \quad \sum_{n=-\infty}^{\infty} f_n e^{-jn\omega T} = \sum_{n=-\infty}^{\infty} f_n z^{-n} = F(z) \tag{2.10}$$

This transform is called the **Z-transform** (*z-transformation*). It may be used as a formal way of dealing with sequences, i.e. mostly a way of book keeping. But it can also be evaluated on the unit circle, $|z| = 1$, to give the discrete time Fourier transform, (2.2), for a fixed T. However, we prefer to maintain the explicit dependence on T and the relation to the Fourier Transform. For the Z-transform, a convolution rule as (2.7) also exists

$$c_n = (f*g)_n = \sum_{m=-\infty}^{\infty} T f_m g_{n-m} \quad \Leftrightarrow \quad C(z) = F(z)G(z)$$

It is proven in the same way.

2.3 Linear systems

We shall assume some knowledge about linear continuous time systems (i.e. circuit theory, e.g. [2.1]), and we shall usually follow a closely related notation for discrete time systems. We are only considering time-invariant linear systems and from linear continuous time systems, you will remember that such systems are characterized by a fixed function, h(t), the impulse response, which in the frequency domain is known as H(ω), the transfer function.

A **time-invariant discrete time linear system** may be characterized by its response to an **impulse** (*impuls*)

$$\delta_n = \begin{cases} \frac{1}{T} & \text{for } n = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

Note that we scale the input by T to maintain the dimension of the continuous time delta function. The linear system acts by convolving the input with the **impulse response** (*impulsrespons* or *impulssvar*), **h**, which for the impulse input gives the impulse response as output:

$$(h * \delta)_n = \sum_{m=-\infty}^{\infty} T h_m \delta_{n-m} = h_n$$

The dimension of the impulse response is time^{-1} to preserve the dimension through the convolution. The spectrum of an impulse is

$$\Delta(\omega) = \sum_{n=-\infty}^{\infty} T \delta_n e^{-jn\omega T} = T \cdot \frac{1}{T} e^{-j0\omega T} = 1, \tag{2.12}$$

i.e. constant, and using the convolution rule with an impulse input, we get the **transfer function** (or system function) (*overføringsfunktion*) as output

$$H(\omega) = \sum_{k=0}^{\infty} T h_k e^{-j\omega kT} \tag{2.13}$$

With the dimension time^{-1} of **h**, the transfer function becomes dimensionless. Note that we restricted the impulse response **h** to start at some point in time (k = 0). This is a reasonable requirement such that no output is seen before the input starts. The linear system is said to

be **causal**. Sometimes it is more convenient for the calculations to start somewhat before 0, e.g. to obtain an **h** that is symmetrical around 0 and easy to transform, and this may also be done, keeping the delay that makes the system causal out of the calculations.

As mentioned, we have here restricted ourselves to linear systems which are time-invariant so that **h** and H(ω) are fixed functions. Some transmission channels to be discussed in Chapter 7 and modulators (Chapter 6) are linear, but have time-varying impulse response.

For a general finite energy input signal **f**, the output **g** is found as the convolution with **h** (as for the impulse response, since **f** may be described as a series of scaled impulses):

$$g_n = \sum_{k=0}^{\infty} T h_k f_{n-k} \quad \Leftrightarrow \quad G(\omega) = H(\omega)\, F(\omega) \tag{2.14}$$

Here it becomes clear that the dimension of **h** is time^{-1} since **f** and **g** are to have the same dimension. In the frequency domain, H(ω) has to be dimensionless for this to happen. From (2.14) it is see that a time-invariant linear system does not create new frequencies at the output compared to the frequencies which are input. Equation (2.14) describes the linear system as a convolution as in (2.7) where the proof for the relation in the frequency domain is found. With a periodic input as **f**, (2.8) has to be applied instead, such that H(ω) is only relevant for $\omega = m \cdot 2\pi/(NT)$, where N is the period.

An example of a system with a finite impulse response is shown in Figure 2.3.

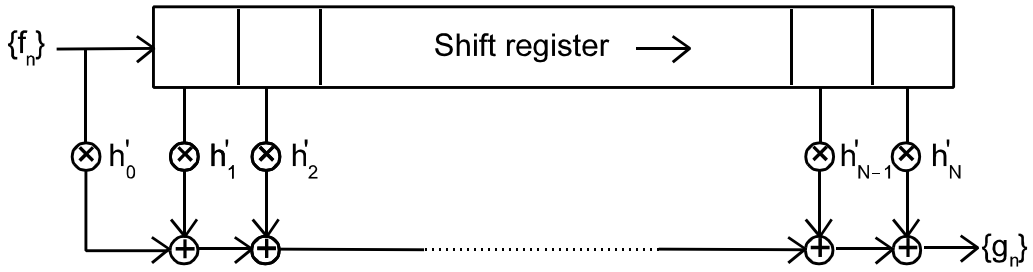

Figure 2.3
Discrete time linear system with finite impulse response (known as FIR). The realization is a tapped delay line with the delays of T implemented as a shift register shifting at a rate of 1/T, and the taps are connected to multiplicators for the coefficients, $h_k' = Th_k$ of the impulse response.

Example 2.7 Linear system with finite impulse response

Consider a system with impulse response $\mathbf{h} = (1, -1, 0, 0, \ldots)/T$ which by (2.14) gives the difference equation

$$g_k = f_k - f_{k-1}$$

From (2.13) we get

$$H(\omega) = \sum_{k=0}^{\infty} T h_k e^{-j\omega kT} = 1 - e^{-j\omega T}$$

Let the input be $\mathbf{f} = (1, 2, 1, 0, 0)$ (as in Example 2.1). We may get the output from the right side of (2.14):

$$\begin{aligned} G(\omega) &= H(\omega)F(\omega) = (1 - e^{-j\omega T})T(1 + 2e^{-j\omega T} + e^{-j2\omega T}) \\ &= T(1\cdot 1 + (2-1)e^{-j\omega T} + (1-2)e^{-j\omega 2T} + (-1)e^{-j\omega 3T}) \\ &= T(1 + 1\cdot e^{-j\omega T} + (-1)e^{-j\omega 2T} + (-1)e^{-j\omega 3T}) = \sum_{k=-\infty}^{\infty} T g_k e^{-j\omega kT} \end{aligned}$$

where the last identity is (2.2), and we get $\mathbf{g} = (1, 1, -1, -1)$ as output. However, it is much easier to use the following MATLAB program

```
T=2;
f=[1 2 1];
h=[1 -1]/T;
g=T*conv(f,h)
g =  1     1    -1    -1
```

We may also use the frequency approach in MATLAB. Here it is important to have the same frequency resolution for all sequences so we have to choose a length of the transform that is long enough to contain both **f**, **g**, and **h**. From above we know that length 4 is enough, but if we did not know it exactly, one could be on the safe side and choose a longer length. All the vectors should have the same length, so we append zeros to the shorter vectors. We assume use of a periodic input signal **f** corresponding to measurement practice, but with slightly different MATLAB commands, the output result **g** becomes the same.

```
N=4;
F=fft([f 0])/N
F =   1.0000      0 - 0.5000i         0          0 + 0.5000i
H=T*fft([h 0 0])       % (2.13)
H =   0          1.0000 + 1.0000i   2.0000      1.0000 - 1.0000i
G=F.*H
G =   0          0.5000 - 0.5000i    0          0.5000 + 0.5000i
g=ifft(G)*N
g =   1     1     -1     -1
```

In Example 3.12 we shall see how to calculate an impulse response having certain properties in the frequency domain. In [2.3], much more theory of time discrete systems is presented, e.g. how to construct filters with specified impulse responses or transfer functions and various approximations and optimizations. A simple method for such a filter is found in Example 3.12. In Section 2.8 we present ways of measuring an unknown impulse response.

A more general discrete time linear system may be defined by the difference equation for input **f** and output **g**:

$$\mathbf{g_n} = -\sum_{\mathbf{k=1}}^{\mathbf{N}} \mathbf{q_k g_{n-k}} + \sum_{\mathbf{k=0}}^{\mathbf{N}} \mathbf{p_k f_{n-k}} \tag{2.15}$$

Note that **p** and **q** are dimensionless here in contrast to **h** in (2.14). If this is not the case, appropriate T's have to be added in the equation. Here the previous output values g_{n-k} also influence the current output value through q_k. The impulse response then becomes infinite (IIR) if some of these are non-zero. The system is shown in Figure 2.4.

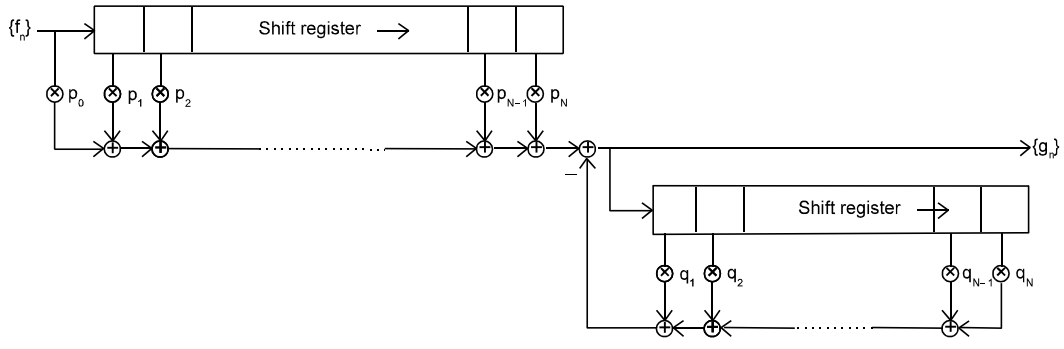

Figure 2.4
Discrete time linear system with infinite impulse response (known as IIR) due to the feedback of the output values through q_j coefficients.

The infinite impulse response is not so easy to formulate, but it is easy to find transfer function from the relation $Q(\omega)G(\omega) = P(\omega)F(\omega)$ (Q and P defined below) which is found after transforming a slightly rearranged version of (2.15). The transfer function then becomes:

$$H(\omega) = \frac{G(\omega)}{F(\omega)} = \frac{P(\omega)}{Q(\omega)} = \frac{p_0 + p_1 e^{-j\omega T} + \ldots + p_N e^{-j\omega NT}}{1 + q_1 e^{-j\omega T} + \ldots + q_N e^{-j\omega NT}} \qquad (2.16)$$

The **order** (*orden*) of the system is index of the largest index non-zero q_k.

Example 2.8 Linear system with infinite impulse response

Consider a first order system defined by the difference equation

$$g_n = -q g_{n-1} + f_n + p f_{n-1}$$

The impulse response **h** is found as the response to the impulse δ applied to the input, i.e. $f_n = \delta_n$, but it will in general depend on the contents of the shift register, i.e. g_{-1}. If we assume $g_{-1} = 0$, we find

$$h_0 = g_0 = -qg_{-1} + \delta_0 + p\delta_{-1} = 1/T$$
$$h_1 = g_1 = -qg_0 + \delta_1 + p\delta_0 = -q/T + p/T = (p - q)/T$$
$$h_2 = g_2 = -qg_1 + \delta_2 + p\delta_1 = -qh_1 = -q(p - q)/T$$
$$h_n = g_n = -qg_{n-1} + \delta_n + p\delta_{n-1} = -qh_{n-1} = (-q)^{n-1}(p - q)/T \text{ for } n \geq 1$$

Thus it is possible to determine the impulse response in this simple case, but it is also seen that it never vanishes for $q \neq 0$. Thus the system is an IIR system. For $|q| < 1$, the system is said to be stable since for any input sequence **f** with finite values, the values of the output **g** will be finite. If $|q| \geq 1$, the system becomes unstable, and e.g. a sequence of all ones input to a system with $q = -1$ gives an ever increasing output.

MATLAB does not have a simple way to calculate the response for IIR systems as `conv` does for the FIR systems. The following program calculates the impulse response (for T = 1) from above and it may be generalized to all systems of the (2.15) type. The signal names may also be modified to give output **g** (instead of **h**) for an arbitrary sequence **f** (instead of **d**).

```
T=1;
p=-1/2;                 % FIR part (numerator)
q=1/4;                  % IIR part (denominator)
d=[1 zeros(1,1000)]/T;% Delta function for impulse response
h=zeros(1,length(d)); % Initialize output (and IIR-register)
h(1)=d(1);              % First output equals input
                        % since shift register assumed to be zero
for k=2:length(h)
    h(k)=-q*h(k-1)+d(k)+p*d(k-1); % (2.12)
end

h(1:18)
1.0000   -0.7500    0.1875   -0.0469    0.0117   -0.0029
0.0007   -0.0002    0.0000   -0.0000    0.0000   -0.0000
0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000
```

A second order system is defined by the difference equation

$$g_n = -(q_1 g_{n-1} + q_2 g_{n-2}) + p_0 f_n + p_1 f_{n-1} + p_2 f_{n-2}$$

We shall not discuss details here, but note that there are general methods for solving difference equations (see e.g. [2.3]) and that the general solution is powers of the roots in the equation $x^2 + q_1 x + q_2$ (compare to the denominator in (2.16)) with constants adjusted to the starting condition, e.g. 0s in the shift register. Thus stability is found when the absolute values of these roots are < 1, just as for the first order case above.

Linear systems may be cascaded and the resulting impulse response and transfer function are easily calculated. Let $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$ be the two impulse responses:

$$g_n = \sum_{k=0}^{\infty} T h_k^{(2)} \left(\sum_{m=0}^{\infty} T h_m^{(1)} f_{n-k-m} \right) = \sum_{p=0}^{\infty} T \left(\sum_{k=0}^{\infty} T h_k^{(2)} h_{p-k}^{(1)} \right) f_{n-p}$$

$$= \sum_{p=0}^{\infty} T (h^{(2)} * h^{(1)})_p f_{n-p} \Leftrightarrow$$

$$G(\omega) = H^{(1)}(\omega) H^{(2)}(\omega) F(\omega)$$

so the impulse responses are just convolved together and the transfer functions are multiplied.

As it is known from the theory of continuous time linear systems, the relation (2.14) and the relation above between the transfer functions also hold there. We shall often use linear systems as filters in communication systems and usually we show them in block diagrams

as boxes with the continuous/discrete impulse response, h(t)/**h** or the transfer function, H(ω), as text. Figure 5.10 is an example of this.

2.4 Energy and power spectra

For a real signal **f**, the **energy** (*energien*) is defined as

$$E_f = \sum_{k=-\infty}^{\infty} T f_k^2 \tag{2.17}$$

provided that the infinite sum exists. Such a signal with finite energy is known as an **energy type** signal. The most typical example is a signal that is finite in time. As described earlier, the physical dimension of a signal will often be referred to as a voltage. When the squared signal is multiplied by T as above we get the energy, although to be correct, it should be divided by an impedance (alternatively one may think of the signal as a current, and the square should be multiplied by the impedance). The reason for this abuse of terminology is that the signals will be considered without explicit connection to circuits. One may think of the factor T as indicating that the signal is constant during the sampling period (or (2.17) as an approximation to an integral of f^2 with respect to t).

It is of great interest to know how the energy is distributed as a function of the frequency. This is known as **energy spectrum** (*energispektrum*), and it is calculated by taking the absolute square of the spectrum

$$S_f(\omega) = |F(\omega)|^2 \tag{2.18}$$

The spectrum is a **spectral density** giving energy/Hz. This is easily seen from the dimension of F(ω), V/Hz.

Parseval's relation gives the relation between the energy calculated in the time domain and in the frequency domain:

$$E_f = \sum_{k=-\infty}^{\infty} T f_k^2 = \frac{1}{2\pi}\int_{-\pi/T}^{\pi/T} |F(\omega)|^2 \, d\omega \tag{2.19}$$

Note that only frequencies in the basic period of F(ω) are of interest here. The proof is simple:

$$\int_{-\pi/T}^{\pi/T} |F(\omega)|^2 \, d\omega = \int_{-\pi/T}^{\pi/T} F(\omega)F^*(\omega) \, d\omega = \int_{-\pi/T}^{\pi/T} \sum_{k=-\infty}^{\infty} T f_k e^{-jk\omega T} \sum_{n=-\infty}^{\infty} T f_n e^{jn\omega T} \, d\omega$$

$$= \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} T^2 f_k f_n \int_{-\pi/T}^{\pi/T} e^{j(n-k)\omega T} \, d\omega = 2\pi \sum_{k=-\infty}^{\infty} T f_k^2$$

since the integration in the last line gives $2\pi/T$ for n = k and 0 otherwise.

If we use the DFT as in (2.4), i.e. like the MATLAB `fft`, to give (an approximation to) the spectrum F_m at the frequency $m \cdot 2\pi/NT$, the relation for the energy of a finite signal becomes

$$\sum_{k=0}^{N-1} T f_k^2 = \frac{1}{NT} \sum_{m=0}^{N-1} |F_m|^2 \tag{2.20}$$

The sequence $|F_m|^2$ here is also known as the **energy spectrum**, and it is an approximation to $S_f(\omega) = |F(\omega)|^2$ defined in (2.18) simply because the righthand side of (2.20) is just an approximation of the integral in (2.19) with $d\omega = 2\pi/(NT)$ as frequency steps. A more direct proof of (2.20) could also be given.

Not all signals are of energy type with a finite energy. Especially, this is the case for periodic signals. As it is known from physics, the quantity **power** (*effekt*) is found by calculating energy per time unit. Thus the power of a signal **f** is defined as

$$P_f = \lim_{L \to \infty} \frac{1}{(2L+1)T} \sum_{k=-L}^{L} T f_k^2 \tag{2.21}$$

A signal for which the power is finite is known as a **power type** signal. As for the energy, an impedance should be included to get a physically correct power. For a **periodic** signal (period NT) the power becomes

$$P_f = \frac{1}{N} \sum_{k=0}^{N-1} f_k^2 \tag{2.22}$$

Again it is of great interest to know how the power is distributed as a function of the frequency. This is known as the **power spectrum** (*effektspektrum*), and it is calculated by taking the absolute square of the spectrum

$$S_f\left(m\frac{2\pi}{NT}\right) = |F_m|^2 \tag{2.23}$$

where F_m is calculated by (2.3). Note that the physical dimension of the power spectrum is power giving the amount of power for each of the multiples of the basic frequency for the periodic signal, $2\pi/(NT)$ in contrast to the energy spectrum for a energy type signal which is a spectral density.

Here we may also express the relation between power calculated in the time domain and in the frequency domain by **Parseval's relation** for a **periodic** signal as

$$P_f = \frac{1}{N}\sum_{k=0}^{N-1} f_k^2 = \sum_{m=0}^{N-1} |F_m|^2 \qquad (2.24)$$

where both sides express the power of a periodic signal f as the signal transformed by the DFT in (2.3). The proof may be done just as for a direct proof of (2.20).

Example 2.9 Energy and power spectra

If we continue with the signal

$$\mathbf{f} = (1,1,1,1,1,0,0,0)$$

from Example 2.3 we may first regard **f** as a finite energy signal and use MATLAB to verify the Parseval relation for the energy

```
T=1;
N=8;
f=[1 1 1 1 1 0 0 0];
energy_time=T*sum(f.^2)
energy_time =
     5
F=T*fft(f)            % (2.4)
F =
  Columns 1 through 4
   5.0000   0.0000 - 2.4142i   1.0000   0.0000 - 0.4142i
  Columns 5 through 8
   1.0000   0.0000 + 0.4142i   1.0000   0.0000 + 2.4142i
energy_spectrum=abs(F).^2
energy_spectrum =
   25.0000  5.8284  1.0000  0.1716  1.0000  0.1716
    1.0000  5.8284
energy_freq=sum(energy_spectrum)/N/T
energy_freq =
     5
```

The energy is 5T. Note that the amplitude spectrum in contrast to Example 2.3 now has non-zero imaginary parts, but since the absolute values are unchanged, selecting another starting time for the signal does not change the energy spectrum

since it is just the squared of the amplitude spectrum found in Example 2.3. The last line shows that the Parseval relation Eq. (2.20) is fulfilled.

Let us now use **f** as the period in a periodic signal, i.e. N = 8:

```
power_time=sum(f.^2)/N
power_time =
    0.6250
F=fft(f)/N;           % (2.3)
power_spectrum=abs(F).^2
power_spectrum =
  0.3906  0.0911  0.0156  0.0027  0.0156  0.0027
  0.0156  0.0911
power_freq=sum(power_spectrum)
power_freq =
    0.6250
```

The power is 5/8. Note that to calculate the power spectrum, the correct transform, (2.3), has to be used as in Example 2.3. The power is just the energy divided with the period NT = 8T, but to calculate the power spectrum for the periodic signal from a period of the energy spectrum for the non-periodic signal one has to divide by $(NT)^2$ since as in (2.20) the energy spectrum is a density so it has to be scaled with 1/(NT) to give the proper energy. The last line shows that the Parseval relation Eq. (2.24) is fulfilled.

	Time	
Signal type	Continuous	Discrete
Periodic (period P ~ NT)	Power spectrum $S_f\left(m\frac{2\pi}{P}\right) = \lvert F_m\rvert^2$ for integer m MATLAB: `abs(fft(f)/N).^2`	Power spectrum $S_f\left(m\frac{2\pi}{NT}\right) = \lvert F_m\rvert^2$ for $m = 0 \ldots N-1$ MATLAB: `abs(fft(f)/N).^2`
Pulse (energy type signal)	Energy spectrum, spectral density $S_f(\omega) = \lvert F(\omega)\rvert^2$ for $-\infty < \omega < \infty$ MATLAB: `abs(T*fft(f)).^2`	Energy spectrum, spectral density $S_f(\omega) = \lvert F(\omega)\rvert^2$ for $-\frac{\pi}{T} \le \omega \le \frac{\pi}{T}$ MATLAB: `abs(T*fft(f)).^2`

Table 2.2
Energy and power spectra with MATLAB expressions.

The energy/power spectra are interesting in order to characterize a signal, e.g. to determine the use of the available frequency range in some transmission system. However, they are not always easily calculated from a specification of a signal. In such cases it is sometimes easier to consider first the time domain version of the energy/power spectra which are called **autocorrelation functions** (*autokorrelationsfunktioner*). These are obtained as scalar products of a signal with a time-shifted version (or as the convolution of a signal with a version reversed in time). There are again two versions, (2.25) and (2.26):

For f_n a discrete time signal with **finite energy** we have:

$$\begin{aligned} S_f(\omega) &= \sum_{k=-\infty}^{\infty} T R_f(kT)\, e^{-jk\omega T} \\ R_f(kT) &= \sum_{n=-\infty}^{\infty} T f_{n+k} f_n \quad \Leftrightarrow \quad S_f(\omega) = |F(\omega)|^2 \end{aligned} \tag{2.25}$$

Note that this **non-periodic autocorrelation** function is **even** in time, i.e. $R_f(-kT) = R_f(kT)$. This implies that the energy spectrum must be a sum of cosine functions. Another property is that $R_f(0)$ is the energy of the signal (cf. (2.17)) and $R_f(0) \geq |R_f(kT)|$. The relation (2.25) is easily proven since R_f is a convolution of f_n with the time reversed version f_{-n} where the latter has the spectrum $F^*(\omega)$. Using the convolution rule (2.7) the right hand relation is obtained since $F(\omega)F^*(\omega) = |F(\omega)|^2$.

For a **periodic** signal with period NT we have

$$\begin{aligned} S_f\left(m\frac{2\pi}{NT}\right) &= \frac{1}{N}\sum_{k=0}^{N-1} R_f(kT)\, e^{-j2\pi\frac{km}{N}} \\ R_f(kT) &= \frac{1}{N}\sum_{n=0}^{N-1} f_{n+k} f_n \quad \Leftrightarrow \quad S_f\left(m\frac{2\pi}{NT}\right) = |F_m|^2 \end{aligned} \tag{2.26}$$

Note that the autocorrelation functions have different dimensions for periodic and non-periodic signals. Note also that $R_f(kT)$ is itself **periodic** for the periodic signal, and thus normally we only provide values of $R_f(kT)$ for $0 \leq k \leq N-1$. About half of these are redundant since the function is **even**, $R_f(kT) = R_f(-kT)$. $R_f(0)$ is the power of the signal (cf. (2.22)) and $R_f(0) \geq |R_f(kT)|$.

Similarly we may define the **crosscorrelation function** (*krydskorrelationsfunktion*) of two signals with **finite energy** by

$$\mathbf{R_{fg}(kT)} = \sum_{n=-\infty}^{\infty} \mathbf{T f_{n+k} g_n} \tag{2.27}$$

and its transform $S_{fg}(\omega)$ as the **cross energy spectrum** (*krydsenergispektrum*). Note that the relation between positive and negative arguments is a bit more complicated, $R_{fg}(kT) = R_{gf}(-kT)$ so the result is not even. As for the autocorrelation, the values of R_{fg} are upperbounded: $\left|R_{fg}(kT)\right| \le \sqrt{R_f(0)R_g(0)}$.

For **f** and **g periodic** signals (period NT) we define the **crosscorrelation function** by

$$\mathbf{R_{fg}(kT)} = \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{f_{n+k} g_n} \tag{2.28}$$

and its transform $S_{fg}(m2\pi/NT)$ as the **cross power spectrum** (*krydseffektspektrum*).

Example 2.10 Calculation of correlation functions in MATLAB

Comparing (2.25) and (2.7), one sees that the autocorrelation function may be calculated by convolving a sequence with the sequence reversed (MATLAB `fliplr`), but MATLAB also provides a function `xcorr` that directly gives correlations. It is intended for crosscorrelations as the next part of the program shows.

```
T=1;
f=[1 4 2 0 3];
Rf=T*xcorr(f,f)
Rf =      3    12     8    12    30    12     8    12     3
Rf1=T*conv(f,fliplr(f))
Rf1 =     3    12     8    12    30    12     8    12     3
```

Note that $R_f(0T)$ appears in the middle of the result, and that the vector is symmetric around this element. If the length of **f** is denoted N, the $\mathbf{R}_f$ has 2N − 1 elements. Thus an immediate transform of $\mathbf{R}_f$ has a frequency resolution different from that of a transform of **f** (unless **f** is extended with some zeros). Another problem with an immediate transform is that the result has non-real values since

the elements of an even function should be placed correctly to give a real result, cf. Example 2.3. For crosscorrelation we continue with:

```
g=[1 1 -1];
Rfg=T*xcorr(f,g)
Rfg =
    0.0000    0.0000   -1.0000   -3.0000    3.0000
    6.0000   -1.0000    3.0000    3.0000
Rfg1=T*conv(f,fliplr(g))
Rfg1 = -1   -3    3     6    -1    3    3
```

Note that $R_{fg}(0T)$ also appears in the middle of the result when using `xcorr`, but this vector is not symmetric around k = 0. If the maximum length of **f** and **g** is denoted N, the $\mathbf{R}_{fg}$ has 2N − 1 elements since MATLAB pads the shortest vector with zeros. Note that MATLAB up to versions 5.x uses another definition of `xcorr` which reverses the result compared to Equation (2.27). It may be said that in the earlier versions MATLAB calculates R_{gf} with `xcorr(f,g)`. Use of `fliplr` and `conv` makes it difficult to locate $R_{fg}(0T)$ since no padding is done so it could not be advised.

Calculation of the periodic autocorrelation by (2.26) is a bit more complicated since the function `xcorr` just assumes zero values for all non-defined values of f_{n+k} whereas the periodic autocorrelation has to use the periodic repetition of the signal. We use the same trick as for the periodic convolution in Example 2.6 to solve this. The result from `xcorr` also has to be divided by the period as seen in (2.26). Below we assume that **f** now is the period (N = 5) in the signal. As it may be seen, use of `fliplr` and the circular `cconv` is not recommendable since several extra manipulations are needed. This is due to that `fliplr` does not treat a periodic function correctly.

```
N=length(f);
ff=[f f];
Rtemporary=xcorr(f,ff)/N
Rtemporary =
0.6000  2.4000  1.6000  2.4000  6.0000  3.0000  4.0000
4.0000  3.0000  6.0000  2.4000  1.6000  2.4000  0.6000
0       0       0       0.0000  0.0000
Rf=Rtemporary(N:2*N-1)
Rf =  6     3     4     4     3
Rf1=fliplr(cconv(f,fliplr(f),N)/N)
Rf1 = 6     3     4     4     3
Rf2=cconv(f,[1 3 0 2 4],N)/N
Rf2 = 6     3     4     4     3
```

Note how the periodic autocorrelation is extracted from the non-periodic result found using the double period. Note also the symmetry in the values: $R_f(T) = R_f(4T) = R_f((4-5)T) = R_f(-T)$ and the same for 2T and 3T.

The periodic crosscorrelation function may also be calculated - with the same problems as above. First **g** needs to have the same period (length) as **f**:

```
gper=[1 1 -1 0 0];
gg=[gper gper];
Rtemporary=xcorr(f,gg)/N;
Rfg=Rtemporary(N:2*N-1)
Rfg  = 0.6000  1.2000 -0.2000  0.4000  0
Rfg1=fliplr(cconv(fliplr(f),gper,N)/N)
Rfg1 = 0.6000  1.2000 -0.2000  0.4000  0
Rfg2=cconv(f,[1 0 0 -1 1],N)/N
Rfg2 = 0.6000  1.2000 -0.2000  0.4000  0
```

The power/energy spectra may be found in two ways: Simple squaring of the amplitude spectra or calculation of the autocorrelation function followed by a transform. The two ways are demonstrated in Example 2.13 for a power spectrum.

Often we have a signal **f** that is input to a **linear system** and we want the spectrum of the output **g**. Taking the numerical square of both sides of the frequency part of (2.14) we get the important relation between the energy/power spectra:

$$S_g(\omega) = |H(\omega)|^2 S_f(\omega) \tag{2.29}$$

The power spectrum for a periodic **f** (and thus **g**) only exists at certain frequencies, $\omega = m \cdot 2\pi/(NT)$, where N is the period. Finally we may find the cross energy/power spectra between input and output as

$$S_{fg}(\omega) = H^*(\omega) S_f(\omega) \tag{2.30}$$

The cross power spectrum for a periodic **f** (and thus **g**) only exists at certain frequencies, $\omega = m \cdot 2\pi/(NT)$, where N is the period. The relation is a little involved to prove. For a non-periodic **f** we have:

$$R_{fg}(kT) = \sum_{n=-\infty}^{\infty} Tf_{n+k}g_n = \sum_{n=-\infty}^{\infty} Tf_{n+k}\sum_{s=0}^{\infty} Th_s f_{n-s} = \sum_{s=0}^{\infty} Th_s \sum_{n-s=-\infty}^{\infty} Tf_{n+k}f_{n-s}$$

$$= \sum_{s=0}^{\infty} Th_s R_f((k+s)T)$$

$$S_{fg}(\omega) = \sum_{r=-\infty}^{\infty} TR_{fg}(rT)e^{-jr\omega T} = \sum_{r=-\infty}^{\infty} T\sum_{s=0}^{\infty} Th_s R_f((r+s)T)e^{-jr\omega T}$$

$$= \sum_{s=0}^{\infty} Th_s e^{js\omega T} \sum_{r+s=-\infty}^{\infty} TR_f((r+s)T)e^{-j(r+s)\omega T} = H^*(\omega)S_f(\omega)$$

Example 2.11 Barker sequences

In radar systems a signal (a pulse) of known form is transmitted periodically and for the remaining time the signal is 0. In discrete time the pulse could be:

$$\mathbf{s} = (s_0, s_1, s_2, \ldots\ldots s_{M-1})$$

The receiver shall determine the arrival times for reflected pulses which determine the distance to the reflecting object. We describe the received signal as

$$f_n = \begin{cases} a\, s_{n-r} & r \le n < r+M \\ 0 & \text{otherwise} \end{cases}$$

and want to determine r. The receiver calculates the correlation

$$c_n = T\sum_{k=0}^{M-1} f_{n+k}\, s_k$$

At time r the correlation reaches a large positive value

$$c_r = a\,T\sum_{k=0}^{M-1} s_k^2$$

but hopefully it should be small for other n. This depends of course on the pulse, and examples of very good sequences are **Barker sequences**. Here $s_j = \pm 1$ and the autocorrelation function for odd M is

$$R_s(kT) = T\sum_{j=0}^{M-1} s_{j+k}s_j = \begin{cases} MT \text{ for } k = 0 \\ bT \text{ for even } |k| < M \\ 0 \text{ otherwise} \end{cases}$$

where b = 1 or b = − 1 for a given sequence. The even M sequences have a similar property. Very few such sequences are found and they are all listed in the following table from [2.4]

M	Sequence
2	(− 1, 1)
3	(− 1, 1, 1)
4	(− 1, 1, 1, 1), (1, − 1, 1, 1)
5	(1, − 1, 1, 1, 1)
7	(− 1, 1, − 1, − 1, 1, 1, 1)
11	(− 1, 1, − 1, − 1, 1, − 1,− 1, − 1, 1, 1, 1)
13	(1, − 1, 1, − 1, 1, 1, − 1, − 1, 1, 1, 1, 1, 1)

Table 2.3
Barker sequences.

The Barker sequence of length 7 has autocorrelation function

$$R_s(kT) = \begin{cases} 7T \text{ for } k = 0 \\ -T \text{ for } |k| \in \{2,4,6\} \\ 0 \text{ otherwise} \end{cases}$$

Transforming $R_s(kT)$ for the sequence gives the energy spectrum

$$\begin{aligned} S_s(\omega) &= (7 - 2\cos 2\omega T - 2\cos 4\omega T - 2\cos 6\omega T)T^2 \\ &= \left(8 - \frac{\sin 7\omega T}{\sin \omega T}\right)T^2 \end{aligned}$$

As seen from the figure below showing a period of $S_s(\omega)$, the spectrum is approximately white, i.e. constant (for frequencies having a meaning for a discrete signal). The frequency contents at 0 and π/T is rather small.

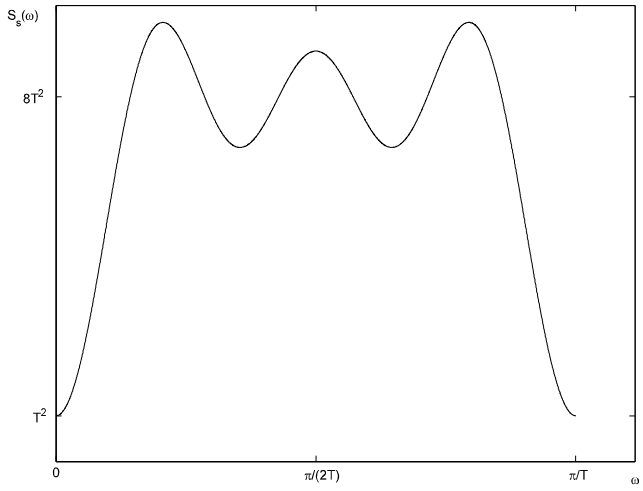

Figure 2.5
Energy spectrum of length 7 Barker sequence

2.5 Pseudorandom sequences

In this section we shall consider periodic sequences with the property that the power spectrum is close to a constant and the autocorrelation function is close to 0 for $k \neq 0$ (it follows from (2.26) that these properties are related by the DFT). These properties give the sequences some similarity to sequences that are generated randomly, although the numbers are in fact produced by an algorithm and repeat periodically. Very long pseudorandom sequences may be used in computers or in experiments to simulate random events or random signals (in Chapter 3 we shall study the properties of random signals and discuss such simulations). Here we shall concentrate on much shorter periods and discuss some applications in communications signals.

We shall be mostly interested in binary sequences, and initially we may represent the symbols as 0 and 1. The symbols may represent standard logic values, but it is an advantage here to think of them as integers modulo 2 (or residue classes to be more formal). Thus addition and multiplication has the usual meaning, except that 1+1 = 0 (exclusive-or). With this notation we may write a linear recursion as

$$x_k = g_1 x_{k-1} + g_2 x_{k-2} + \ldots + g_m x_{k-m} \mod 2, \quad g_m = 1 \tag{2.31}$$

If the m initial values are all 0, that will also be true of the following values. Since the m binary values can assume at most 2^m values, and the sequence thus has to repeat, the longest period is $N = 2^m - 1$. A sequence with this period is called **a maximal length (shift register) sequence** *(maksimallængdesekvens),* **m-sequence**. The term shift register sequence is sometimes used to indicate that such a sequence may be generated by a binary shift register with a linear feed-back (mod 2). Such a shift register implementation of the recursion (2.31) is shown in Figure 2.6. Note that since the coefficients g_n are binary, each multiplication corresponds to connection or no connection.

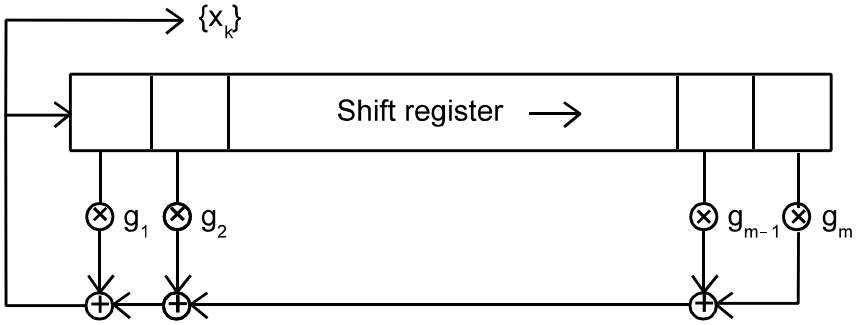

Figure 2.6
Shift register for generation of maximal length sequence

Rearranging terms we get (since + and − are the same mod 2):

$$0 = x_k + g_1 x_{k-1} + \ldots + g_{m-1} x_{k-(m-1)} + g_m x_{k-m}$$

Using the Z-transform (2.10) we get

$$0 = G(z) \cdot X(z)$$

where G(z) is the m'th degree ($g_m = 1$) binary polynomial

$$G(z) = 1 + g_1 z^{-1} + \ldots + g_{m-1} z^{-(m-1)} + g_m z^{-m} \qquad (2.32)$$

Two solutions for $0 = G(z) \cdot X(z)$ exist, one is the all zero sequence and the other is the m-sequence. A necessary condition for the coefficients g_j to define an m-sequence is that G(z) does not factor into binary polynomials of lower degree (in particular 1 should not be a root, and thus the number of nonzero coefficients must be odd). However, this is not sufficient, we need what is known as a primitive polynomial. Such polynomials exist for all m (and also for some non-binary alphabets), and they are usually obtained from tables. Some polynomials are shown in Table 2.4.

m	Period	Polynomial
3	7	$1 + z^{-2} + z^{-3}$
4	15	$1 + z^{-3} + z^{-4}$
5	31	$1 + z^{-3} + z^{-5}$
7	127	$1 + z^{-6} + z^{-7}$
8	255	$1 + z^{-4} + z^{-5} + z^{-6} + z^{-8}$
12	4095	$1 + z^{-6} + z^{-8} + z^{-11} + z^{-12}$
16	65535	$1 + z^{-4} + z^{-13} + z^{-15} + z^{-16}$

Table 2.4
Polynomials for maximal length sequences.
Some tables have these multiplied with z^m or in
reversed order which does not change the properties.

For our purpose it is sufficient to note that the period can be calculated by initiating the m values to [0, 0, …, 1] and applying the recursion (2.31) until this combination reappears. Maximal length sequences form the most important class of pseudorandom sequences, and they have many interesting properties. We shall note a few simple facts:

1: m consecutive values of x_k assume all N nonzero combinations exactly once.
2: A m-sequence consists of (N+1)/2 1s and (N−1)/2 0s.
3: The sequence is unique up to cyclic shifts. Adding (mod 2) two shifts of the sequence gives another shift.
4: In signal processing we often use a **bipolar sequence** version, i.e. the logical values 0 and 1 are replaced by the real numbers −1 and 1. If this sequence is used periodically we can calculate the autocorrelation function $R_x(kT)$ by (2.26)

$$R_x(kT) = \frac{1}{N}\sum_{n=0}^{N-1} x_{n+k}x_n = \begin{cases} 1 \text{ for } k = 0 \\ -\frac{1}{N} \text{ for } 1 \le k \le N-1 \end{cases}$$

Proof: Property 1 follows from the fact that the m values form a state of the recursion, and it is assumed that there are N distinct states (otherwise the period would be shorter). The recursion polynomial G(z) has an odd number of nonzero coefficients, since $1+z^{-1}$ would

be a factor otherwise. Thus x_j is expressed as a sum of an even number of terms always including x_{k-m}. These terms contain an even number of 1s $(N+1)/2-1 = (N-1)/2$ times (since the all zero state does not appear) and an odd number of 1s $(N+1)/2$ times. Since the first situation gives 0 as output and the second gives 1, property 2 follows. If the recursion (2.31) is given, the sequence is unique up to cyclic shifts. Adding two shifts of the sequence mod 2 gives another shift since, by linearity, the recursion (2.31) is still satisfied. In the bipolar version, a term by term product of two shifts gives +1 if the two terms agree and −1 if they disagree which is exactly the negative bipolar version of the two shifts added. Since this is just another shift by property 3, property 2 assures that the sum of the term by term products is −1 (for $k \neq 0$, N for $k = 0$) and property 4 is proved.

In some applications, the crosscorrelation properties of the pseudorandom sequences are as important as the autocorrelation properties discussed above. It turns out that the maximal length sequences are not very good in this aspect. Pseudorandom sequences with better crosscorrelation properties are studied in [2.5] and the references therein.

Example 2.12 Generation of maximal length sequence

Let the sequence satisfy the recursion

$$\mathbf{x}_n = \mathbf{x}_{n-3} + \mathbf{x}_{n-4}$$

from Table 2.4. Note how the polynomials from the table are connected to the recursion (2.31) and remember that every calculation here is mod 2.

We represent the register as rows of a matrix, *r*, and get the output as the first column. Initializing the register with $(x_{-1},x_{-2},x_{-3},x_{-4}) = (1,1,1,1)$ gives a period of the sequence $\mathbf{x} = (0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1)$:

```
r=[1 1 1 1];
for j=1:15
   % Add new row in r with shift register state
   r=[r;xor(r(j,3),r(j,4)) r(j,1:3)];
end
r
r =
   1     1     1     1
   0     1     1     1
```

```
   0     0     1     1
   0     0     0     1
   1     0     0     0
   0     1     0     0
   0     0     1     0
   1     0     0     1
   1     1     0     0
   0     1     1     0
   1     0     1     1
   0     1     0     1
   1     0     1     0
   1     1     0     1
   1     1     1     0
   1     1     1     1
% Output sequence
x=r(2:16,1)'
x =
0  0  0  1  0  0  1  1  0  1  0  1  1  1  1
```

Check that the sequence has the claimed properties 1) and 2).

Example 2.13 Spectrum of maximal length sequence

If 0 and 1 from the **x** sequence are replaced by the real values − 1 and 1, we obtain the periodic sequence **xreal** from which we can calculate the spectrum by applying the DFT, (2.3):

```
xreal=2*x-1
xreal =
  -1    -1    -1     1    -1    -1     1     1
  -1     1    -1     1     1     1     1
X=fft(xreal)/length(xreal)
X =
 0.0667            -0.0667 + 0.2582i   -0.0667 + 0.2582i
-0.2667 - 0.0000i  -0.0667 + 0.2582i    0.2667 + 0.0000i
-0.2667 - 0.0000i  -0.0667 - 0.2582i   -0.0667 + 0.2582i
-0.2667 - 0.0000i   0.2667 + 0.0000i   -0.0667 - 0.2582i
-0.2667 + 0.0000i  -0.0667 - 0.2582i   -0.0667 - 0.2582i
Sx=abs(X).^2
Sx =
0.0044  0.0711  0.0711  0.0711  0.0711  0.0711  0.0711
0.0711  0.0711  0.0711  0.0711  0.0711  0.0711  0.0711
0.0711
```

By trigonometric identities, it is "easy" to see that the real part of X_1 is − 1/15 and the imaginary part can be calculated to 3.873/15. It is more difficult to realize that it is exactly $\sqrt{15}/15$. However, this will be seen shortly. The last part of the program shows a direct calculation of the power spectrum as $|X_m|^2$ (not the m defining the m-sequence). This gives 1/225 for m = 0 and 16/225 for m ≠ 0. The

approximately constant power spectrum (a so-called white spectrum) is one of the special properties of a maximal length sequence. Note, that since the signal is periodic, the signal has a discrete spectrum (frequencies $2\pi m/15T$) so the meaning of 'constant' is valid for these frequencies, not frequencies in between. We shall also find the power spectrum of the m-sequence from the autocorrelation function. If the **xreal** is repeated periodically, a period of the autocorrelation function is given by property 4 above:

$$R_x(kT) = \frac{1}{15}\sum_{n=0}^{14} x_{n+k}x_n = \begin{cases} 1 \text{ for } k = 0 \\ -\frac{1}{15} \text{ for } 1 \le k \le 14 \end{cases}$$

The transform of R_x is easy calculated by splitting R_x into a constant term $-1/15$ for all k and a δ-function term 16/15 for k=0 (k=15p, p integer). Thus the transform is similarly a sum of two such terms

$$S_x\left(m\frac{2\pi}{15T}\right) = \frac{1}{15}\sum_{k=0}^{14} -\frac{1}{15}e^{-j2\pi km/N} + \frac{1}{15}\frac{16}{15}e^{-j0} = \begin{cases} \frac{1}{225} \text{ for } m = 0 \\ \frac{16}{225} \text{ for } 1 \le m \le 14 \end{cases}$$

This calculation confirms the power spectrum calculated directly above. Further, it confirms the $|X_m| = 4/15$ for $m \neq 0$ such that the imaginary part of X_1 really is $1/\sqrt{15}$.

```
Rx=xcorr(xreal,[xreal xreal])/length(xreal);
Rxper=Rx(length(xreal):2*length(xreal)-1)
Rxper =
 1.0000 -0.0667 -0.0667 -0.0667 -0.0667 -0.0667 -0.0667 -0.0667
-0.0667 -0.0667 -0.0667 -0.0667 -0.0667 -0.0667 -0.0667
Sx=fft(Rxper)/length(xreal)
Sx =
0.0044             0.0711 - 0.0000i   0.0711 + 0.0000i
0.0711 - 0.0000i   0.0711 - 0.0000i   0.0711 + 0.0000i
0.0711 - 0.0000i   0.0711 + 0.0000i   0.0711 - 0.0000i
0.0711 + 0.0000i   0.0711 - 0.0000i   0.0711 + 0.0000i
0.0711 + 0.0000i   0.0711 + 0.0000i   0.0711 + 0.0000i
```

The results are as calculated analytically above.

Example 2.14 High speed generation of maximal length sequence

As in Example 2.12, the sequence satisfies the recursion

$$x_n = x_{n-3} + x_{n-4}$$

Since there is a limit to how fast a shift register as that in Figure 2.6 may be clocked, there is a need to have a circuit that generates several, q, bits at each clock tick. Often q = m, but for large m, another number may be chosen, e.g. q = 8 or 16. Below we calculate a transition matrix which may be used both in programs and in hardware realizations. First we introduce an alternative way of generating the sequence using a state transition matrix **S** which may be built from the connection vector **f** (f for feedback) and a unit matrix giving the shifts. We demonstrate below that the same sequence is generated

```
m=4;
S=[eye(m-1);zeros(1,m-1)];
f=[0 0 1 1]';
S=[f S]
S =
     0     1     0     0
     0     0     1     0
     1     0     0     1
     1     0     0     0

r=[1 1 1 1];
x=[];
for j=1:2^m-1
   % Calculation of next contents of shift register
   % The first column of S gives the feedback part of the sum
   % to go into r(1).
   % The "eye(m-1)" moves the contents of all shift register
   % cells one position to the right.
   r=mod(r*S,2);
   % Add new output to x
   x=[x r(1)];
end
% Output sequence
x =
0   0   0   1   0   0   1   1   0   1   0   1   1   1   1
```

It is now easy to set up a program (or circuit) generating q = m bits at a time:

```
r=[1 1 1 1];
Sm=mod(S^m,2);
x=[];
for j=1:8
   r=mod(r*Sm,2);
   % Each r holds m bits of output sequence in reversed order
   % oldest bit is to the right
   x=[x fliplr(r)];
end
```

```
% Output sequence
x =
0   0   0   1   0   0   1   1   0   1   0   1   1   1   1   0
0   0   1   0   0   1   1   0   1   0   1   1   1   1   0   0
```

In this 8·m = 32 bits sequence you see a little more than two periods of x. You may also check another property of m-sequences: If you take out every other bit, you get the same sequence starting from the beginning and the same sequence shifted about half a period, if you start at the second symbol.

The shift register in Figure 2.6 is an example of a discrete time system that is linear in the sense that scaling and addition are modulo 2 operations. Since the only constants are 0 and 1, multiplication is performed as connection/no connection, while additions are xor functions. With these changes we can use difference equations like (2.15) and apply a formal transform like the Z-transform (2.10) to get transfer functions. We shall not need general systems, but only finite impulse response systems (multiplication by a polynomial) and their inverses (division by a polynomial). The transform of binary sequences and systems is only a method of manipulating sequences, and there are no spectra associated with the binary sequences. Similarly there is no notion of stability for these systems. The spectrum calculated in Example 2.13 relates to the bipolar sequence, where the values are integers, and this spectrum does not follow directly from the polynomial describing the binary system.

2.6 Scrambling

It is often an advantage to have a random distribution of symbol sequences, since the correct statistical properties of various steps of processing would otherwise not be assured. One way to achieve a random like distribution is a linear preprocessing of the signal called **scrambling** (*scrambling*). As demonstrated below, this approach does not always produce the desired distribution. It is often recommended, however, since it improves the properties of many sequences that occur often in practice like long sequences of zeros or periodic sequences like digitized tones.

We shall only discuss scrambling of binary sequences $\{a_k\}$ and all calculations are modulo 2 (exclusive-or). The most rightaway solution is the structure shown in Figure 2.7. Here the same pseudorandom sequence $\{p_k\}$ is generated in the **scrambler** and descrambler. The scrambler generates $y_k = a_k + p_k$, and $y_k + p_k = (a_k + p_k) + p_k = a_k$ is recovered in the de-

scrambler. There is a need for synchronizing the two sequences, but this is possible in many communication systems where data are transmitted in frames anyway. Thus the registers may be initialized at the beginning of each frame.

The scrambling sequences may be stored or generated by a block division process to be discussed below. Long zero sequences may occur if the scrambling sequence is found in the data, but this is expected to be a very rare situation.

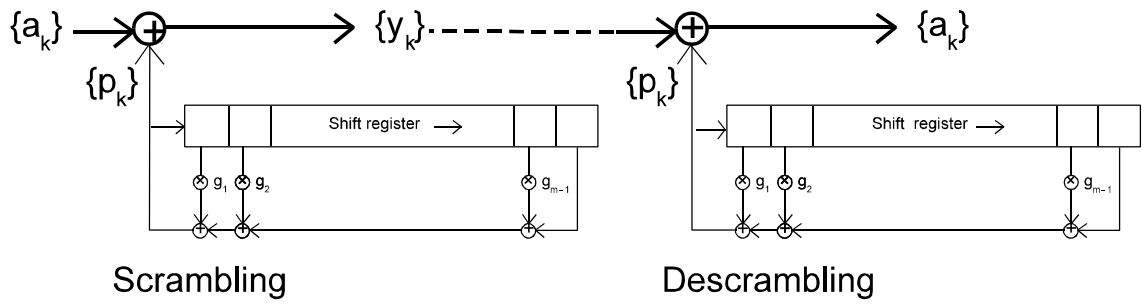

Figure 2.7
Scrambling and descrambling using two synchronized pseudorandom sequences.

To avoid the synchronization, a different structure is often used e.g. in the ITU-T recommendations [2.6]. Figure 2.8 shows a general scrambler of this self-synchronizíng type.

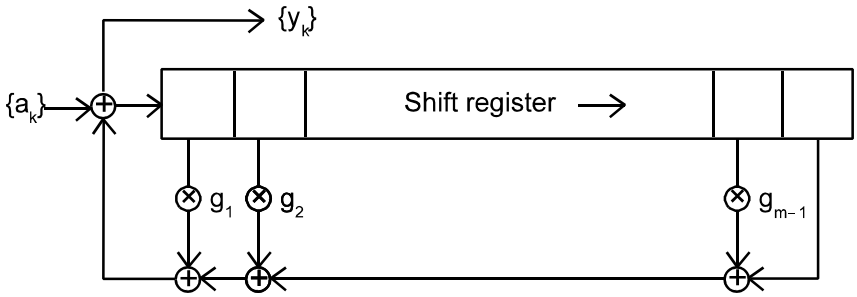

Figure 2.8
Scrambler.

Note that the only difference from the pseudorandom generators in Figures 2.6 and 2.7 is that an input has been added to influence the content of the shift register. The output sequence $\{y_k\}$ becomes

$$y_k = a_k + g_1 y_{k-1} + \ldots + g_{m-1} y_{k-(m-1)} + y_{k-m} \tag{2.33}$$

Rearranging terms we get:

$$a_k = y_k + g_1 y_{k-1} + \ldots + g_{m-1} y_{k-(m-1)} + y_{k-m}$$

and the Z-transform (2.10) gives

$$A(z) = G(z) \cdot Y(z), \quad Y(z) = \frac{A(z)}{G(z)} \tag{2.34}$$

where G(z) is the **scrambling polynomial** (*scramblingspolynomiet*)

$$G(z) = 1 + g_1 z^{-1} + \ldots + g_{m-1} z^{-(m-1)} + z^{-m}$$

The properties of the scrambling sequence depend on the polynomial G(z). It is important to have a long period before the sequence repeats, and the polynomials generating m-sequences (Table 2.4) or closely related polynomials are frequently used. Since A(z) = G(z)·Y(z), the information sequence may be recovered by polynomial multiplication, which is what happens in the self-synchronizing descrambler in Figure 2.9. The term self-synchronizing refers to the fact that no synchronization between scrambler and descrambler is required.

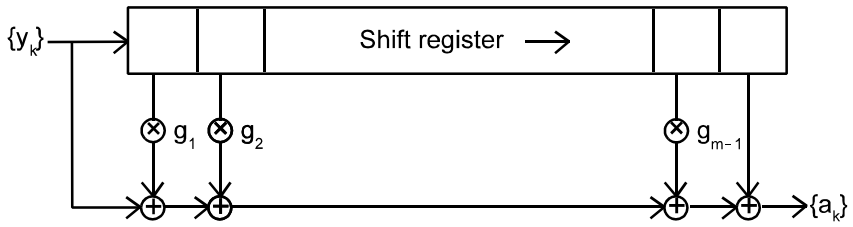

Figure 2.9
Self-synchronizing descrambler.

For a given initial state of the system, there is a one-to-one relation between input and output sequences. Thus the scrambler does not really provide independent output symbols if the input has a different distribution. In particular, A(z) = 0 gives Y(z) = 0 unless the register is initialized with a nonzero value (see the following Example 2.15). Such an initialization does not harm the self-synchronizing property. The effect of a nonzero content

is that an m-sequence of period $2^m - 1$ is added to Y(z), and multiplication of an m-sequence with the polynomial G(z) results in zero as we saw before. Another example of bad behavior is $A(z) = G(z) \cdot z^{-k}$ resulting in $Y(z) = z^{-k}$ (for an uninitialized shift register), i.e. a 1 followed by a string of 0s, since the register is driven into the zero state. This situation is called *scrambler lock-up*, but one may hope that this is a rare event. For an initialized shift register, lock-up is also possible, since there is always an input that drives the register into the zero state. Another problem occurs in case of transmission errors, since a single input error gives an output error for every coefficient in G(z) that equals 1.

However, if we assume that lock-up does not occur, and the input is a periodic sequence with an undesirable line spectrum, the output would be periodic with a much longer period. Thus even though it still has a discrete spectrum in principle, that would not be observed in a real signal. (Note that the scrambler is a linear binary system, but not linear over the real numbers. Thus relations like (2.14) and (2.29) do not apply).

Example 2.15 Scrambler and descrambler

We shall use $G(z) = 1 + z^{-4} + z^{-5} + z^{-6} + z^{-8}$. We may calculate the state of the register as in Example 2.12, but as in Example 2.14, we shall use the alternative expression in matrix form:

```
S=[eye(7);zeros(1,7)];
% Connections g1,g2,g3...,g7,1 in Figure 2.8
f=[0 0 0 1 1 1 0 1]';
S=[f S];
```

If the register is initialized to (1,1,1,1,1,1,1,1), the response to an input of **a** = (0, 1,1,0,0,1,0,1) may be calculated as

```
s=ones(1,8);
% a is input sequence
a=[0 1 1 0 0 1 0 1];
% Calculations of output
for i=1:length(a)
   % Calculation of next contents of shift register
   % (assuming zero input)
   s=mod(s*S,2);
   % s(1) is modified with the current input
   s(1)=mod(s(1)+a(i),2);
   % s(1) is also the output of the scrambler
   y(i)=s(1);
   % Formatted display of i followed by s
   fprintf(1,'%2i: ',i);
```

```
   fprintf(1,' %1i',s);
   fprintf(1,'  output: %1i\n',y(i));
end
 1:  0 1 1 1 1 1 1 1  output: 0
 2:  1 0 1 1 1 1 1 1  output: 1
 3:  1 1 0 1 1 1 1 1  output: 1
 4:  0 1 1 0 1 1 1 1  output: 0
 5:  1 0 1 1 0 1 1 1  output: 1
 6:  0 1 0 1 1 0 1 1  output: 0
 7:  1 0 1 0 1 1 0 1  output: 1
 8:  0 1 0 1 0 1 1 0  output: 0
```

The recovery of the data with the circuit of Figure 2.9 may be done with a straightforward implementation:

```
% The shift register is initialized with ones
s=ones(1,8);
% Calculations of data
for i=1:length(y)
   p=s*f;
   % Shift 1 position to the right
   % s(1) is modified with the current input
   s=[y(i) s(1:7)];
   % Output data
   aa(i)=mod(y(i)+p,2);
end
aa
aa =
     0     1     1     0     0     1     0     1
```

For high data rates the bitwise processing (as in Example 2.15) of the signal is not desirable. However, we can obtain the same result with a block division and multiplication. For this purpose we write the linear system in matrix form (representation of linear systems as first order vector/matrix systems is a standard approach in control and other areas of systems theory):

$$\mathbf{y}_k = \mathbf{s}_k\mathbf{f} + \mathbf{a}_k$$
$$\mathbf{s}_{k+1} = \mathbf{s}_k\mathbf{S} + \mathbf{a}_k\mathbf{e}$$

where the vectors and symbols are from Example 2.15, and **e** is a vector used for updating the state with the input, (1,0,0,0,0,0,0,0). So far we have assumed a scalar input and output. If we consider a block of n input symbols, $\mathbf{a}_k$, we can find the output, $\mathbf{y}_k$, and the final state by repeated application of these equations. From Figure 2.8 it is seen that the final state is last n bits of the output reversed. Due to this correspondence and the linearity of the system, we can express the relations as just one equation

$$\mathbf{y}_k = \mathbf{s}_k\mathbf{V} + \mathbf{a}_k\mathbf{U}$$

We shall demonstrate the details of determining **V** and **U** in the following example.

Example 2.16 High speed scrambler and descrambler

Since the first term is the output for zero input, we may find the matrix **V** by reversing the columns of $\mathbf{S}^8$ since the output is the state reversed. The second term is the output for a zero initial state. Thus the first row of **U** is the first 8 bits of the impulse response, and the other rows are obtained by shifting the impulse response to the right. The impulse response is found as the output when (1,0,0,....) is applied.

```
V=fliplr(mod(S^8,2))
V =
     0     0     0     1     1     1     0     0
     0     0     1     1     1     0     0     0
     0     1     1     1     0     0     0     1
     1     1     1     0     0     0     1     0
     1     1     0     1     1     0     0     0
     1     0     1     0     1     1     0     1
     0     1     0     0     0     1     1     1
     1     0     0     0     1     1     1     0

% First row of U is impulse response
U=[1 0 0 0 1 1 1 0];
for i=2:8
   U(i,:)=[0 U(i-1,1:7)];
end
U
U =
     1     0     0     0     1     1     1     0
     0     1     0     0     0     1     1     1
     0     0     1     0     0     0     1     1
     0     0     0     1     0     0     0     1
     0     0     0     0     1     0     0     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     1

% The shift register is initialized with 1s
s=ones(1,8);
% a is one input block
a=[0 1 1 0 0 1 0 1];
y=mod(s*V+a*U,2)
y =
     0     1     1     0     1     0     1     0

s=fliplr(y)
s =
     0     1     0     1     0     1     1     0
```

The output and the final state correspond to the results in Example 2.15.

In the descrambler, the relation becomes

$$\mathbf{a}_k = \mathbf{s}_k\mathbf{U}\mathbf{r} + \mathbf{y}_k\mathbf{V}\mathbf{r}$$

The final state is equal to the reversed input, and of course the two registers should be in the same state. As before, **Ur** may be found from the impulse response. But since the state vector consists of past inputs, the rows are obtained by shifting the impulse response to the left. **Vr** is also the impulse response, but shifted right.

```
% First row of Ur is impulse response of Figure 2.9
Ur=f';
for i=2:8
   Ur(i,:)=[Ur(i-1,2:8) 0];
end
Ur
Ur =
     0     0     0     1     1     1     0     1
     0     0     1     1     1     0     1     0
     0     1     1     1     0     1     0     0
     1     1     1     0     1     0     0     0
     1     1     0     1     0     0     0     0
     1     0     1     0     0     0     0     0
     0     1     0     0     0     0     0     0
     1     0     0     0     0     0     0     0

Vr=[1 f(1:7)'];
for i=2:8
   Vr(i,:)=[0 Vr(i-1,1:7)];
end
Vr
Vr =
     1     0     0     0     1     1     1     0
     0     1     0     0     0     1     1     1
     0     0     1     0     0     0     1     1
     0     0     0     1     0     0     0     1
     0     0     0     0     1     0     0     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     1
% The shift register is initialized with ones
s=ones(1,8);
% Recovery of received y
aa=mod(s*Ur+y*Vr,2)
aa =
     0     1     1     0     0     1     0     1
```

For implementations it is often easier to precalculate the matrices and store them rather than to calculate the matrix products each time.

2.7 CRC checks

The same binary polynomials as discussed above may be used in a different way to protect a frame against transmission errors. As a first step we note that a single error can be detected by adding a parity bit to the frame. Thus the entire frame has even parity, an even number of ones. If the binary symbols of the sequence $\{c_k\}$ are read as a polynomial[1]

$$C(x) = c_{N-1}x^{N-1} + \cdots + c_1x + c_0$$

we may express this by saying that (x+1) is a factor of C(x) (Why?). We could introduce this factor by multiplying the original sequence by x+1, but we usually prefer to leave the data unchanged, and add the parity bit. We may obtain a better protection by requiring that the frame as a polynomial has a less trivial polynomial

$$G(x) = g_mx^m + \cdots + g_1x + g_0$$

as a factor.

An immediate method for encoding is to use the N−m bits as coefficients in a polynomial K(x) (degree N−m−1) and then multiply by G(x) giving the degree N−1 polynomial C(x). As above, the immediate method has the drawback that the data are not seen directly in the encoded sequence so we will accomplish this by dividing the data sequence (leaving it unchanged) with G(x) and adding the remainder from this division of degree at most m−1. The added remainder cancels out the remainder from the data sequence so the net result is that G(x) divides C(x). Thus we calculate

$$C(x) = K(x)\cdot x^m + \text{Remainder}\left(\frac{K(x)\cdot x^m}{G(x)}\right) = K(x)\cdot x^m + CRC(x)$$

where the remainder, CRC(x), is known as a **Cyclic Redundancy Check**, **CRC**. The process is known as systematic encoding. Thus the N−m information bits are found immediately as the highest N−m powers followed by the CRC as a **redundancy** (*redundans*) with m bits as the lower powers since the remainder from division by G(x) always has degree at most m−1. Since G(x) divides C(x), C(X) is a "legal" sequence, a

---

[1] Note a slight change of notation where x has replaced z^{-1}. This is usually the case when dealing with error-detection (and correction). Usually the high powers are transmitted first and we write the polynomials with the highest power first.

codeword. If the division is programmed, the most immediate method is close to the method used when dividing polynomials by hand. In hardware it may be realized by a circuit similar to Figure 2.8 where we are not interested in the quotient, but the remainder which is formed by moving the switches and performing m more shifts. The circuit is shown in Figure 2.10. In many applications this is not fast enough and a block division approach like in Example 2.16 will be used. We have not yet explained the reason for "cyclic" in the name. It turns out that if G(x) divides the polynomial $x^N + 1$, then the codewords using this G(x) have a cyclic property, i.e. $(c_{N-1}, c_{N-2}, ..., c_1, c_0)$ being a codeword implies that the cyclic shift $(c_{N-2}, ..., c_1, c_0, c_{N-1})$ is also a codeword. In many cases the check procedure is used for sequences much shorter than the N for which it is cyclic.

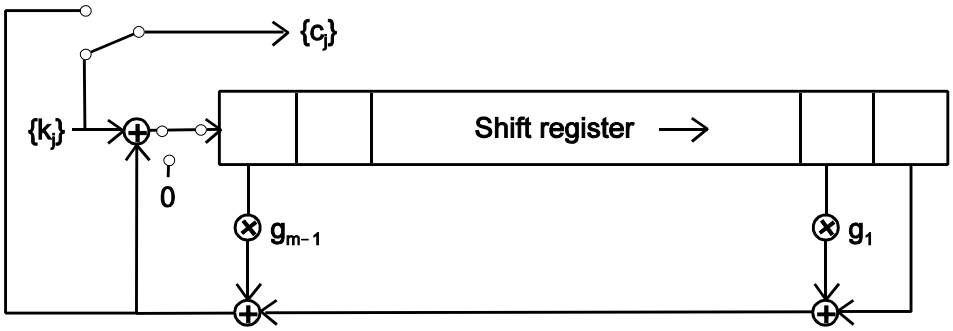

Figure 2.10
CRC encoder. The shift register is initialized with 0...0. Both switches are shown in input position and they are moved after input of N−m symbols.

The receiver may check that the frame is error free by dividing it by G(x). The remainder of this division is called a **syndrome** (*syndrom*). A single error in position j may be described as $C(x) + x^j$, since adding (exclusive-or) x^j just inverts position j. Receiving this and calculating the remainder of the division by G(x) gives a syndrome that only depends on x^j since the remainder from C(x) is zero due to the way we constructed C(x). Likewise, more errors result in a nonzero syndrome if the error positions do not unfortunately give a polynomial divisible by G(x). Thus any nonzero syndrome indicates that a transmission error has occurred, and the probability of getting a zero syndrome for a random sequence is 2^{-m}, which can be made suitably small by taking m big enough. For an error sequence with error probability less than 1/2 one may expect that the probability gets better. This is the case for good polynomials, but rather complicated to figure out [2.7]. Figure 2.11 shows a circuit that may create the syndrome from a received sequence with some errors ($e_j = 1$ for some positions and 0 otherwise).

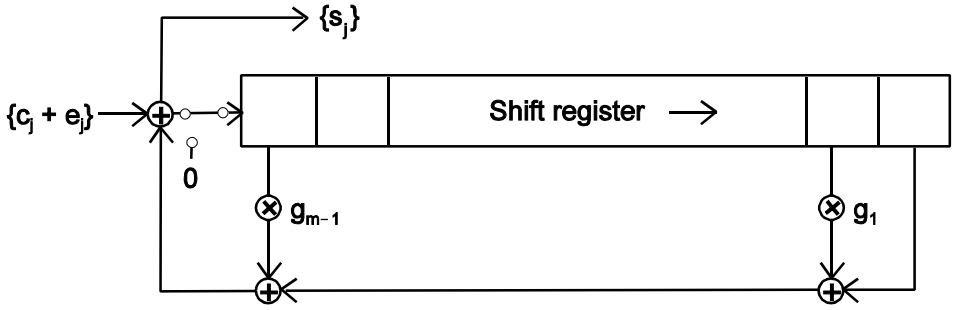

Figure 2.11
Syndrome former for CRC. The shift register is initialized with 0...0. The switch is shown in the initial position and it is moved after input of N−m symbols. The last m shifts then produce the m bits of the syndrome.

If these syndromes (remainders) are different for different error patterns, we are able to correct the errors. The question is now how G(x) should be selected to obtain this. This is the starting point of the technique of error-correcting codes, but we shall not pursue this subject here (cf. the DTU course in Error-correcting codes, [2.8]). If we select G(x) to be one of the polynomials generating m-sequences (e.g. from Table 2.4), we may compare the division process of Figure 2.11 with that of generating m-sequences (Figure 2.6), and we see that it is the same process. Since the circuitry is linear the content of the shift register is always a sum of the remainder for C(x) and a possible remainder for x^j. Since the first one will end up as zero after N shifts, we only have to consider the remainder for x^j. Before position in error, j, this is zero, but when x^j enters into the register, the register keeps on calculating remainders, as if it was initialized with 10...00. We see that the polynomials generating m-sequences have the right properties for G(x) since the shift register passes through all different states, and after j shifts a remainder comes out that is unique dependent on j. Codes with these generator polynomials were the first error-correcting codes found (the so-called Hamming codes). A complete decoder may be based on the syndrome former of Figure 2.11 which is allowed to cycle through the states twice instead of being emptied with 0s and a comparator telling when the error position is reached.

Two errors will produce a syndrome that is the sum of the two individual remainders, and thus it will be nonzero, i.e. double errors can always be detected. Thus the length of the frame, including the CRC check can be at most $N = 2^m - 1$. However we often combine this approach with the single parity check, and take G(x) as a product of an m-polynomial and (x+1). In this case an extra redundant symbol is needed, but any combination of 3 errors can be detected. It is also easy to see that any error of length at most m can be detected. The 16

bit CRCs in the Table 2.5 are constructed in this way. The Table 2.5 (partly from [2.9]) shows some commonly used check polynomials.

It might appear that the scrambling and the addition of a CRC should be combined and the same polynomial might be used for this purpose. However, these functions belong to different levels of the communication process. The CRC is added to the data before transmission to ensure the integrity of the received frame, whereas the scrambling occurs in the modem or similar equipment immediately before transmission. On the other hand these two functions might interfere with each other in unexpected ways if the scrambling process introduced special error patterns.

N	N−m	Polynomial	Checks up to	Name
all	N−1	$x+1$	1 error	Parity check bit
7	4	x^3+x+1	2 errors	Hamming code
7	3	$x^4+x^3+x^2+1$	3 errors	Mult. by x+1
15	8	$x^7+x^6+x^4+1$	3 errors	CRC-7
93	85	$x^8+x^7+x^6+x^4+x^2+1$	3 errors	CRC-8
2047	2035	$x^{12}+x^{11}+x^3+x^2+x+1$	3 errors	CRC-12
32767	32751	$x^{16}+x^{12}+x^5+1$	3 errors	CRC-ITU-T
32767	32751	$x^{16}+x^{15}+x^2+1$	3 errors	CRC-16, CRC-ANSI
$2^{32}-1$	$2^{32}-33$	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	2 errors	CRC-32, Hamming code (Ethernet frames)

Table 2.5
Examples of Cyclic Redundancy Checks. Some examples taken from [2.9].

Clearly CRC on a long frame will have little influence on the properties of the transmitted signal. However, we note that in general, CRCs and other forms of parity checks do not change the spectrum of independent data since the CRC bits themselves become independent.

Example 2.17 Simple Hamming code

As in the previous simple example, we shall use $G(x) = x^4 + x + 1$, which has period 15. Thus we may use it as a check polynomial for a data frame of at most $15 - 4 = 11$ bits and we shall also demonstrate the ability to correct one error. Instead of the circuits in Figures 2.10 and 2.11 we shall use MATLAB functions `conv` and `deconv` which result in integers, so a modulo 2 operation is done afterwards. The reader may convince himself that this will work as long as the data sequences are not too long (15 here is very small):

```
G=[1 0 0 1 1];
m=length(G)-1;  % Degree of G
K=[1 0 0 1 0 1 1 1 0 1 1]; % Information bits
N=length(K)+m;  % Length of codeword
% Get remainder R from division by G
[Q,R]=deconv([K zeros(1,m)],G);
% Create codeword
C=[K mod(R(end-m+1:end),2)]
C =
1   0   0   1   0   1   1   1   0   1   1   0   1   1   0
```

We may find the syndrome corresponding to an error in position j as the remainder for x^j modulo G(x) (thus the highest power of x is transmitted first). We first see that the syndrome for no error is 0000 and we then produce a list af syndromes for positions 1 to 15:

```
% No errors should give syndrome 0000
[Q,R]=deconv(C,G);
syndrome=zeros(N+1,m);
syndrome(1,:)=mod(R(end-m+1:end),2);
% One error
for j=1:N
    err=zeros(1,N);
    err(j)=1;
    % Received sequence
    Y=xor(C,err);
    [Q,R]=deconv(Y,G);
    syndrome(j+1,:)=mod(R(end-m+1:end),2);
end
syndrome =
     0     0     0     0
     1     0     0     1
     1     1     0     1
     1     1     1     1
     1     1     1     0
     0     1     1     1
     1     0     1     0
     0     1     0     1
     1     0     1     1
     1     1     0     0
```

```
     0     1     1     0
     0     0     1     1
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

The 16 syndromes are seen to be different so we may recognize that no error occurred (0000) or we may correct one error in the 15 positions. Two errors give one of these syndromes as well, so we are not able to correct more than one error. If used for error detection only, one or two errors may be detected if the syndrome is non-zero. Some patterns of more errors may be detected as well, but there exist some 3-error patterns which are not detectable (Why?). In order to detect at least 3 errors, we may include the factor (x+1) to obtain the polynomial G2 (x)= x^5 + x^4 + x^2 + 1. In this case the data frame can be at most 10 bits. We again list the syndromes for zero and one error:

```
G2=mod(conv([1 0 0 1 1],[1 1]),2)
G2 =
1   1   0   1   0   1
m=length(G2)-1;  % Degree of G2
K=[1 0 0 1 0 1 1 1 0 1]; % Information bits - one less
N=length(K)+m;  % Length of codeword
[Q,R]=deconv([K zeros(1,m)],G2);
C=[K mod(R(end-m+1:end),2)]
C =
1   0   0   1   0   1   1   1   0   1   0   0   1   0   1
% No errors should give syndrome 00000
[Q,R]=deconv(C,G2);
syndrome=zeros(N+1,m);
syndrome(1,:)=mod(R(end-m+1:end),2);
% One error
for j=1:N
    err=zeros(1,N);
    err(j)=1;
    % Received sequence
    Y=xor(C,err);
    [Q,R]=deconv(Y,G2);
    syndrome(j+1,:)=mod(R(end-m+1:end),2);
end
syndrome =
     0     0     0     0     0
     1     1     0     1     0
     0     1     1     0     1
     1     1     1     0     0
     0     1     1     1     0
     0     0     1     1     1
     1     1     0     0     1
     1     0     1     1     0
     0     1     0     1     1
     1     1     1     1     1
     1     0     1     0     1
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

The syndromes are again distinguishable, but two errors now produce a syndrome outside this list, e.g. errors in position 5 and 15 give 0 0 1 1 0, so this situation may be detected as it is the case of all other combinations of two errors.

2.8 Measurements of impulse responses

Many communication systems are linear, and knowledge of the transfer function or impulse response is important for the receiver. The transfer function or the impulse response of a linear system may be measured by observing simultaneous input and output signals. In communication systems this may be difficult to do, but if the input signal is known from the method of generating it, only the output has to be observed. In order to get a correct measurement we must use input signals which contain sufficient energy for all relevant frequencies, and in the time domain the output must be observed long enough for the impulse response to vanish. From the definition of impulse response it follows that the obvious choice for the input signal is an impulse function (delta function, (2.11) for a discrete time linear system). However, this is often not possible since it has all its energy confined to a small time interval, and many systems have such a large attenuation that much input energy is required in order to observe anything at the output. Normally it is better to spread the energy over a larger time interval, and thus a better choice is a periodic input provided that the period (denoted NT in the following) is longer than the impulse response. Even though only some frequencies, $\omega_m = m \cdot 2\pi/(NT)$, are contained in the signal, a reasonable approximation to the impulse response may be calculated as the solution to a system of linear equations found from applying (2.13) and (2.14) for all the frequencies:

$$H(\omega_m) = \frac{G(\omega_m)}{F(\omega_m)} = \sum_{k=0}^{N-1} Th_k e^{-j\omega_m kT}, \quad m = 0,\ldots,N-1$$

where $G(\omega_m)$ and $F(\omega_m)$ are the transforms of output (**g**) and input (**f**), respectively. Several problems are found in this calculation. We have assumed that enough energy is found at all frequencies, such that the denominator is non-zero, but the division of the complex numbers may be difficult to implement. The solution of the equations to get h_k may be done with the inverse DFT. In the following we shall present a method that simplifies the calculations.

In some applications the input is selected as a pseudorandom sequence. By this term, we mean a signal where all the input frequencies have the same power, which we shall denote by A/N (= $S_f(\omega)$). Note that the pseudorandom sequences presented in Section 2.5 do not have this property for the frequency zero. We shall return to this in Example 2.18. We find the transfer function for the frequency ω_m by conjugating (2.30) as

$$H(\omega_m) = \frac{S_{fg}^*(\omega_m)}{S_f^*(\omega_m)} = \frac{S_{fg}^*(\omega_m)}{A/N} = \frac{N}{A}\frac{1}{N}\sum_{k=0}^{N-1} R_{fg}(kT)e^{j\omega_m kT} = \frac{1}{A}\sum_{k=0}^{N-1} R_{gf}(kT)e^{-j\omega_m kT}$$

since $R_{fg}(kT) = R_{gf}(-kT)$ and R_{fg} is periodic with period N. Comparing with (2.13) we see that it is easy to calculate

$$h_k = \frac{R_{gf}(kT)}{AT} = \frac{1}{ATN}\sum_{n=0}^{N-1} g_{n+k} f_n, \quad k = 0,...,N-1 \tag{2.35}$$

i.e. we find the impulse response from a (periodic) correlation of the input and output sequence, thus avoiding division and complex numbers. We have used the definition (2.28) of the periodic crosscorrelation.

Example 2.18 - Measurement of unknown impulse response with m-sequences

Often m-sequences are used as inputs. These sequences have constant power spectra except for $S_f(0)$, which is small. For N = 15 the spectrum was calculated in Example 2.13 as

$$S_f(m\frac{2\pi T}{15}) = \begin{cases} \frac{1}{225} & \text{for } m = 0 \\ \frac{16}{225} & \text{for } 1 \le m \le 14 \end{cases}$$

The reduced power at the frequency zero is often of no practical importance, since many electrical systems are known to have transfer functions where H(0) = 0. If $S_f(m\cdot 2\pi/NT) = A/N$ for $1 \le m \le N-1$ the solution for $h_1, h_2, \ldots, h_{N-1}$ was given as (2.35). Going through the calculation with the assumption that H(0) = 0, we get

$$0 = H(0) = \frac{S_{fg}^*(0)}{S_f^*(0)} = \frac{1}{S_f^*(0)N}\sum_{k=0}^{N-1} R_{gf}(kT)e^{-j0kT} \Rightarrow \sum_{k=0}^{N-1} R_{gf}(kT) = 0$$

$$0 = H(0) = \sum_{k=0}^{N-1} Th_k \Rightarrow h_0 = -\sum_{k=1}^{N-1} h_k = -\frac{1}{AT}\sum_{k=1}^{N-1} R_{gf}(kT) = \frac{1}{AT}R_{gf}(0)$$

so it turns out that (2.35) also works for k = 0. As an example let us assume that some H(ω) is unknown except for H(0) = 0 and that the input is a bipolar m-sequence with N = 15. For an input with period

$$\mathbf{f} = (-1,-1,-1,\ 1,-1,-1,\ 1,\ 1,-1,\ 1,-1,\ 1,\ 1,\ 1,\ 1)$$

the system produces a periodic output with period

$$\mathbf{g} = (\ 2,-6,-2,-2,\ 8,-4,-4,\ 6,\ 4,-8,\ 6,-6,\ 4,\ 2,\ 0)$$

We get

$$h_k = \frac{15}{16}\cdot\frac{1}{15}\sum_{n=0}^{14} g_{n+k} f_n = \begin{cases} -1 \text{ for } k = 0 \\ \ \ 4 \text{ for } k = 1 \\ -2 \text{ for } k = 2 \\ -1 \text{ for } k = 3 \\ \ \ 0 \text{ otherwise} \end{cases}$$

The assumption that h vanishes within a period of the signal is seen to be satisfied and it is also clear that $H(0) = \Sigma Th_k = 0$.

In MATLAB the calculations are easily done:

```
f=[-1 -1 -1 1 -1 -1 1 1 -1 1 -1 1 1 1 1];
N=length(f);
g=[2 -6 -2 -2 8 -4 -4 6 4 -8 6 -6 4 2 0];
T=1;
A=(N+1)/N;              % S_f for an m-sequence is (N+1)/N^2
h=xcorr(g,[f f])/A/T/N % (2.35) and periodic
                        % autocorrelation from Example 2.10

h =
    0.1250   -0.2500   -0.3750   -0.5000   -0.2500    0.5000
   -0.5000    0.8750   -0.6250    0.1250    0.0000    0.3750
    0.1250   -0.0000   -1.0000    4.0000   -2.0000   -1.0000
   -0.0000    0.0000   -0.0000   -0.0000    0.0000   -0.0000
```

```
    0.0000    0.0000    0.0000    0.0000   -0.0000   -1.0000
    3.8750   -1.7500   -0.6250    0.5000    0.2500   -0.5000
    0.5000   -0.8750    0.6250   -0.1250   -0.0000   -0.3750
   -0.1250    0.0000    0.0000    0.0000         0   -0.0000
         0   -0.0000   -0.0000    0.0000    0.0000   -0.0000
    0.0000   -0.0000   -0.0000   -0.0000   -0.0000
h=h(N:2*N-1)           % periodic correlation from Example 2.10

h =
   -1.0000    4.0000   -2.0000   -1.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000
```

Since the correlation calculations of (2.35) become time consuming for large N, it is often preferable to calculate the (conjugated) cross power spectrum using (2.30)

$$S_{fg}^*(\omega) = H(\omega)S_f^*(\omega) = H(\omega)F(\omega)F^*(\omega) = G(\omega)F^*(\omega)$$

and then find R_{gf} in (2.35) with the Inverse Fourier transform (with an efficient implementation, FFT, which exists for some N, e.g. powers of 2):

```
F=fft(f)/N;
G=fft(g)/N;
SfgConj=G.*conj(F)
SfgConj =
         0             0.0716 + 0.0576i   0.1916 - 0.0281i
   0.1894 - 0.2287i    0.0163 - 0.3801i  -0.2133 - 0.3695i
  -0.3672 - 0.2348i   -0.4217 - 0.0752i  -0.4217 + 0.0752i
  -0.3672 + 0.2348i   -0.2133 + 0.3695i   0.0163 + 0.3801i
   0.1894 + 0.2287i    0.1916 + 0.0281i   0.0716 - 0.0576i
h=N*ifft(SfgConj)/A/T
h =
  -1.0000    4.0000   -2.0000   -1.0000    0.0000
  -0.0000    0.0000         0         0   -0.0000
         0    0.0000   -0.0000   -0.0000   -0.0000
```

The result is seen to be the same. The same result is of course also obtained by

```
h=ifft(G./F)/T
```

which contains division of complex numbers.

2.9 References for Chapter 2

2.1 B.P. Lathi: "Signal Processing and Linear Systems". ISBN 0-19-521917-1, Oxford University Press N.Y., 1998.
(Used in the DTU courses 31605 and 31606: Signals and Linear Systems)

2.2 J.G. Proakis & M. Salehi: “Communication Systems Engineering (2nd Edition)”, ISBN 0-13-095007-6, Prentice Hall, 2002.

2.3 J.G.Proakis & D.G. Manolakis: “Digital Signal Processing (4th Edition)”, ISBN 0-13-187374-1, Prentice Hall, 2006.
(Used in DTU course 02451: Digital Signal Processing)

2.4 R.H. Barker: “Group Synchronization of Binary Digital Systems, Communication Theory”, Academic Press, 1953.

2.5 S.W. Golomb and G. Gong: “Signal Design for Good Correlation. For Wireless Communication, Cryptography, and Radar”, ISBN 978-0-521-82104-9, Cambridge University Press, 2005.

2.6 ITU-T: "Recommendations from IXth Plenary Assembly, Melbourne 1988, Blue Book, Recommendation G.703", ISBN 92-61-03341-5, International Telecommunication Union, Geneva 1989.

2.7 T. Kløve and V.I. Korzhik: “Error Detecting Codes”, ISBN 0-7923-9629-4, Kluwer Academic Publishers 1995.

2.8 J. Justesen and T. Høholdt: “A Course in Error-Correcting Codes, Second edition”, ISBN 978-3-03719-179-8, European Mathematical Society Publishing House 2017.
(Used in DTU course 01405 Error-correcting Codes)

2.9 D.J. Costello, J. Hagenauer, H. Imai, and S.B. Wicker: “Applications of Error-Control Coding”, IEEE Transactions on Information Theory, ISSN 0018-9448, Vol. IT-44, October 1998, pp. 2531-2560.

3. STOCHASTIC SIGNALS

3.1 Basic concepts

We shall assume that the reader has some background in probability and/or statistics. An introductory text may be [3.1]. Thus only the required terminology will be introduced here.

Stochastic signals (*stokastiske signaler*) are usually referred to as voltages that vary as functions of time. Other physical variables could be used, but it may cause considerable difficulties to leave out the physical dimensions. We shall describe the stochastic signal at time t as a **random variable** (*stokastisk variabel*), X(t). We note that the abstract definitions of probability theory are not always translated to technical terms in any simple way, and some caution may be needed in interpreting the models.

A measurement of X(t) gives an **outcome** (*udfald*), x(t), which is a real number in some range. The set of possible numbers is called the **sample space** (*udfaldsrummet*). Many of the signals we consider will have a finite set of values, and the sample space is referred to as **discrete** (that term also includes the case of a countably infinite set like the integers).

A **probability measure**, P, assigns values in the interval [0, 1] to events, which are certain subsets of the sample space. Thus for a discrete space, P(X = a) indicates the probability (*sandsynlighed*) that the signal has outcome a. For a finite sample space $\{a_1, a_2, ..., a_N\}$ it may be convenient to think of the probabilities as a vector

$$\mathbf{P}(X) = (P(X=a_1), P(X=a_2), \ldots, P(X=a_N))$$

P(X) is sometimes known as the **probability mass function** and denoted **pmf**. An important property of a probability measure is that

$$\sum_{i=1}^{\infty} P(X = a_i) = 1$$

A stochastic variable assuming a continuous set of values, can be characterized by its **probability distribution function**, F (*fordelingsfunktion*)

$$F(\alpha) = P(x \leq \alpha)$$

The probability distribution function is sometimes called the cumulative distribution function and abbreviated as **cdf**. In most cases of interest to us it is also possible to define a **probability density function**, f (*tæthedsfunktion*)

$$f(\alpha) = \frac{dF}{d\alpha} \quad \Rightarrow \quad F(\alpha) = \int_{-\infty}^{\alpha} f(x)\,dx$$

The probability density function is sometimes abbreviated **pdf**. Similar to the sum of $P(X=a_i)$ over all i, we here have $F(\infty) = 1$.

We shall often need the **expected value** (*forventningsværdien*) or **mean** (*middelværdi*) of a stochastic signal

$$E[X(t)] = \int_{-\infty}^{\infty} x(t) f(x)\,dx$$

and similarly for a discrete space

$$E[X(t)] = \sum_{a_i} a_i P(X(t) = a_i)$$

It is important to note that this should not be interpreted as a time average since each sample signal is going to be a function of time. In order to get a physical interpretation it may be useful to think of the average as computed over a large number of signals carried in the same cable. Of course we are going to be interested in time averages also, and in many cases the two values will coincide. However, whether that is true depends on the sample space, and we may want to construct different sample spaces depending on the problem we are studying.

Expected values may also be used for functions of a stochastic signal, e.g. $E[X^k(t)]$ defines the so-called k'th moment of X (k =1 is the mean). A property of particular importance here is the **variance** (*variansen*) of the signal

$$\sigma_X^2 = E[(X - E[X])^2] = E[X^2] - E[X]^2$$

where σ_X is known as the **standard deviation** (*standardafvigelsen, spredningen*). In most cases we may assume E[X] = 0, and the variance is simply the expectation of the square value. In physical terms we shall also refer to this as the **power of the signal** as in Section 2.4, although this is a slight abuse of physical dimensions (the missing impedance will again be neglected throughout, thus the power here has unit V^2 rather than W).

The definitions of probabilities may of course also involve more than one stochastic variable. For two variables, X(t) and Y(t), we have

- **Joint probability** (*simultan sandsynlighed*) for discrete variables, P(X,Y),

$$\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} P(X = a_i, Y = b_j) = 1$$

- **Joint distribution function** (*simultan fordelingsfunktion*) for continuous variables, $F(\alpha,\beta) = P(x \le \alpha, y \le \beta)$, $F(\infty,\infty) = 1$
- **Joint probability density function** (*simultan tæthedsfunktion*) if it exists,

$$f(\alpha,\beta) = \frac{\partial^2 F(\alpha,\beta)}{\partial\alpha\,\partial\beta}$$

The variables may have some relation to each other, e.g. knowing $Y = b_j$ may give information on X, so we define the **conditional probability** (*betinget sandsynlighed*) as

$$P(X|Y) = \frac{P(X,Y)}{P(Y)} \quad \Leftrightarrow \quad P(X|Y)P(Y) = P(X,Y)$$

An important property among stochastic variables is **independency** (*uafhængighed*) which is found when the joint probability or density function factors into the product of the individual probabilities or density functions, respectively. For two independent variables we have

$$P(X|Y) = \frac{P(X,Y)}{P(Y)} = \frac{P(X)P(Y)}{P(Y)} = P(X)$$

Other relations between stochastic two stochastic variables, X(t) and Y(t), are the **correlation** (*korrelation*) of X and Y

$$R_{XY}(t) = E[X(t)Y(t)]$$

and the **covariance** (*kovarians*) of X and Y:

$$C_{XY}(t) = E[(X(t)-E[X(t)])(Y(t)-E[Y(t)])]$$

Two variables with zero covariance are called **uncorrelated** (*ukorrelerede*). Most of the signals that we deal with in this course have zero mean and thus correlation and covariance are identical. Note that independency of two variables also implies that they are uncorrelated since

$$E[X(t)Y(t)] = \sum_{X,Y} P(X,Y)X(t)Y(t) = \sum_{X} P(X)X(t)\sum_{Y} P(Y)Y(t) = E[X]E[Y]$$

However, the opposite implication does not hold.

Sometimes it is necessary to calculate mean and variance of a sum of several stochastic variables and the following relations hold

$$\begin{aligned} E\left[\sum_i c_i X_i\right] &= \sum_i c_i E[X_i] \\ \text{Variance}\left[\sum_i c_i X_i\right] &= \sum_i c_i^2 \sigma_{X_i}^2 + \sum_i \sum_{j\neq i} c_i c_j C_{X_i X_j} \end{aligned} \tag{3.1}$$

where c_i are constants. Note that variances for uncorrelated variables are just summed.

Mean values of products of more than two variables can be calculated from the correlation functions. We shall only show an example:

$$E[X_1X_2X_3X_4] = E[X_1X_2]E[X_3X_4]+E[X_1X_3]E[X_2X_4]+E[X_1X_4]E[X_2X_3]$$

In this course, more examples of continuous distribution functions will be found, but first we shall treat two very important cases.

The first and simple case is that of a variable being uniformly distributed over some interval of length Δ. **Uniform distribution** (*rektangulær fordeling*) means that the probability

density, f(x), is constant over the interval and since F(∞) = 1, we get f(x) = 1/Δ. Let us for a moment assume that the interval is $0 \le x \le \Delta$. We then get

$$E[X(t)] = \int_{-\infty}^{\infty} x(t) f(x)\, dx = \int_{0}^{\Delta} x(t) \frac{1}{\Delta} dx = \frac{\Delta}{2}$$

the midpoint in the interval as you would expect. The variance is

$$\sigma_X^2 = E[X^2] - E[X]^2 = \int_{0}^{\Delta} x^2(t) \frac{1}{\Delta} dx - \left(\frac{\Delta}{2}\right)^2 = \frac{\Delta^2}{12}$$

The other important case is often used as a model of random noise: **Gaussian** (*Gaussisk*) also known as **normal distribution** (*normalfordeling*). A zero mean Gaussian variable with variance σ^2 has density function

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{3.2}$$

The density is shown for various σ in Figure 3.1. The corresponding distribution function

$$F(x) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{x} e^{-\frac{z^2}{2\sigma^2}} dz$$

cannot be expressed in terms of basic functions. It is found in tables either directly or may be calculated from tables of erf(x) which is related to F(x) by

$$F(x) = \frac{1}{2} + \frac{1}{2} \mathrm{erf}\left(\frac{x}{\sqrt{2}\sigma}\right)$$

In this course we shall make much use of the so-called **Q-function**, Q(x), giving the probability that a Gaussian variable (with variance $\sigma^2 = 1$) is larger than x:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} e^{-\frac{z^2}{2}} dz \underset{\sigma=1}{=} F(-x) \underset{\sigma=1}{=} 1 - \frac{1}{2} - \frac{1}{2} \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right) = \frac{1}{2} \mathrm{erfc}\left(\frac{x}{\sqrt{2}}\right) \tag{3.3}$$

where erfc(x) = 1 − erf(x). Note that the probability of being larger than x for a Gaussian variable with variance σ^2 is calculated as Q(x/σ).

Q(x) is found in MATLAB directly, `qfunc`, as the functions erf(x) and erfc(x) also are. Some calculators may also have erf(x), erfc(x) or Q(x) directly. If such assistance is not available, we may use the approximation

$$Q(x) \simeq \frac{1}{x\sqrt{2\pi}} e^{-x^2/2}, \quad \text{good for } x > 3 \tag{3.4}$$

In Figure 3.1 we show how Q(2) is calculated on basis of the density function and the distribution function, both for $\sigma = 1$.

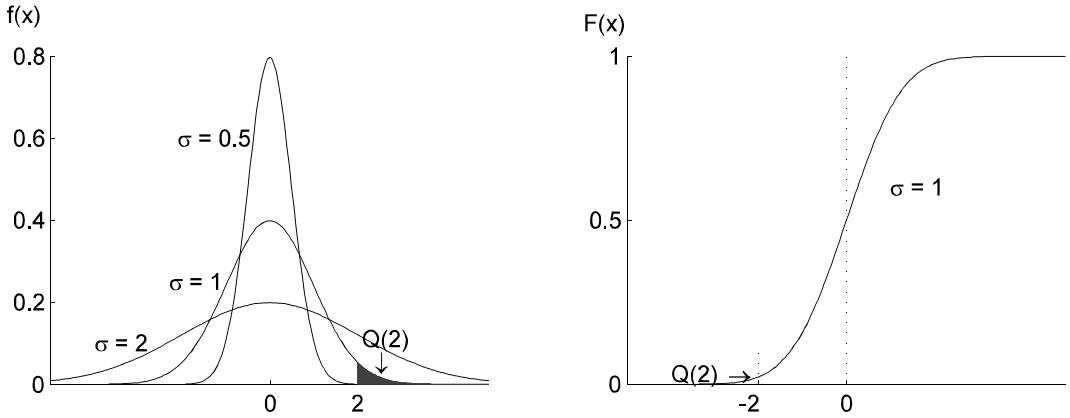

Figure 3.1
Densities for Gaussian variables with three different standard deviations.
Distribution function for Gaussian variable with standard deviation 1.
Q-function shown as area (left-hand figure) and value (right-hand figure).

If m variables are considered, they are said to be **jointly Gaussian** if the density function satisfies

$$\mathbf{f(x)} = \frac{1}{(\sqrt{2\pi})^m \sqrt{|\mathbf{R_X}|}} e^{-\frac{1}{2}\mathbf{x}\mathbf{R_X^{-1}}\mathbf{x}'} \tag{3.5}$$

where $|\mathbf{R}_X|$ means the determinant of the correlation matrix defined by

$$\mathbf{R_X} = \left[R_{rc}\right] = \left[E[X_r X_c]\right] \tag{3.6}$$

where r and c mean row and column. We note that in particular two real Gaussian variables may be interpreted as a complex Gaussian variable, and we shall need this concept in a later section. If the complex variable (with zero mean and variance σ^2 in each coordinate) is expressed in polar coordinates, (r, Θ), the Θ becomes uniformly distributed on $0 \le \Theta \le 2\pi$ and the distribution function for r becomes the so-called **Rayleigh distribution**:

$$f(r) = \frac{r}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}}, \quad F(r) = 1 - e^{-\frac{r^2}{2\sigma^2}}, \quad r > 0 \tag{3.7}$$

Gaussian variables have a number of properties that greatly facilitate the analysis. One important property is that any linear combination of Gaussian variables is again Gaussian. If **x** is a vector of jointly Gaussian variables, and **y** is obtained by the linear mapping **y** = **Ax**, we may find the correlation matrix of **y** from (3.13) and the new density function follows.

Example 3.1 Simulation of random variables

In order to perform a simulation involving random signals, it is usually convenient to start out from a random variable uniformly distributed on the interval [0,1]. Most programming languages have this facility, but it should be noted that the statistical properties are not always good. In MATLAB the function **`rand(m,n)`** produces an m by n matrix of independent random variables between 0 and 1. To be very precise the interval is $2^{-53} \le x \le 1 - 2^{-53}$, but for common applications this is of no importance:

```
% x uniform distribution 0 to 1
x=rand(1,10)
x =
    0.9501    0.2311    0.6068    0.4860    0.8913
    0.7621    0.4565    0.0185    0.8214    0.4447
x=rand(1,1000);
mean_uni=mean(x)
mean_uni =
    0.5024
var_uni=var(x)
var_uni =
    0.0805
```

The MATLAB function `mean` is just the sum of all 1000 variables divided by 1000 which is the best estimate for the mean of the variable, and `var` works in a similar way for the variance. It is seen that even for 1000 samples, the calculated values are slightly different from the theoretical values calculated above. As for all random variables, another mean may be obtained by subtraction, and other variances may be obtained by scaling with a constant factor giving a variance scaled by the square of the factor as it is seen from the following example:

```
% x uniform distribution -5 to 5
Delta=10;
x=Delta*(rand(1,100000)-0.5);
mean_uni2=mean(x)
mean_uni2 =
    -0.0012
var_uni2=var(x)
var_uni2 =
    8.3571
```

A discrete variable with a particular probability distribution may be obtained by dividing the interval [0,1] into some intervals with lengths equal to the desired probabilities (assuming that they are not very small). A simple demonstration is how to obtain a variable with values ±1 where +1 has probability p:

```
% Independent +1/-1 variables from a with probability p for +1
p=1/4;
x=rand(1,10);
a=sign(p-x)
a =
    -1     1    -1    -1     1    -1    -1     1    -1    -1
```

For a more general distribution it may be difficult to obtain the boundaries between the intervals each corresponding to a value of the generated value. If we want to generate a random variable A with probabilities $P(A = a_j)$, j = 1, 2, ..., we calculate the probability F_i that $A \leq a_i$:

$$F_i = \sum_{j=1}^{i} P(A=a_j), \quad i = 1, 2, \ldots$$

and store them in a table augmented with the value $F_0 = 0$ (if the number of values is infinite we have to stop at a reasonable value where the probability of higher values is negligible or we need some expression for calculation of F_i). We now generate a random variable, X, uniformly distributed on [0,1] and for each generated value, x, we search in the table to determine the i for which $F_{i-1} \leq x < F_i$ since x then contributes to the probability that the A variable has value a_i. The procedure is shown in the figure below for a simple example.

If A is a variable assuming a continuous set of values, the same method may be applied after the continuous set of values has been divided into (small) intervals, i.e. “quantized”, each having a probability as above. In some cases, the quantization and the search could be avoided, by noting that the F_i above is just the distribution function F(a) and the search corresponds to finding the a for which x

= F(a). In some cases the function is invertible, and thus the value of the random variable A is generated just as a = $F^{o-1}(x)$.

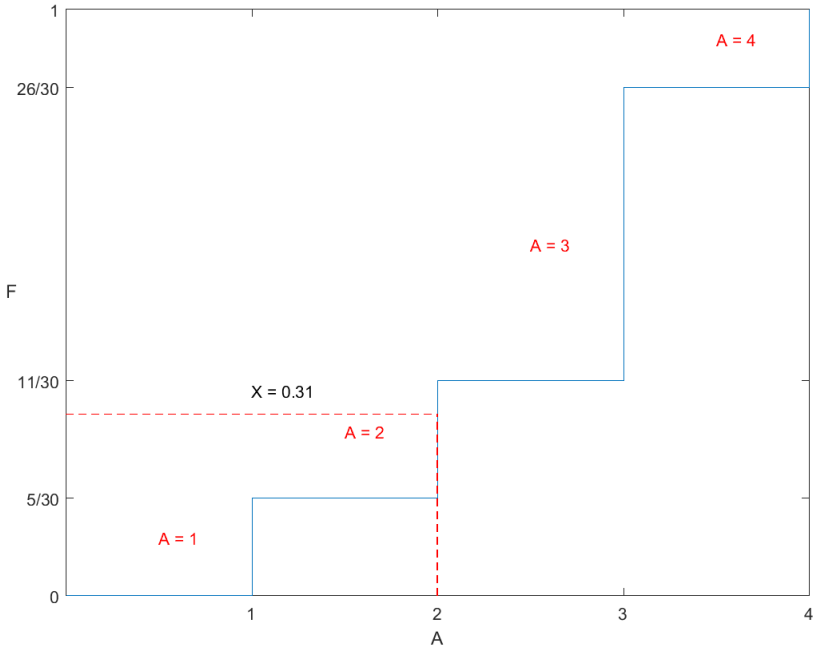

Figure 3.2
Generation of variable A with sample space {1, 2, 3, 4} having probabilities [1/6 1/5 1/2 4/30].

An example of this is shown for the Laplacian-density:

$$f(a) = \frac{1}{\sqrt{2}\sigma} e^{-\frac{\sqrt{2}|a|}{\sigma}} \qquad F(a) = \begin{cases} \frac{1}{2} e^{\frac{\sqrt{2}a}{\sigma}} & a < 0 \\ 1 - \frac{1}{2} e^{-\frac{\sqrt{2}a}{\sigma}} & a \geq 0 \end{cases}$$

in the MATLAB program:

```
sigma=1;
x=rand(1,10000);
a=(x<0.5)*sigma.*log(2*x)/sqrt(2)...
 -(x>=0.5)*sigma.*log(2*(1-x))/sqrt(2);
stem([-10:0.1:10],hist(a,[-10:0.1:10]))
```

The last program line displays a histogram which looks like the density function with a resolution interval of 0.1.

Unfortunately, the inverse of the distribution function for a Gaussian variable is not directly expressible in basic functions and the above search method is often too slow, so a trick has to be used. A Gaussian distribution with σ = 1, Y, may be

obtained by the so-called Box-Muller method from two variables, X_1 and X_2, with uniform density on [0,1] as

$$y_1 = \sqrt{-2\ln(x_1)}\cos(2\pi x_2)$$

$$y_2 = \sqrt{-2\ln(x_1)}\sin(2\pi x_2)$$

We omit the derivation of the density function (it involves the Rayleigh distribution just introduced above) and the proof that the two Gaussian variables Y_1 and Y_2 are independent. In MATLAB there is a direct function: `randn(m,n)` produces an m by n matrix of Gaussian variables with zero mean and variance 1. Other variances may be obtained by scaling with σ as many examples in this course will show.

3.2 Stochastic processes

We shall use stochastic processes as models of random signals and noise. Since most signal processing takes place in discrete time, we shall be mostly interested in this type of processes. There are several advantages to a discrete time approach, such processes are easily simulated, and the mathematics are much simpler. Many natural signals in telecommunication (to be described in Chapter 4), e.g. audio signals like speech and their analog electrical representations, are naturally thought of as continuous time functions. The complete description of a stochastic process in continuous time is more complicated, and we shall not attempt to give a precise definition of a continuous time process. For our purpose it is often sufficient to consider a sampled form of the signal with a sufficiently high sampling rate. In Section 3.6 we shall consider the sampling process, including the conversion from continuous time signals to discrete time signals. However, the main emphasis will be the conversion from a higher sampling rate to a lower rate. One reason for representing a 'continuous time' signal by a discrete time version with a suitable sampling rate is that this is the only possibility in simulations. In Section 3.5 we shall discuss a particular mixture of discrete and continuous time stochastic process. This mixture is artificially created in transmission systems and thus the description is more controllable.

A **discrete time stochastic process** is described as a collection of random variables,

$$\ldots, X_{n-1}, X_n, X_{n+1}, \ldots$$

We shall usually assume that the process is **stationary** (*stationær*), i.e. the distribution of X_n is independent of n. We can then, at least in principle, give a complete description of the process by specifying, for any N, the simultaneous distribution of $X_1, X_2, .. , X_N$. However, such an approach is usually very complex and provides little insight into the properties of the signal.

As a more useful approach to the description of random processes we define the **auto-correlation function** by

$$R_X(k) = E[X_{n+k} X_n] \tag{3.8}$$

For **stationary** sequences, this value is independent of n (as we have already assumed by writing $R_X(k)$), and it is then an **even** function of k. For k = 0 we get $R_X(0) = \sigma_x^2$ (assuming $E[X_n] = 0$), and this is always the largest value of $|R_X(k)|$. In the cases of interest here, we assume that the mean value $E[X_n] = 0$, which results in vanishing $R_X(k)$ for large k. The autocorrelation function has the advantage of being defined in a straightforward way, and that often makes it a useful starting point for the analysis. Example 3.2 demonstrates a calculation of the autocorrelation function for a simple process.

On the other hand the autocorrelation function does not provide much intuitive insight. In this respect we get a more interesting function by taking the Fourier transform of the autocorrelation function. We define the **power spectrum** (*effektspektret*) of the process by

$$S_X(\omega) = \sum_{k=-\infty}^{\infty} T\, R_X(k)\, e^{-j\omega kT} \tag{3.9}$$

and from (2.2)

$$R_X(k) = \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} S_X(\omega) e^{j\omega kT} d\omega \tag{3.10}$$

$S_X(\omega)$ is a **non-negative** function and it is a density, i.e. it gives the power per Hz. It will be clear later that the term power spectrum is justified. At this point it is simply a definition. However, we note already now that for k = 0 we get an expression indicating that the integral of $S_X(\omega)$ equals the total power, $R_X(0)$. In addition, the terminology is motivated by the relation between autocorrelation functions and power spectra of periodic signals, cf.

Section 2.4. Equation (3.10) is mostly of theoretical interest since normally calculation of the autocorrelation function from the power spectrum is done by identifying terms in (3.9) as we have seen before. In these notes we prefer to write the sampling time, T, between two signal values explicitly. One reason for doing this is that we get the proper physical dimension (power/Hz), but another important reason is that this time interval may change between different stages of signal processing. Note that frequencies in some books may be in Hz, while in these notes they are measured in rad/sec. There is no point in trying to coordinate this, since a change between these units will always occur frequently. However it should be noted that the definitions of the power spectrum are chosen in such a way that it is the same in the two cases. Example 3.3 shows an experimental determination of the power spectrum based upon the energy spectrum of blocks of samples of the process.

The power spectrum $S_X(\omega)$ may be constant (at least for all frequencies of interest) and in this case the signal is said to have a **white spectrum** (*hvidt spektrum*). An example of such signal is

$$\begin{aligned} &X_n \text{ independent, zero mean} \Rightarrow \\ &R_X(k) = E[X_{n+k}X_n] = E[X_{n+k}]E[X_n] = 0 \cdot 0 \text{ for all } k \neq 0 \\ &S_X(\omega) = \sum_{k=-\infty}^{\infty} T R_X(k) e^{-j\omega kT} = T R_X(0) = T\sigma_X^2 \text{ , a constant} \end{aligned} \tag{3.11}$$

Example 3.2 Autocorrelation and power spectrum for a simple process

The stochastic process X outputs a sequence of variables with values +1 and −1 in such a way that they appear in blocks of two symbols and only two such blocks are possible: [+1,−1] and [−1,+1]. These blocks are independent of each other and the probability of the first version is p.

Calculation of an autocorrelation function may be done in different ways, but basicly the line of thought is this: We pick a position *n* at random and try to evaluate $E[X_{n+k}X_n]$ for different *k*.

X_n is +1 or −1 with probability ½ independent of p which gives

$$R_X(0) = E[X_n^2] = \frac{1}{2}(+1)^2 + \frac{1}{2}(-1)^2 = 1$$

The randomly picked position n may be the first one in a block or the last one. Thus 4 different situations for the pair X_nX_{n+1} are found (block boundaries shown as []):

$$[+1,-1] \quad +1],[\pm 1 \quad [-1,+1] \quad -1],[\pm 1$$

and these situations will be picked with the probabilities p/2, (1−p)/2, (1−p)/2, and p/2, respectively. Thus

$$\begin{aligned} R_X(\pm 1) &= E[X_{n+1}X_n] = E[X_nX_{n+1}] \\ &= \frac{p}{2}(+1)(-1) + \frac{1-p}{2}(+1)[p(+1)+(1-p)(-1)] \\ &\quad + \frac{1-p}{2}(-1)(+1) + \frac{p}{2}(-1)[p(+1)+(1-p)(-1)] \\ &= -\frac{1}{2} - \frac{1}{2}(2p-1)^2 \end{aligned}$$

By the nature of the process there is no dependence between the symbol at position n and symbols at n+2 and further away. Thus

$$R_X(k) = E[X_{n+k}X_n] = E[X_{n+k}]E[X_n] \quad \text{for } k \geq 2$$

For $p \neq ½$ the process is not stationary since the distribution of X_n is dependent on n being first or second in a block. As we saw above, $R_X(1)$ is independent of n anyway, and this is also the case for $R_X(k)$ for $k \geq 2$ since $E[X_{2m}] = 2p-1$ and $E[X_{2m+1}] = -(2p-1)$ give $R_X(2m) = (2p-1)^2$ and $R_X(2m+1) = -(2p-1)^2$, $m \geq 1$. For $p = ½$, we have a stationary process with zero mean, $R_X(k) = 0$ for $k \geq 2$, and using (3.9), we get the power spectrum:

$$S_X(\omega) = T\left(-\frac{1}{2}e^{-j\omega\cdot(-1)\cdot T} + 1\cdot e^{-j\omega\cdot 0\cdot T} - \frac{1}{2}e^{-j\omega\cdot 1\cdot T}\right) = T(1 - \cos\omega T)$$

We shall return to this stochastic process later in Section 5.8.

Example 3.3 Experimental calculation of power spectrum

We shall determine the power spectrum for the stochastic process X from Example 3.2 by generating a sample function (of length N) in MATLAB and then calculate the energy spectrum for the sample as a deterministic signal, i.e. (2.18).

Since this gives the distribution of the energy, one has to divide by the length in time for the sample, i.e. NT, in order to determine the distribution of power, the power spectrum. It turns out that one sample is not enough to approach the theoretical spectrum calculated in Example 3.2, so the sample function has to be generated a number of times, `Nexp`. Here we use 100 sample functions, but the reader should try with fewer experiments, in order to see how the theoretical spectrum is approximated. We have left out the figure, 100 experiments give a good approximation.

```
T=1;
N=100;                % Block size gives frequency resolution
p=1/2;                % Probability of +1-1 block
Nexp=100;             % # of experiments
S_X=zeros(1,N);
for n=1:Nexp
    a=rand(1,N/2);
    a=sign(p-a);      % a becomes +1 with probability p
    a(2,:)=-a(1,:);  % Formation of +1-1 or -1+1 in columns of a
    x=reshape(a,1,N);% and concatenation of these blocks
    X=T*fft(x);
    S_X=S_X+abs(X).^2/(N*T);    % Contribution to average
end;
oindex=0:N-1;
omega=oindex*2*pi/(N*T);
stem(omega,S_X/Nexp);           % Experimental power spectrum
hold on
plot(omega,T*(1-cos(omega*T)),'r'); % Ex. 3.2 theory
hold off
```

It follows from (3.9) that the power spectrum and the autocorrelation function contain the same information about the stochastic process. This is far from being a complete description, however. The distribution functions (or density functions) of two signals may differ even though they have the same autocorrelation function. Such differences would also show up as different values of higher order statistics (moments)

$$E[X_n^3],\ E[X_n^4],\ \ldots$$

and different values of

$$E[X_n X_{n+k} X_{n+\ell}],\ E[X_n X_{n+k} X_{n+\ell} X_{n+m}],\ \ldots$$

It is of course also interesting to discuss the properties of more than one stochastic process. We define the **crosscorrelation** (*krydskorrelationen*) between two stationary signals as

$$R_{XY}(k) = E[X_{n+k} Y_n] \tag{3.12}$$

Taking the Fourier transform of $R_{XY}(k)$ we obtain the **cross power spectrum** (*kryds-effektspektret*).

Two stochastic processes are **independent** if for all n and m, the random variables X_n and Y_m are independent. Similarly, they are **uncorrelated**, if this property holds for all n and m. If Y_n is the sum of several stochastic processes $X_n^{(i)}$:

$$Y_n = \sum_i c_i X_n^{(i)}$$

then

$$R_Y(k) = E[Y_{n+k} Y_n] = E[\sum_i c_i X_{n+k}^{(i)} \cdot \sum_j c_j X_n^{(j)}] = \sum_i \sum_j c_i c_j E[X_{n+k}^{(i)} X_n^{(j)}]$$

and if the processes $X^{(i)}$ are zero mean and **uncorrelated**, we have $E[X_{n+k}^{(i)} X_n^{(j)}] = 0$ for $i \neq j$ and we get

$$R_Y(k) = E[Y_{n+k} Y_n] = \sum_i c_i^2 R_{X^{(i)}}(k) \quad \text{and} \quad S_Y(\omega) = \sum_i c_i^2 S_{X^{(i)}}(\omega)$$

3.3 Linear operations on stochastic signals

If we consider several zero mean **random variables** as a vector **X** with correlation matrix $\mathbf{R}_X = [R_{rc}] = [E[X_r \cdot X_c]]$ (cf. (3.6)), and **y** is obtained by the linear mapping $\mathbf{y} = \mathbf{A}\mathbf{x}$, we may find the correlation matrix of **Y** as[1]

$$\mathbf{R}_Y = \mathbf{A}\mathbf{R}_X\mathbf{A}' \tag{3.13}$$

In control theory and multiple-input signal processing such linear operations are commonly used.

If **X** is a **stochastic process** and used as input to a linear filter (cf. Section 2.3), the output is described by a linear difference equation, (2.13)

$$y_n = \sum_{j=0} T h_j x_{n-j} \tag{3.14}$$

Remember that the dimension of h_j here is time^{-1} such that x_n and y_n have the same dimension. If h_j has another dimension, the use of T should be adjusted accordingly. The effect

---

[1] The result may also be stated using the covariance matrix, but it equals $\mathbf{R}_X$ for zero mean variables.

of the linear filter may found by calculating the autocorrelation function of Y for a stationary X process:

$$\begin{aligned} R_Y(k) &= E[Y_{n+k}Y_n] = \sum_{s=0}^{\infty}\sum_{t=0}^{\infty} T^2 h_s h_t E[X_{n+k-s}X_{n-t}] \\ &= \sum_{s=0}^{\infty}\sum_{t=0}^{\infty} T^2 h_s h_t R_X((n+k-s)-(n-t)) \\ &= \sum_{r=-\infty}^{\infty} R_X(k-r)\sum_{t=0}^{\infty} T^2 h_t h_{r+t} \end{aligned} \tag{3.15}$$

Thus the autocorrelation function is convolved with a function that is obtained by correlating the impulse response with itself. Taking the Fourier transform we obtain the important relation

$$S_Y(\omega) = S_X(\omega)\,|H(\omega)|^2 \tag{3.16}$$

This relation should be compared to (2.29) for deterministic signals.

Usually (3.16) is the simplest to calculate. However, in some problems the direct calculation using the time domain relation (3.15) may be convenient. Note that these results do not depend on the distribution function of the random signal. An important property of Gaussian processes is that the output process from a linear filter is Gaussian if the input process is. For discrete and other distribution functions, the distribution at the output will usually be different, and it may not be easy to calculate it.

The effect of the linear filter as expressed in (3.16) can be used to obtain an interpretation of the power spectrum: Assume that a filter is chosen such that the system function is close to 0 except for a narrow frequency band of length $2\pi\delta$ around ω' where $H(\omega') = 1$. It now follows from (3.16) that the output power spectrum equals the input spectrum $S_X(\omega')$ in this interval and is zero otherwise, and from (3.10) we get that $R_Y(0)$, the power of the output, equals $\delta S_X(\omega')$. This is exactly the interpretation we expect for the power spectral density.

We may calculate the crosscorrelation between input and output for the linear system in (3.14) as

$$\begin{aligned} R_{XY}(k) &= E[X_{n+k}\,Y_n] \\ &= E[X_{n+k}\sum_{j=0}^{\infty} T h_j X_{n-j}] = \sum_{j=0}^{\infty} T h_j E[X_{n+k}X_{n-j}] = \sum_{j=0}^{\infty} T h_j R_X(k+j) \end{aligned}$$

and the cross power spectrum becomes

$$S_{XY}(\omega) = S_X(\omega)\, H^*(\omega) \tag{3.17}$$

which should be compared to (2.30). In a way similar to the approach discussed in Section 2.8, (3.17) may serve as a starting point for measuring an unknown transfer function by applying a stochastic process with a constant power spectrum to the input.

Example 3.4 Spectrum of output from linear system

A sequence of independent binary variables (with probability of 1 and −1 both ½) are used as input to a filter with difference equation

$$y_n = \frac{1}{2}x_n - \frac{1}{2}x_{n-1}$$

The filter has $\mathbf{h} = (½, -½)/T$ and $H(\omega) = (1 - e^{-j\omega T})/2$.

The distribution on the output is found to be (¼, ½, ¼) for (−1, 0, 1). In this case we find the output distribution by considering all four combinations of input values that combine to form an output value. The autocorrelation function and power spectrum do not depend on the input distribution, but follow from the white spectrum (T) of the input and the transfer function of the filter. We find

$$S_Y(\omega) = T\left|\frac{1}{2}(1 - e^{-j\omega T})\right|^2 = T\sin^2\frac{\omega T}{2} = \frac{T}{2}(1 - \cos\omega T)$$

It is readily seen that in this sequence, 1 and −1 have to alternate, since an output value of 1 can only be produced by a −1, 1 input, and the following outputs are 0 as long as the input is 1 and −1 when the first −1 input occurs. We shall discuss this sequence again shortly as the so-called AMI (Alternate Mark Inversion) code.

The difference equation for a general first order linear system is (cf. (2.15) with $p_0 = 1$ and Example 2.8)

$$y_n = -qy_{n-1} + x_n + px_{n-1} \tag{3.18}$$

with transfer function

$$H(\omega) = \frac{1+pe^{-j\omega T}}{1+qe^{-j\omega T}}$$

Remember from Example 2.8 that $|q| < 1$ is required for the system to be stable. If the system is driven by a signal with a white spectrum and variance 1 the response has power spectrum

$$\begin{aligned} S_Y(\omega) &= T\cdot|H(\omega)|^2 = TH(\omega)H^*(\omega) \\ &= T\frac{(1+pe^{-j\omega T})(1+pe^{j\omega T})}{(1+qe^{-j\omega T})(1+qe^{j\omega T})} = T\frac{1+p^2+2p\cos\omega T}{1+q^2+2q\cos\omega T} \end{aligned} \qquad (3.19)$$

The autocorrelation function is found by partial fraction decomposition and reformulation as an infinite power series (requires also $|q| < 1$)

$$|H(\omega)|^2 = \frac{1+p^2-2pq}{1-q^2} + \frac{(p-q)(1-pq)}{1-q^2}\left(\frac{e^{-j\omega T}}{1+qe^{-j\omega T}} + \frac{e^{j\omega T}}{1+qe^{j\omega T}}\right) =$$

$$\frac{1+p^2-2pq}{1-q^2} + \frac{(p-q)(1-pq)}{1-q^2}\left(e^{-j\omega T}\sum_{k=0}^{\infty}(-qe^{-j\omega T})^k + e^{j\omega T}\sum_{k=0}^{\infty}(-qe^{j\omega T})^k\right)$$

Since $S_Y(\omega)$ is found by transformation of $R_Y(k)$ (Eq. (3.9)), we can now find $R_Y(k)$ by identifying terms

$$R_Y(0) = \frac{1+p^2-2pq}{1-q^2} \text{ and } R_Y(k) = \frac{(p-q)(1-pq)}{1-q^2}(-q)^{|k|-1} \text{ for } |k| > 0 \qquad (3.20)$$

The condition $|q| < 1$ assures that $R_Y(k)$ decays with k. The autocorrelations could also have been calculated from the infinite impulse response in Example 2.8 using (3.15).

Similarly we might express an N'th order rational spectrum by the numerical square of a transfer function like (2.16). Again the autocorrelation function might be obtained by partial fraction expansion. We shall omit the details, but note that using the difference equation (2.15),

$$y_n = -\sum_{j=1}^{N} q_j y_{n-j} + \sum_{j=0}^{N} p_j x_{n-j}$$

multiplying both sides by y_{n-N-k}, an earlier output independent on any of the current inputs, and taking expected values, we get

$$R_Y(N+k) = -\sum_{j=1}^{N} R_Y(N+k-j)\,q_j \tag{3.21}$$

Thus for k large enough, the autocorrelation function satisfies a recursion given by the denominator of H (cf. (2.16)). As a simple example we have the first order (N = 1) system in (3.20) where it is easily seen that $R_Y(k+1) = -qR_Y(k)$ for $k > 0$.

Example 3.5 Second order linear system

Let a linear system be defined by the difference equation

$$y_k = y_{k-1} - \frac{1}{2}y_{k-2} + x_k - x_{k-2}$$

with system function (from (2.16)):

$$H(\omega) = \frac{1 - e^{-j2\omega T}}{1 - e^{-j\omega T} + \frac{1}{2}e^{-j2\omega T}}$$

If the system is driven by a signal with a white spectrum and variance 1, $S_X(\omega) = T$, the power spectrum of the output becomes

$$S_Y(\omega) = T\cdot|H(\omega)|^2 = T\,H(\omega)H^*(\omega) = T\,\frac{2-2\cos 2\omega T}{\frac{9}{4} - 3\cos\omega T + \cos 2\omega T}$$

To find the autocorrelation function we substitute $z = e^{-j\omega T}$ to make the notation simpler. Noticing that complex conjugation takes z to z^{-1}, we have

$$\sum_{k=-\infty}^{\infty} TR_Y(k)z^k = S_Y(z) = T\,H(z)H(z^{-1}) = T\,\frac{1-z^2}{1-z+\frac{1}{2}z^2}\cdot\frac{1-z^{-2}}{1-z^{-1}+\frac{1}{2}z^{-2}} =$$

$$T\,\frac{1+1-(z^2+z^{-2})}{1+1+\frac{1}{4}-(z+\frac{1}{2}z+z^{-1}+\frac{1}{2}z^{-1})+\frac{1}{2}(z^2+z^{-2})} = T\,\frac{2-(z^2+z^{-2})}{\frac{9}{4}-\frac{3}{2}(z+z^{-1})+\frac{1}{2}(z^2+z^{-2})}$$

The product may be decomposed into two partial fractions

$$\sum_{k=-\infty}^{\infty} R_Y(k)z^k = R_Y(0) + \frac{az+bz^2}{1-z+\frac{1}{2}z^2} + \frac{az^{-1}+bz^{-2}}{1-z^{-1}+\frac{1}{2}z^{-2}}$$

We may find the unknowns, $R_Y(0)$, a, and b, by inserting certain values into $S_Y(z)$, e.g. $z = \pm 1$ and another value at the unit circle, $z = j$. The first two values give the first two rows in the equation below, but for the last value the calculations are tedious. An easier way to obtain the last equation is to consider the constant term (= 2) in the numerator and determine how this is calculated from the unknowns. We get

$$\begin{bmatrix} 1 & 4 & 4 \\ 1 & -\frac{4}{5} & \frac{4}{5} \\ \frac{9}{4} & -2 & 1 \end{bmatrix} \begin{bmatrix} R_Y(0) \\ a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

We let MATLAB take over to actually determine the autocorrelations

```
M=[ 1   4    4;...
    1 -4/5  4/5;...
   9/4 -2    1];
unknown=M^-1*[0 0 2]'
unknown =
    4.0000
    2.0000
   -3.0000
R0=unknown(1);
a=unknown(2);
b=unknown(3);
Rplus=deconv([a,b,zeros(1,10)],[1,-1,.5])
Rplus =
    2.0000   -1.0000   -2.0000   -1.5000   -0.5000
    0.2500    0.5000    0.3750    0.1250   -0.0625
```

The last operation is found by noticing that $R_Y(k)$ for positive k are found from the first of the partial fractions, and multiplying $\sum R_Y(k)z^k$ with the denominator gives the numerator of that fraction. Another way to express this is that there exists a recursion among the autocorrelation values as given by (3.21):

$$R_Y(2+k) = -\left(R_Y(2+k-1)(-1) + R_Y(2+k-2)\frac{1}{2}\right)$$

You may check that this is fulfilled for the values of **Rplus**.

3.4 Markov sources and their spectra

The most important method for describing discrete sequences is Markov chains, or more generally, functions of Markov chains, **Markov sources** (*Markovkilder*).

A finite state **Markov chain** (*Markovkæde*) is based upon a finite set of n **states**, $\{s_1, ..., s_n\}$. The stochastic process starts in one of these states and moves successively from one state to another. Each move is called a **step** and later we assume that the time spent in each state is T. The basic property is now that the probability of moving to the next state s(k) depends only on the current state s(k−1) and not all steps before reaching s(k−1), i.e.

$$\mathbf{P(s(k) \mid s(k-1),\ s(k-2),\ \ldots) = P(s(k) \mid s(k-1))} \tag{3.22}$$

The conditional probabilities in (3.22) may be expressed as an n×n matrix, $\mathbf{Q} = [q_{ij}]$ of **transition probabilities** (*overgangssandsynligheder*) where q_{ij} (row i, column j) is the probability of making a transition from state i to state j (note the sequence). **Q** is a so-called stochastic matrix, i.e. all elements $q_{ij} \geq 0$, and the rows sum to 1.

If the probability distribution on the states s_i at time k is given by the vector

$$\mathbf{r(k) = (P(s(k)=s_1),\ P(s(k)=s_2),\ \ldots,\ P(s(k)=s_n))}$$

then the distribution after the next transition is

$$\mathbf{r(k+1) = r(k)\,Q}$$

which follows from the definition of **Q** and the property (3.22). The probabilities of m-step state transitions may then be found as

$$\mathbf{Q(m) = Q^m}$$

We shall be interested only in **regular Markov chains**, which are defined by the property that all elements of $\mathbf{Q}^m$ are positive (i.e. > 0) for sufficiently large m. This means that all states may be reached from all other states after m transitions (or before). It follows from this property that there is a unique stationary probability distribution **p**, which may be determined as the left eigenvector with $\Sigma p_i = 1$ for the eigenvalue 1:

$$\mathbf{p = (p_1,\ ...,\ p_n) = p\,Q} \tag{3.23}$$

This eigenvector may also be found as rows in

$$\lim_{m \to \infty} \mathbf{Q}^m = \begin{bmatrix} \mathbf{p} \\ \mathbf{p} \\ \vdots \\ \mathbf{p} \end{bmatrix}$$

since the transition probabilities must converge to the stationary probabilities after many steps.

There exists a larger class of Markov chains (**ergodic** chains) where the requirement is that all states may be reached from all other states, but not necessarily after some common number of transitions as for the regular chains. As an example, one could think of a chain with a periodic behavior where one subset of the states appears at some time and another subset at the next step and so on until the first subset appears again. Such a chain is not regular since there may be transitions which are impossible no matter how many steps we have done. Ergodic chains also have a stationary distribution calculated as in (3.23), but it is more difficult to interpret, e.g. $\mathbf{Q}^m$ does not converge as above. For further details on Markov chains, the reader may consult [3.2] or another standard text on probability theory.

When Markov chains are applied as descriptions of signals, some function to describe the output is needed. Thus the state sequence is a Markov chain, but the signal is a function of the state or a function of the transition. An example is shown graphically in Figure 3.3. In general we do not encourage the use of graphical descriptions since there are many chances of misinterpretation and errors. It is always better to use the **transition matrix Q**, and to describe the output we introduce the **output matrix X** = $[x_{ij}]$ where x_{ij} is the real output symbol produced at the transition from state i to state j. (If the output is only a function of the state j, all symbols in column j are the same). If you are familiar with design of digital logic, you may know the structure as a **Finite State Machine**, and in the form described here as a **Mealy-type** machine [3.3]. The only difference here is that now probabilities are assigned to the transitions instead of the input signals in digital logic, but sometimes the probabilities just describe a property of an input signal, e.g. when we are using Markov chains to describe line coding (Section 5.8) as in the following Example 3.7. The analysis of Markov sources is greatly simplified if the state sequence can be identified from

observations of the signal. (Sources which do not have this property are known as Hidden Markov Models, and such models are useful, but difficult).

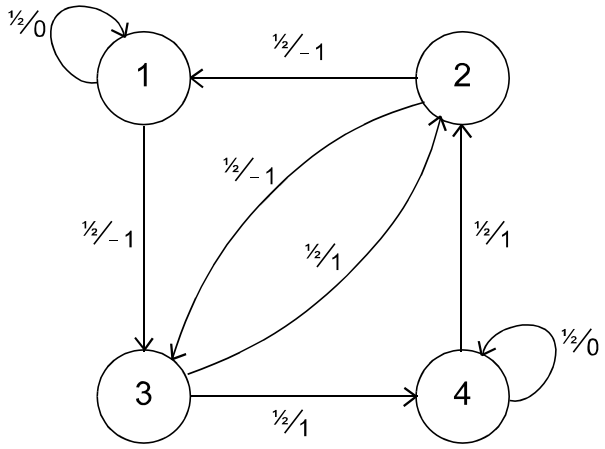

Figure 3.3
Example of Markov source. The transition probabilities are shown before the slash at each transition; the corresponding output is after the slash.

We shall derive the autocorrelation function and power spectrum for a signal generated by a Markov source derived from **regular** Markov chains. It is convenient to introduce the average of the symbols following each state i as the vector $\mathbf{x}^+$ with elements

$$x_i^+ = \sum_{j=1}^{n} q_{ij} x_{ij}$$

Similarly we may calculate the average value of the symbol produced before entering state i, x_i^-. An explicit formula is found below, however in the examples it is usually a simple calculation:

$$x_i^- = \frac{\sum_j p_j q_{ji} x_{ji}}{p_i}$$

We may now start calculating the autocorrelation function. First we find directly from the definition of the power

$$R_X(0) = E[X_m^2] = \sum_{i=1}^{n} p_i \sum_{j=1}^{n} q_{ij} x_{ij}^2 \tag{3.24}$$

Then from the definitions of $\mathbf{x}^+$ and $\mathbf{x}^-$ we immediately get

$$R_X(1) = E[X_{m+1} X_m] = \sum_{i=1}^{n} p_i x_i^- x_i^+$$

Since the probability of making a transition from state i to state j in k > 0 steps is the ij'th element of $\mathbf{Q}^k$, we can now express the autocorrelation function as

$$R_X(k) = E[X_{m+k} X_m] = (p_1 x_1^-, ..., p_n x_n^-)\, \mathbf{Q}^{k-1}\, (x_1^+, ..., x_n^+)' \quad \text{for } k > 0 \qquad (3.25)$$

Here the vectors $\mathbf{p}\mathbf{x}^- = (p_j x_j^-)$ and $\mathbf{x}^+$ are independent of k. Thus $R_X(k)$ depends on only the powers of $\mathbf{Q}$.

The calculation outlined above may seem quite complicated. We shall use MATLAB to do some numerical examples, e.g in the following example we will perform these calculations on the source given in Figure 3.3 and present a rather general MATLAB program for the numerical calculations. If the calculations are to be done by hand, it is a good idea to look for symmetries which may be of help.

Example 3.6 A MATLAB program for a Markov source

From Figure 3.3 we obtain the followings matrices

$$\mathbf{Q} = \begin{bmatrix} .5 & 0 & .5 & 0 \\ .5 & 0 & .5 & 0 \\ 0 & .5 & 0 & .5 \\ 0 & .5 & 0 & .5 \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 0 & -1 \\ -1 & -1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Outputs for not existing transitions are not shown, but in a MATLAB program they may be set to any value. The stationary distribution is found as the eigenvector $\mathbf{p}$ = (¼,¼,¼,¼). The remaining calculations are rather simple, but to illustrate the use of MATLAB in such cases we give the following program. Note that the MATLAB function `eig` finds the column eigenvector to be multiplied to the right of the matrix, but in (3.23) we need a row vector to the left of the matrix. Since $(\mathbf{Q}'\mathbf{p}')' = (\mathbf{p}')'(\mathbf{Q}')' = \mathbf{p}\mathbf{Q}$ the $\mathbf{p}$ may be found from the transposed matrix $\mathbf{Q}'$.

```
Q=[.5  0 .5  0;...
   .5  0 .5  0;...
    0 .5  0 .5;...
    0 .5  0 .5];
[V,D]=eig(Q')
V =
 0.5000 + 0.0000i  0.5000 - 0.0000i  0.5000           0.6964
-0.5000 - 0.0000i -0.5000 + 0.0000i  0.5000          -0.6964
 0.5000            0.5000            0.5000          -0.1228
```

```
-0.5000 - 0.0000i -0.5000 + 0.0000i  0.5000            0.1228
D =
 0.0000 + 0.0000i       0                  0                  0
      0            0.0000 - 0.0000i       0                  0
      0                 0            1.0000                  0
      0                 0                 0                  0

% The eigenvalues are found as the diagonal of D, and it is
% seen that eigenvalue 1 corresponds to column 3 of V
% The stationary distribution then becomes
p=V(:,3)'/sum(V(:,3))
```

The stationary distribution is **p** = (.2500, .2500, .2500, .2500) as we saw above.

```
% The output matrix is X. Some of the 0-elements are irrelevant
% since the corresponding transition is not found
X=[ 0  0 -1  0;...
   -1  0 -1  0;...
    0  1  0  1;...
    0  1  0  0];
% (3.24)
R0=p*sum(Q.*X.^2,2)
R0 =
    0.7500
% x+, the average of the symbol after each state
% is found from the rows of Q
Xplus=sum(Q.*X,2)'
Xplus =
   -0.5000   -1.0000    1.0000    0.5000
% x-, the average of the symbol before each state
% is found from the columns of Q
Xminus=(p*(Q.*X))./p
Xminus =
   -0.5000    1.0000   -1.0000    0.5000
% (3.25)
for j=1:6
   Rj(j,:)=[j (p.*Xminus)*Q^(j-1)*Xplus'];
end
Rj
Rj =
    1   -0.3750
    2    0.0625
    3    0
    4    0
    5    0
    6    0
```

The vectors $\mathbf{x}^+$ and $\mathbf{x}^-$ are

$$\mathbf{x}^+ = (-0.5000, -1.0000, 1.0000, 0.5000)$$

$$\mathbf{x}^- = (-0.5000, 1.0000, -1.0000, 0.5000).$$

The autocorrelations become $R_X(0) = 0.75$, $R_X(1) = -0.375$, $R_X(2) = 0.0625$, and $R_X(k) = 0$ for $k > 2$. The last property may not be seen directly, but as we shall see in the following, recursions exist among the autocorrelation coefficients, such that a reasonable number of zeros, e.g. corresponding to the size of **Q**, leads to that the remaining autocorrelations also become zero.

Using (3.9) we get the power spectrum

$$S_X(\omega) = T\left(\frac{3}{4} - \frac{3}{4}\cos\omega T + \frac{1}{8}\cos 2\omega T\right)$$

We could also use MATLAB to calculate the power spectrum for a finite set of frequencies. Remember to place the autocorrelations correctly starting with $R_X(0)$ and then $R_X(k)$ for positive k followed by $R_X(k)$ for negative k. A 16 point power spectrum, **Sw**, could be calculated from

```
R=[R0 Rj(:,2)' zeros(1,16-1-2*6) fliplr(Rj(:,2)')];
T=1;
Sw=T*fft(R);
```

In many cases, the (3.25) provides autocorrelation coefficients that never seem to disappear, and it may be difficult to find a closed expression for the result. To help with that, we may apply the result from matrix algebra that a matrix satisfies its own *characteristic equation*, $g(z) = \det(\mathbf{Q} - z\mathbf{E}) = 0$. If the coefficients of g(z) are g_0, ... g_n, we have:

$$\sum_{j=0}^{n} g_j \mathbf{Q}^j = \mathbf{0} \quad \Rightarrow$$

$$0 = \mathbf{p}\mathbf{x}^- \mathbf{Q}^{k-1} \sum_{j=0}^{n} g_j \mathbf{Q}^j \mathbf{x}^{+\prime} = \sum_{j=0}^{n} g_j (\mathbf{p}\mathbf{x}^- \mathbf{Q}^{k+j-1} \mathbf{x}^{+\prime}) = \sum_{j=0}^{n} g_j R_X(k+j) \quad \text{for } k \geq 1$$

Thus $R_X(k)$ satisfies a linear recursion, g(z), just as we found it for processes generated by linear filters, and we may do the summation in (3.9) explicitly to obtain the power spectrum if we know some initial values of $R_X(k)$.

The procedure outlined above may still seem quite complicated, but for a small number of states and some symmetry in the problem, the analysis is not so difficult. We have shown how to use MATLAB to do numerical calculations, and using MAPLE or other tools for symbolic computation, it is possible to find explicit formulas for all spectra of practical interest. In specific cases it may be possible to use shortcuts in this procedure, but we recommend that the calculation is always based on finding the autocorrelation function first. In [3.4] there is an outline of a procedure for finding the power spectra of sources with memory, and this method is based on a direct expression for the spectrum.

Interesting examples of discrete processes are discussed in the Section 5.8. At this point we give a few properties of an important code, the AMI code.

Example 3.7 AMI, Alternate Mark Inversion Code

The so-called 'Alternate Mark Inversion' code is an important technique for suppressing low frequency content in a digital signal (the name refers to the telegraph symbols 'mark' and 'space'). In this format, logical 0 is transmitted as a real 0, where as logical 1 is transmitted as 1 or −1, where the polarity is chosen as the opposite of the previous logical 1. We may describe this signal by a Markov source with 2 states, describing the polarity of the last nonzero symbol, i.e. +1 or −1. If the two logical values are assumed to have probability ½, we get the transition matrix

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

since a 0 leaves the source in the same state, and a logical 1 changes it because of the new symbol +1 or −1 that is emitted. The output is

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

The stationary distribution is found as the eigenvector $\mathbf{p} = (½,½)$. Calculating $\mathbf{Q}^2$, we find that it equals $\mathbf{Q}$, and thus all higher powers of the transition matrix are the same. The distribution of the output symbols (1, 0, −1) is (¼, ½, ¼) which gives $R_X(0) = ½$. The average output from the two states are $\mathbf{x}^+ = (-½,½)$, and before the states, $\mathbf{x}^- = (½,-½)$. From these values we get (3.25) $R_X(1) = -¼$. Other values of $R_X(k)$ are 0, since the last symbol of any sequence of length k + 1 has equal probability of being 1 or −1. This may also be seen in a more complex way by calculating (3.25) for k = 2 and remembering that $\mathbf{Q}^{k-1} = \mathbf{Q}$ for $k \geq 2$.

Using (3.9) we get the power spectrum

$$S_X(\omega) = \frac{T}{2}(1 - \cos\omega T)$$

This spectrum was found in a different way in Example 3.4.

A little more complex problem is found if the transition probabilities are not equal for 0 and ±1. Let us assume that we instead have the transition matrix

$$\mathbf{Q} = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$

p still becomes (½, ½), but $\mathbf{x}^+ = (-p, p)$ and $\mathbf{x}^- = (p, -p)$. The distribution of the output symbols (1, 0, −1) is (½p, 1−p, ½p) which gives $R_X(0) = p$. Inserting into (3.25), we get $R_X(1) = -p^2$ and

$$\begin{aligned} R_X(k) &= (\tfrac{1}{2}p, -\tfrac{1}{2}p)\,\mathbf{Q}^{k-1}\begin{pmatrix} -p \\ p \end{pmatrix} = (\tfrac{1}{2}p, -\tfrac{1}{2}p)\,\mathbf{Q}^{k-2}(1-2p)\begin{pmatrix} -p \\ p \end{pmatrix} \\ &= (\tfrac{1}{2}p, -\tfrac{1}{2}p)\,(1-2p)^{k-1}\begin{pmatrix} -p \\ p \end{pmatrix} = -p^2(1-2p)^{k-1} \end{aligned}$$

This is in agreement with the fact that the characteristic equation for **Q** is $z^2 + (2p-1)z + 1 - 2p = (z - 1)(z - (1-2p))$. The recursion derived from the first factor would be a constant $R_X(k)$ which is illegal, but the second factor gives $R_X(k+1) = (1-2p)R_X(k)$ which is just as the explicit calculation. Inserting into (3.9) we get

$$\begin{aligned} S_X(\omega) &= \sum_{k=-\infty}^{\infty} T\,R_X(k)\,e^{-j\omega kT} \\ &= TR_X(0) + T\sum_{k=1}^{\infty} R_X(k)\,e^{-j\omega kT} + T\sum_{k=1}^{\infty} R_X(k)\,e^{-j\omega(-k)T} \\ &= Tp - Tp^2 e^{-j\omega T}\sum_{m=0}^{\infty}((1-2p)e^{-j\omega T})^m - Tp^2 e^{j\omega T}\sum_{m=0}^{\infty}((1-2p)e^{j\omega T})^m \\ &= Tp - Tp^2\left(\frac{e^{-j\omega T}}{1-(1-2p)e^{-j\omega T}} + \frac{e^{j\omega T}}{1-(1-2p)e^{j\omega T}}\right) \\ &= T\frac{2p(1-p)}{1+(1-2p)^2-2(1-2p)\cos\omega T}(1-\cos\omega T) \end{aligned}$$

The following example presents an extension to the above described method where we were able to calculate the autocorrelation function for Markov sources emitting one symbol from each transition. The extension allows more symbols for each transition, i.e. $\mathbf{x}_{ij}$ is now a vector.

Example 3.8 Extension to vector output from the states

This example will extend the above treatment a little since we now allow vectors of symbols as output from each state. This will in many cases reduce the number of states significantly, but it introduces other problems, e.g. the simple expression (3.25) could not be used immediately.

We want a sequence of ±1 with the property that there are never more than two of the same symbols following each other. This number is known as runlength (cf. Section 5.8) so we want a source with maximum runlength of two. We also want the probability of the runs of two to be a little higher than that of isolated symbols. This is to create a more interesting spectrum. It is not difficult to specify an irregular Markov source with this property having 4 states and emitting one symbol in each transition. However, here we shall try a different specification with two states and two symbols output in each transition:

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{4} & \frac{3}{4} \\ \frac{3}{4} & \frac{1}{4} \end{bmatrix}$$

It is easily verified that the stationary probability is ½ for each state. The possible output sequences are (1 1), (−1−1) (−1+1) (+1−1) with output matrix

$$\mathbf{X} = \begin{bmatrix} -1+1 & -1-1 \\ +1+1 & +1-1 \end{bmatrix}$$

You may check that state 1 always has at least one +1 immediately before and the next symbols are one or two −1, and further, the probability of the runs of two is higher than that for the isolated symbols, so both requirements are met. Since all symbols have magnitude 1, $R_X(0) = 1$. Our approach to the calculation of the power spectrum will be to first calculate the autocorrelation function, and then manipulate infinite series of the power spectrum (3.9) to a rational expression. As discussed above, the autocorrelation function will eventually satisfy a linear recursion, which we will determine. Once the first values have been calculated directly, the rest can be determined from the recursion. The recursion is

determined by the characteristic polynomial det($\mathbf{Q} - z\mathbf{E}$) which is calculated to $g(z) = z^2 - \frac{1}{2}z - \frac{1}{2}$. (MATLAB provides the function `poly` for this). g(z) has two roots, the eigenvalues of the matrix. The root z = 1 gives a constant $R_X(k)$ which is illegal, but the other root $-\frac{1}{2}$ gives a usable recursion. Since the source produces two symbols in each state transition, this recursion is

$$R_X(k) = -\frac{1}{2}R_X(k-2), \quad k > 2$$

and we need some initial values, $R_X(1)$ and $R_X(2)$. Using similar arguments as in Example 3.2, we get $R_X(1) = -\frac{1}{4}$ and $R_X(2) = -\frac{1}{2}$. The arguments are slightly more easy to overview due to the state formulation. The resulting autocorrelations are

$$\begin{aligned}
&\text{Odd } k > 0\text{: } k = 2p+1 \text{ (p integer):}\\
&R_X(k) = R_X(2p+1) = (-\tfrac{1}{2})R_X(2(p-1)+1) = \ldots = (-\tfrac{1}{2})^p R_X(1)\\
&\text{Even } k > 0\text{: } k = 2p \text{ (p integer):}\\
&R_X(k) = R_X(2p) = (-\tfrac{1}{2})R_X(2(p-1)) = \ldots = (-\tfrac{1}{2})^{p-1} R_X(2)
\end{aligned}$$

We may now calculate the power spectrum from (3.9) remembering that $R_X(-k) = R_X(k)$. Note that some extra manipulations are necessary since the recursion is split into one for the even terms and another for the odd terms:

$$\begin{aligned}
&S_X(\omega)\\
&= T\left(R_X(0) + \sum_{k=-1}^{-\infty} R_X(k) e^{-j\omega kT} + \sum_{k=1}^{\infty} R_X(k) e^{-j\omega kT} \right)\\
&= T\left(R_X(0) + \sum_{k=1}^{\infty} R_X(k) e^{j\omega kT} + \sum_{k=1}^{\infty} R_X(k) e^{-j\omega kT} \right)\\
&= T\left(R_X(0) + R_X(1)\sum_{p=0}^{\infty} (-\tfrac{1}{2})^p e^{j\omega(2p+1)T} + R_X(1)\sum_{p=0}^{\infty} (-\tfrac{1}{2})^p e^{-j\omega(2p+1)T} \right)\\
&\quad + T\left(R_X(2)\sum_{p=1}^{\infty} (-\tfrac{1}{2})^{p-1} e^{j\omega 2pT} + R_X(2)\sum_{p=1}^{\infty} (-\tfrac{1}{2})^{p-1} e^{-j\omega 2pT} \right)\\
&= T\left(1 - \frac{1}{4}\left(\frac{e^{j\omega T}}{1-(-\frac{1}{2})e^{j2\omega T}} + \frac{e^{-j\omega T}}{1-(-\frac{1}{2})e^{-j2\omega T}} \right) \right.\\
&\qquad \left. - \frac{1}{2}\left(\frac{e^{j2\omega T}}{1-(-\frac{1}{2})e^{j2\omega T}} + \frac{e^{-j2\omega T}}{1-(-\frac{1}{2})e^{-j2\omega T}} \right) \right)\\
&= T\frac{3 - 3\cos\omega T}{5 + 4\cos 2\omega T}
\end{aligned}$$

Example 3.9 The Miller code

This example will demonstrate some of the problems involved in computing the spectra of Markov sources, but also discuss the tools that may be used in the computation. The **Miller code**, [3.4], was developed to shape the spectrum of binary signals in magnetic recording by suppressing both low frequencies (many consecutive 1s or −1s) or high frequencies (sequences of 1, −1, 1, −1). The output may be described as a pair of binary symbols as in Example 3.2. All 4 pairs (1 1), (1 −1), (−1 1), (−1 −1) occur, and the state is identified by the output pair (in that order) leading to the state. The transition matrix is

$$\mathbf{Q} = \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

and it is easily verified that the stationary probability is ¼ for each state. The reader should go through the 4 transitions from state 1 and 2 and observe how the number of sign changes is controlled. State 4 and 3 are mirror images of state 1 and 2. The output matrix may look as

$$\mathbf{X} = \begin{bmatrix} - & 1\,-1 & - & -1\,-1 \\ - & - & -1\;1 & -1\,-1 \\ 1\;1 & 1\,-1 & - & - \\ 1\;1 & - & -1\;1 & - \end{bmatrix}$$

where output pairs for non-existing transitions are shown as −. Since all symbols have magnitude 1, $R_X(0) = 1$. As in the previous example, we shall determine a linear recursion, which the autocorrelations should satisfy. The recursion is determined by the characteristic polynomial $\det(\mathbf{Q} - z\mathbf{E})$ which is calculated to $g(z) = z^4 - ½z^2 - ½z = z(z-1)(z^2 + z + ½)$. g(z) has 4 roots, the eigenvalues of the matrix. The root z = 0 does not give any recursion and the root z = 1 gives a constant $R_X(k)$ which is illegal. We are left with the last factor which gives a

usable recursion. Since the code produces two symbols in each state transition, this recursion is

$$R_X(k) = -R_X(k-2) - \frac{1}{2}R_X(k-4)$$

and we need several initial values. As discussed with the blocking in Example 3.2, two symbols at distance 2k are separated by k state transitions, but symbols at distance 2k+1 may be in pairs separated by k or k+1 transitions. If we start from state 1 (preceding block (1 1)) or 2 (preceding block (1 −1), the average value of the following block is (0,−1) and (−1,0), respectively, and a symmetric situation is found starting in state 4 and 3. Thus $R_X(2) = -½$, and now the recursion gives $R_X(4) = 0$, $R_x(6) = ¼$, etc. $R_X(1)$ gets a term 0 from symbols in the same block, and ¼ from symbols in consecutive blocks. Similarly $R_X(3)$ consists of a term −¼ from neighboring blocks and another −¼ from blocks at distance 2. Now $R_X(5)$ may be found from the recursion as 3/8, $R_X(7) = -1/8$, etc. Whereas this calculation of the initial values was fairly easy, the remaining manipulations of the rational expressions are rather tedious, and a program for symbolic computation is helpful. However, the `conv` operation in MATLAB is actually sufficient to get the numerator and denominator of the rational function, and it is not so easy to make symbolic programs like MAPLE (or the MAPLE symbolic package associated with MATLAB) to work with positive and negative powers of the indeterminate. We omit the details here, but the calculations follow the line from Example 3.8 and use Z-transform as in Example 3.5:

$$\sum_{k=-\infty}^{\infty} R_X(k)z^k = \frac{1+\frac{1}{4}z+\frac{1}{2}z^2-\frac{1}{4}z^3}{1+z^2+\frac{1}{2}z^4} + \frac{1+\frac{1}{4}z^{-1}+\frac{1}{2}z^{-2}-\frac{1}{4}z^{-3}}{1+z^{-2}+\frac{1}{2}z^{-4}} - 1$$

which gives the power spectrum

$$S_X(\omega) = T\,\frac{3+\cos\omega T+2\cos 2\omega T-\cos 3\omega T}{9+12\cos 2\omega T+4\cos 4\omega T}$$

We may compare a plot (64 points) of the computed power spectrum to an fft of the autocorrelation function (64 points):

```
T=1;
omega=[0:63]*2*pi/64;
NU=3+cos(omega*T)+2*cos(2*omega*T)-cos(3*omega*T);
NOM=9+12*cos(2*omega*T)+4*cos(4*omega*T);
SX=T*NU./NOM;
```

```
plot(omega,SX);
hold on
rn=[1 1/4 1/2 -1/4 0 0 0];
recurs=[1 0 1 0 1/2];
rplus=deconv([rn zeros(1,29)],recurs)
rplus =
    1.0000    0.2500   -0.5000   -0.5000         0
    0.3750    0.2500   -0.1250   -0.2500   -0.0625
    0.1250    0.1250         0   -0.0938   -0.0625
    0.0313    0.0625    0.0156   -0.0313   -0.0313
         0    0.0234    0.0156   -0.0078   -0.0156
   -0.0039    0.0078    0.0078         0   -0.0059
   -0.0039    0.0020
rminus=fliplr(rplus);
rk=[0 rminus(1:31) rplus];
SX_R=T*fft(rk);
stem(omega,abs(SX_R))
```

The result is shown in Figure 3.4.

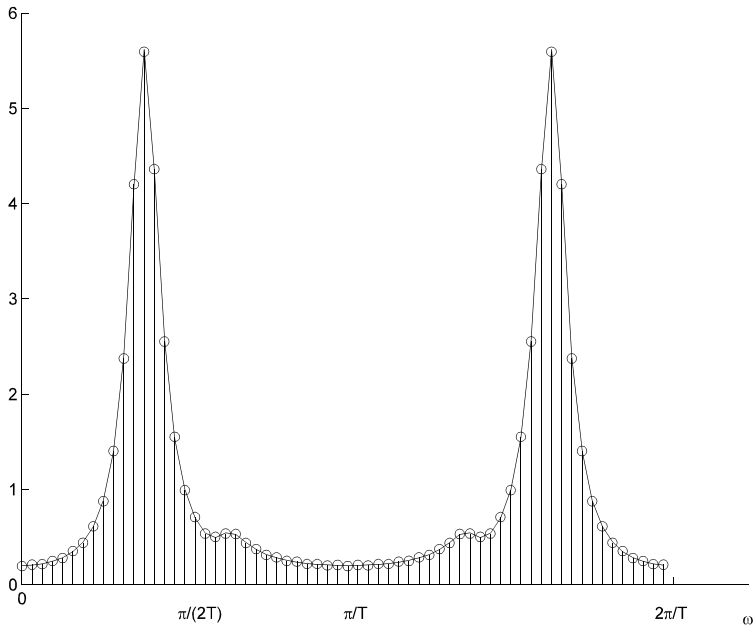

Figure 3.4
Spectrum for Miller code.

3.5 Mixed discrete and continuous time stochastic signals

As we stated in Section 3.2 the complete description of a stochastic process in continuous time is rather complicated, but in digital communication systems a particular mixture of a discrete and continuous time process is found, and we shall describe this mixture in the present section. Among the artificially created communication signals, an important example is the so-called **PAM signals** which are found in digital communication systems. A PAM signal is a sequence of **pulses** (*pulser*) of known shape (*pulsform*)

$$v(t) = \sum_{k=-\infty}^{\infty} a_k \, g(t - kT) \tag{3.26}$$

where g is the pulse shape (in a more general case there could be a finite number of pulse shapes), the sequence $\{a_k\}$ represents discrete data, and $-kT$ describes that the pulse shape is shifted so that the original $t = 0$ now is at $t = kT$. The name PAM means Pulse Amplitude Modulation, and the last two words indicate that the sequence $\{a_k\}$ has effect on the amplitude of the pulses. Figure 3.5 shows an example of a PAM signal for a specific sequence and a simple pulse shape. Note that the pulses in general are **overlapping** as in the figure. We shall always assume that $\{a_k\}$ is stationary, and we also present the result if $\{a_k\}$ is a sequence of independent random variables, but the general discussion is valid to any stationary process with known autocorrelation function $R_A(k)$.

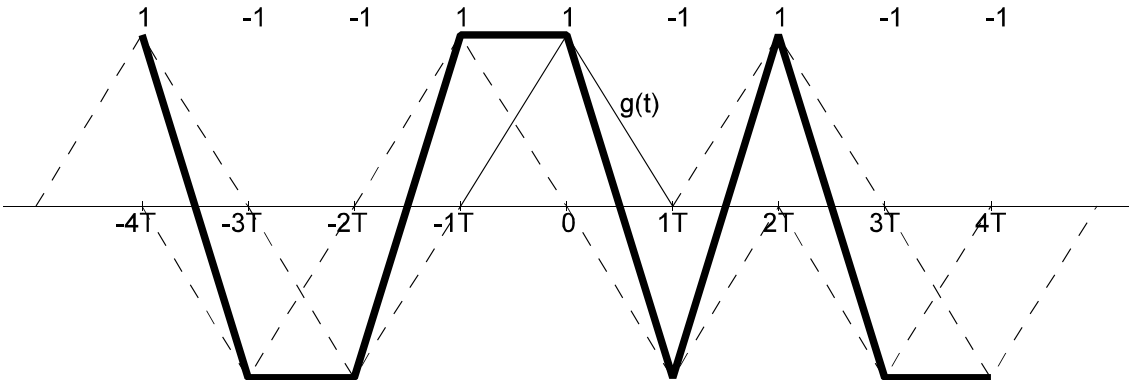

Figure 3.5
Example of PAM-signal (solid bold line) for a specific sequence and the simple triangular pulse shape g(t) shown at $t = 0$ and shown in dashed versions multiplied by a_k and shifted to kT.

When the pulses represent transmitted data, 1/T is referred to as the symbol rate (or baud rate, cf. Chapter 5). If we consider a particular signal, the phase of the clock defining the origins for the pulses is fixed. In order to generate the signal (3.26) from the data sequence and a known pulse shape, we usually introduce a higher sampling frequency by dividing T by a suitable integer (typical values of the **oversampling factor** (*oversamplingsfaktor*), m, are from 2 to 16), $mT_s = T$. We shall return to discussion of the selection of m later in this section. When the signal is generated (in the transmitter), the two clocks are in phase, and we only need to know $g(\ell T_s) = g(\ell T/m)$, ℓ integer.

In order to describe the stochastic process, we need to specify the sample space, i.e. the set of possible functions. We could construct a sample space for signals sharing a specific clock phase but having random data. Such a stochastic signal would not be stationary, since the probability distribution could be quite different for multiples of T and for other points in time. This is clearly seen from Figure 3.5 where the outcome v(t) assumes only two values at kT but in between may assume other values such that the distribution varies with time. If we extend the definition (3.8) of the autocorrelation to this non-stationary case, we may calculate

$$\begin{aligned} R_V(t+kT_S,t) &= E[v(t+kT_S)v(t)] = \underset{a_r a_s}{E}\left[\sum_{r=-\infty}^{\infty} a_r g(t-rT+kT_S)\sum_{s=-\infty}^{\infty} a_s g(t-sT)\right] \\ &= \sum_{r=-\infty}^{\infty}\sum_{s=-\infty}^{\infty} E[a_r a_s] g(t-rT+kT_S)g(t-sT) \\ &= \sum_{p=-\infty}^{\infty} R_A(p)\sum_{s=-\infty}^{\infty} g(t-pT-sT+kT_S)g(t-sT) \end{aligned}$$

where we used the stationarity of $\{a_k\}$ to calculate $R_A(p)$ for $p = r - s$. The R_V-autocorrelation is periodic in time t with a period of $T = mT_S$, and such processes are known as **cyclo-stationary**. If we are not concerned with synchronization, we may consider a bigger sample space where all (m) clock phases occur with equal probability. This makes the process stationary, and we can proceed to find the autocorrelation function. If we were to give a more realistic description of the receiver, we might use a sampling period of T_S, but a random phase offset, or we might even include a small difference between the sampling rates. The average over the m clock phases may be calculated in any interval, e.g. $0 \le \delta T_S < T\ (= mT_S)$, where each phase is assumed with the probability 1/m:

$$\begin{aligned} R_V(kT_S) &= \frac{1}{m}\sum_{\delta=0}^{m-1}\sum_{p=-\infty}^{\infty} R_A(p)\sum_{s=-\infty}^{\infty} g(\delta T_S-pT-sT+kT_S)g(\delta T_S-sT) \\ &= \sum_{p=-\infty}^{\infty} R_A(p)\frac{1}{m}\sum_{\delta=0}^{m-1}\sum_{s=-\infty}^{\infty} g(\delta T_S-sT+kT_S-pT)g(\delta T_S-sT) \\ &= \sum_{p=-\infty}^{\infty} R_A(p)\frac{1}{mT_S}R_g(kT_S-pT) \\ &= \frac{1}{T^2}\sum_{p=-\infty}^{\infty} TR_A(p)R_g(kT_S-pT) \end{aligned}$$

since the double sum over δ and s calculates the autocorrelation function by (2.25) of g sampled at sampling interval of T_S. The sum in the last line is a convolution of the autocorrelation functions for the pulse and for the data sequence. Since a convolution is transformed as a product, we find the spectrum as the product of the power spectrum of the

data signal and the energy spectrum of the pulse Thus we have calculated the spectrum of a PAM signal in the simple case where the clocks are synchronized and the phases are mixed to give a stationary signal. The resulting power spectrum is:

$$S_V(\omega) = \frac{|G(\omega)|^2}{T^2} S_A(\omega) = \frac{|G(\omega)|^2}{T} \sum_{p=-\infty}^{\infty} R_A(p)\, e^{-jp\omega T} \tag{3.27}$$

Even though the proof was done for a sampled pulse, the result also holds for a time continuous pulse (we may just let $m \to \infty$). We shall provide some examples below and later in Section 5.8 more examples will be found.

A very simple situation is found when $\{a_k\}$ is a stationary sequence of statistical independent values. Then

$$R_A(0) = E[A^2], \quad R_A(n) = E[A]^2 \text{ for } n \neq 0$$

Using $\sigma_A^2 = E[A^2] - E[A]^2$ and the identity

$$T \sum_{n=-\infty}^{\infty} 1 \cdot e^{jn\omega T} = \sum_{\ell=-\infty}^{\infty} \delta\left(\omega - \ell \frac{2\pi}{T}\right)$$

in (3.27) gives

$$S_v(\omega) = \frac{\sigma_A^2}{T} |G(\omega)|^2 + \left(\frac{E[A]}{T}\right)^2 \sum_{\ell=-\infty}^{\infty} \left|G\left(\frac{2\pi\ell}{T}\right)\right|^2 \delta\left(\omega - \frac{2\pi\ell}{T}\right) \tag{3.28}$$

The power spectrum consists of a continuous part (first term) and a discrete part where the power is concentrated at the frequencies $2\pi\ell/T$. The discrete part is sometimes called a **line spectrum** (*liniespektrum*). Examples of spectra calculated by (3.28) are seen in Figure 3.6 and Figure 3.7. The purpose of using PAM signal is to convey the information about the sequence $\{a_k\}$ to the receiver and the periodic part of the spectrum (last term) does not provide such information. Thus it is a waste of power to transmit it, and it is easily seen how it is avoided: the expected value E[A] of $\{a_k\}$ shall be made 0. The information about the signal frequency is in the periodic part, and in some way the receiver has to do operations to provide this signal (symbol synchronization). We shall return to this problem in Section 5.7.

From (3.27) it is seen that the pulse shape is very important for the spectrum, and in the following two examples we shall consider two very popular pulse shapes.

Example 3.10 Return to zero (RZ) pulse

The pulse

$$g(t) = \begin{cases} 1 & |t - T/2| \le T/4 \\ 0 & \text{ellers} \end{cases}$$

is known as **return to zero, RZ** since the high value is found in half of the pulse duration. Thus g(t) is transmitted for 1 and 0 for 0 (Illustrated in Figure 3.6). If the binary sequence is stationary with independence between symbols, and the probability for 1 is p, (3.28) with (from Example 2.5)

$$|G(\omega)| = \left| \frac{T}{2} \frac{\sin \omega T/4}{\omega T/4} \right|$$

$\sigma_A^2 = p(1-p)$, and $E[A] = p$ may be used. The spectrum is

$$S_V(\omega) = T \frac{p(1-p)}{4} \left(\frac{\sin \omega T/4}{\omega T/4} \right)^2 + \frac{p^2}{4} \sum_{\ell=-\infty}^{\infty} \left(\frac{\sin \pi \ell/2}{\pi \ell/2} \right)^2 \delta(\omega - \frac{2\pi\ell}{T})$$

Some of the frequencies in the discrete part ($4\pi\ell/T$, $\ell \neq 0$) are cancelled by the pulse, but contributions from the odd ℓ are still found (cf. Figure 3.6). Intuitively it is easy to understand that a long sequence of zeros makes it difficult to maintain the synchronization with the symbols. The discrete part of the spectrum disappears completely if $\{a_k\}$ instead is ± 1 (with probability ½). In that case some processing (e.g. absolute value) is needed to produce the discrete part of the spectrum. The RZ-pulse is often used in connection with a processing of the original $\{a_k\}$-sequence (line coding, Section 5.8), which changes the spectrum since the assumption about independence no longer holds. Most line codes that we know of remove the discrete spectrum.

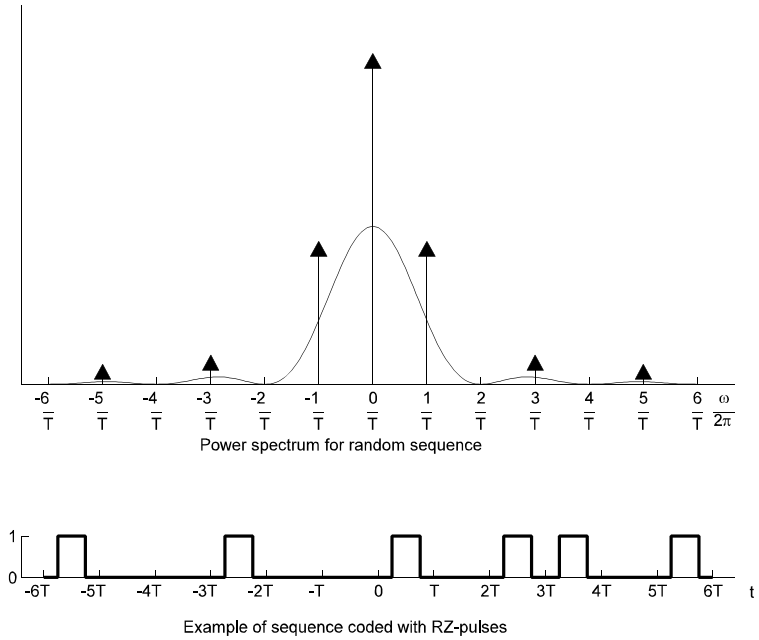

Figure 3.6
Power spectrum for random sequence of RZ-pulses
and example of such a sequence. The arrows
indicate δ-functions

Example 3.11 Non return to zero (NRZ) pulse

The pulse g(t) = 1 for $0 \le t < T$ is known as **non return to zero, NRZ** and an example of a sequence of such pulses is shown in Figure 3.7. Thus g(t) = 1 is transmitted for 1 and 0 for 0. Now we have (from Example 2.5)

$$|G(\omega)| = \left| T \frac{\sin \omega T/2}{\omega T/2} \right|$$

If the binary sequence is stationary with independence between symbols, and the probability for 1 is p, $\sigma_A^2 = p(1-p)$, and E[A] = p inserted into (3.28) gives the spectrum

$$S_V(\omega) = Tp(1-p)\left(\frac{\sin \omega T/2}{\omega T/2} \right)^2 + p^2 \sum_{\ell} \left(\frac{\sin \pi \ell}{\pi \ell} \right)^2 \delta(\omega - \frac{2\pi \ell}{T})$$

$$= Tp(1-p)\left(\frac{\sin \omega T/2}{\omega T/2} \right)^2 + p^2 \delta(\omega)$$

The discrete spectrum is cancelled (by the pulse) for all frequencies but 0. This example shows that E[A] ≠ 0 not always results in a line spectrum. The spectrum is shown in Figure 3.7.

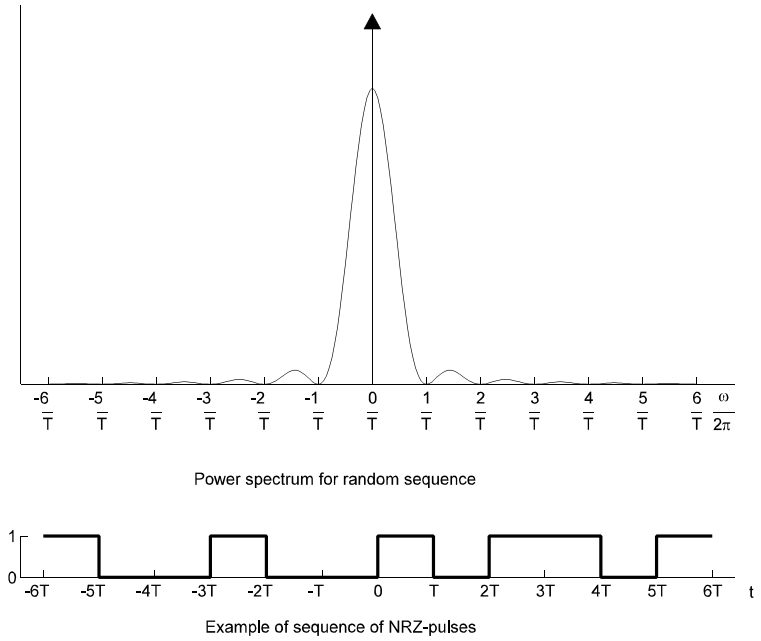

Figure 3.7
Power spectrum for random NRZ-sequence
and example of such a sequence.

As for RZ-pulses, NRZ is also often used, where $\{a_k\}$ is not independent resulting in a different spectrum.

While simple pulses like the ones in the examples above are often described in the time domain and have only a duration of T seconds, more sophisticated pulses are designed to have **limited bandwidth**, and they are first specified in the frequency domain. The transform to the time domain often gives a pulse which extends over several times T (in principle it is often of infinite duration). We shall describe the design of such pulses and emphasize the design of sampled pulses.

The data sequence has a periodic spectrum, $S_A(\omega)$, with period $2\pi/T$. This spectrum appears as the last factor in (3.27). The spectrum of the pulse should be approximately constant over the first period of the spectrum to produce a main lobe of the transmitted spectrum representing the information in the signal. However, the pulse spectrum should suppress the following periods (which are often referred to as side lobes as for antenna

radiation patterns), which take up additional bandwidth without adding to the information. By using an oversampling factor, m as discussed above, we increase the frequency range of the sampled pulse by a factor m. Within this frequency range we may specify $G(\omega)$, but it will then repeat periodically. Thus the bandwidth where we can control $G(\omega)$ is

$$-m\frac{\pi}{T} \le \omega \le m\frac{\pi}{T} \quad \text{or} \quad -m\frac{1}{2T} \le f \le m\frac{1}{2T}$$

Taking the Fourier series of the periodic G we obtain a sampled pulse, usually infinite. In order to get a practical design, we may sample $G(\omega)$ to get a finite number of samples, N, and apply the discrete inverse Fourier transform to obtain a pulse g(t) of finite length, say $K \cdot T$, i.e. the pulse is zero outside

$$-\frac{K}{2}T \le t \le \frac{K}{2}T$$

A typical value is 4T. The time interval is divided into N slots of length T/m which gives

$$N \cdot \frac{T}{m} = K \cdot T \quad \Rightarrow \quad N = m \cdot K$$

and the sample intervals for $G(\omega)$ at the frequency axis then become

$$\frac{2m\pi}{NT} = \frac{2\pi}{KT} \quad \text{or} \quad \frac{1}{KT} \text{ [Hz]}$$

In the following example we shall use this method to generate a sampled pulse. This sampled pulse can now be used to generate the signal (3.26). For each interval of length T, N/m pulses have to be added to produce the signal.

Example 3.12 Oversampled pulse with limited bandwidth

We want to have G to be constant up to $\omega = 1/4 \cdot 2\pi/T$, and then fall off linearly to become zero at $3/4 \cdot 2\pi/T$ as shown in the following figure. Using an oversampling of m = 4, we extend the frequency range to $\pm 2 \cdot 2\pi/T$ (as shown in the figure) and choose a duration of 4T (as the sampling of $G(\omega)$ shown in the figure). Since K = 4, we should use an `ifft` of length $N = m \cdot K = 16$ to get the sampled pulse in the time domain. That is what is described above, but a careful look reveals that dividing the frequency or time axis into 16 intervals requires that the functions are calculated for 16+1 points. It is no problem for the frequency representation

because the function G is anyway assumed to be non-zero only in a small part of the frequency range, and the G(ω) becomes

$$\mathbf{G} = (1, 1, .5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, .5, 1)$$

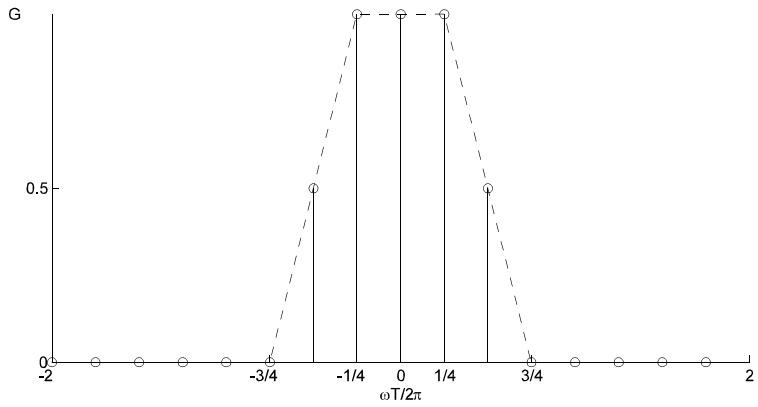

Figure 3.8
Spectrum of G.

where we start with frequency zero, and the ambiguous frequency in the middle (the two endpoints) does not contribute. The corresponding time function, however, may not be zero at the endpoints and something has to be done. From G(ω) (at the frequencies specified) we want to determine the pulse

$$\mathbf{g} = (g_{-\frac{N}{2}}, \ldots, g_0, \ldots, g_{\frac{N}{2}})$$

which is symmetric around 0, (N assumed even), and zero outside the values shown. Note that the pulse has N+1 (possibly non-zero) elements with distance T_S = T/m. The relation between the pulse and its transform is as always given in Equation (2.2) and the problem is how to use `ifft` to actually calculate **g**. Inserting in (2.2) we have

$$\begin{aligned} G\left(p\cdot\frac{2\pi}{NT_S}\right) &= T_S\sum_{q=-N/2}^{N/2} g_q e^{-jp\frac{2\pi}{NT_S}qT_S} \\ &= T_S\left(\sum_{q=0}^{N/2-1} g_q e^{-j2\pi\frac{pq}{N}} + g_{\frac{N}{2}}e^{-j\pi\frac{p}{2}} + g_{-\frac{N}{2}}e^{j\pi\frac{p}{2}} + \sum_{q=-1}^{-N/2+1} g_q e^{-j2\pi\frac{pq}{N}}\right) \\ &= T_S\sum_{n=0}^{N-1} f_n e^{-j2\pi\frac{pn}{N}} \end{aligned}$$

The coefficients f_n are the result of using `ifft(G)/Ts` (cf. Equation (2.4)), and the relation to **g** is

$$f_n = \begin{cases} g_n & \text{for } 0 \le n \le \frac{N}{2}-1 \\ g_{-\frac{N}{2}} + g_{\frac{N}{2}} & \text{for } n = \frac{N}{2} \\ g_{n-N} & \text{for } \frac{N}{2}+1 \le n \le N-1 \end{cases}$$

This relation is like the relation used by `fftshift` except for

$$g_{-\frac{N}{2}} = g_{\frac{N}{2}} = \frac{1}{2} f_{\frac{N}{2}}$$

In the following script we show the calculations in MATLAB:

```
T=1;
m=4;   %Oversampling factor
K=4;   %Length(/T) of time pulse
% Pulse in frequency domain:
Ts=T/m;
N=K*m;
G=[1 1 .5 zeros(1,N-5) .5 1];
% real is used to remove negligible imaginary parts
% ifft scaled as in (2.4)
g=fftshift(real(ifft(G)/Ts));
% Completely symmetric pulse:
g=[g(1)/2 g(2:end) g(1)/2];
t=-N/2*Ts:Ts:(N/2)*Ts;
stem(t,g)
```

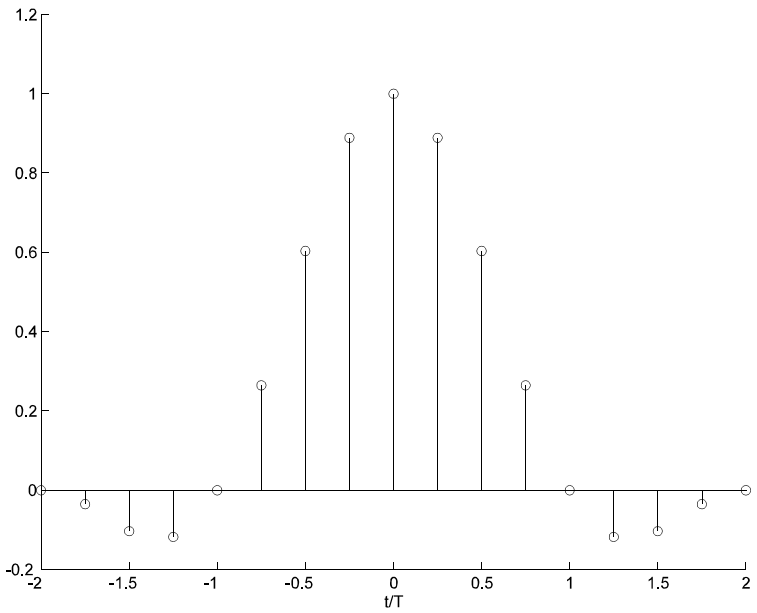

Figure 3.9
Oversampled pulse with limited bandwidth.

It may be seen that the resulting pulse was actually zero at the endpoints, so in this case there was no need to extend the pulse with $g_{-\frac{N}{2}} = g_{\frac{N}{2}} = \frac{1}{2}f_{\frac{N}{2}}$. In some cases, especially in Chapter 5, the energy of the pulse is an issue and this requires proper scaling of `ifft` as done above. Remember from (2.20) that the energy in the frequency domain is calculated summing $G_n^2/(T_S N)$.

Limiting the pulse in the time domain always results in some side lobes as described before the example. To try to see these side lobes we further increase the frequency range:

```
% Transfer function of g with frequency resolution 1/(32T):
g_T=fftshift([zeros(1,56) g zeros(1,55)]);
% fft scaled as in (2.4)
G_T=abs(fft(g_T)*Ts);
semilogy([-2:1/32:63/32],fftshift(G_T));
```

The result is seen in the following figure.

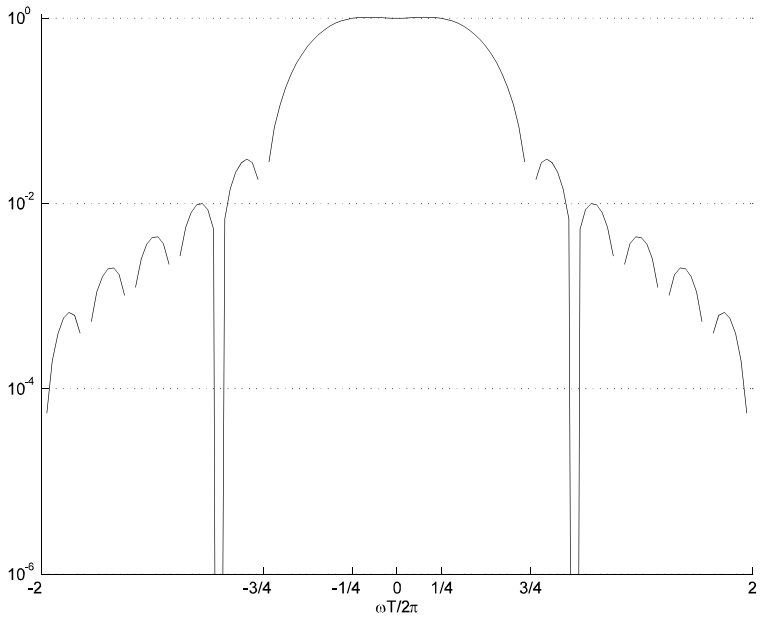

Figure 3.10
Spectrum (absolute value) of actual pulse.

We have used a logarithmic y-axis to see the very small side lobes. The spectrum actually has a zero at $3/4 \cdot 2\pi/T$ and due to the design also zeros at the following frequencies $p/4 \cdot 2\pi/T$, but small side lobes are found in between these zeros. Choosing a linear y-axis also reveals that the spectrum is not quite constant in the interval $\pm 1/4 \cdot 2\pi/T$. Remember that it is the choice of T_S (i.e. m) that describes the

frequency range controlled by the sampled pulse shape. The spectrum repeats itself with distance $2\pi/T_S = m \cdot 2\pi/T$ so some filtering has to be done to remove the copies outside what we see in the figure.

A higher oversampling factor is obtained by increasing the length N. In this way we increase the frequency range that we control with the sampled pulse shape. Below we show the result for m = 8 with the same program as before.

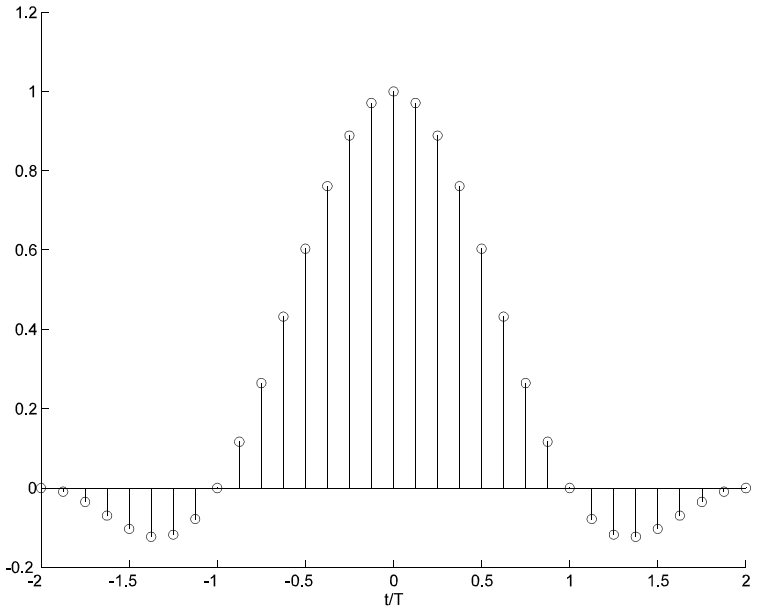

Figure 3.11
Pulse from Figure 3.9 with higher oversampling factor.

Another modification could be to have a longer part of the pulse. This modification will reduce the side lobes. The steps on the frequency axis then have to be modified:

```
KL=8;   %Length(/T) of time pulse
NL=KL*m;
GL=[1 1 1 .75 .5 .25 zeros(1,NL-11) .25 .5 .75 1 1];
gl=fftshift(real(ifft(GL)/Ts));
gl=[gl(1)/2 gl(2:end) gl(1)/2];
t=-NL/2*Ts:Ts:NL/2*Ts;
stem(t,gl)
xlabel('t/T');
```

The result is

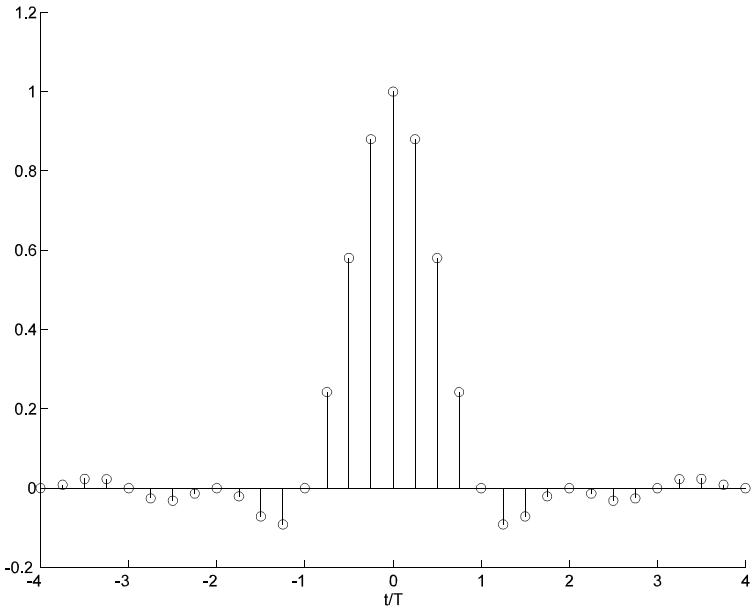

Figure 3.12
Extended version of pulse from Figure 3.9.

In digital implementations another problem arises: The filter coefficients have to be represented with a finite number of bits and so do the signals - quantization which is to be described further in Section 4.3. Now we shall give a short demonstration of the effect of quantization of the coefficients. In a figure similar to Figure 3.10, we show the spectrum of the pulse for quantization of the filter coefficients in 128 levels (7 bits) and 1024 levels (10 bits). It is seen that the coarse quantization gives a spectrum with higher side lobes, whereas 10 bits are difficult to distinguish from the spectrum of Figure 3.10. If this has any importance depends on the application, e.g. restrictions on the use of frequencies, and we shall return to this in some examples later. A similar consideration has to be given to the quantization of the input and output signals and to the precision of the coefficient multipliers. More about realization of impulse responses with finite precision is found in [3.5]

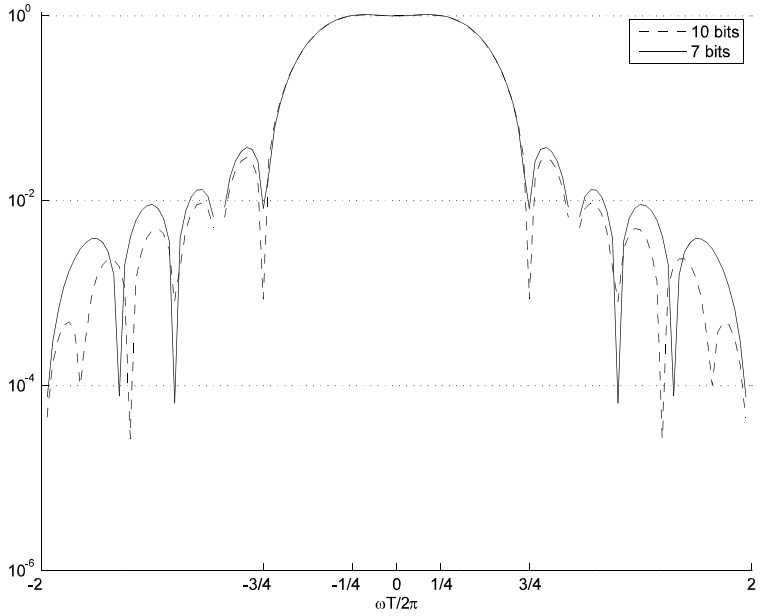

Figure 3.13
Spectrum (absolute value) of actual pulse for two different quantizations of the coefficients.

3.6 Sampling of stochastic signals

We have earlier, in Section 2.2.2, discussed sampling and reconstruction of deterministic signals and stated the spectrum of the sampled signal in (2.6). There are a few differences when the signal is a stochastic process.

If the signal x(t) is a stationary stochastic signal with continuous time autocorrelation function $R_X(\tau) = E[x(t)x(t+\tau)]$ and power spectrum $S_X(\omega)$, and it is sampled with the sampling frequency $\omega_s = 2\pi/T$, the result is a stationary sequence with autocorrelation function

$$\tilde{R}_X(k) = R_X(kT)$$

It follows directly from the definition of the autocorrelation function that the discrete time function, $\tilde{R}_X(k)$, is simply a sampled version of the continuous time function, $R_X(\tau)$. Since the autocorrelation function is deterministic, we can use (2.6) to obtain the power spectrum (the transform of the sampled autocorrelation function) of the sequence:

$$\tilde{S}_X(\omega) = \sum_{n=-\infty}^{\infty} S_X(\omega - n\omega_S) = \sum_{n=-\infty}^{\infty} S_X\left(\omega - n\cdot\frac{2\pi}{T}\right) \qquad (3.29)$$

Note that the phase of the sampling points has no influence on the spectrum. In the deterministic case it would usually not be possible to express the energy spectrum of a sampled signal since the terms in the sum would have to be added with the correct phase.

If a stochastic signal is band limited, i.e. no components outside some frequency W, it may be reconstructed in the same way as a deterministic signal, i.e. by a lowpass filter which filters out exactly the n = 0-component of the sum (3.29). This requires that $\omega_S > 2W$. The exact interpretation is that if x' is the reconstructed signal, then $E[(x-x')^2] = 0$. We shall not give the proof of this theorem which may be found in e.g. [3.4]. If the stochastic signal is not exactly band limited then aliasing is found as for deterministic signals since components in the above sum overlap. In practical systems, the stochastic signal is often passed through a lowpass filter, an **anti-aliasing filter**, before the sampling. The total process of sampling and reconstruction is illustrated in Figure 3.14.

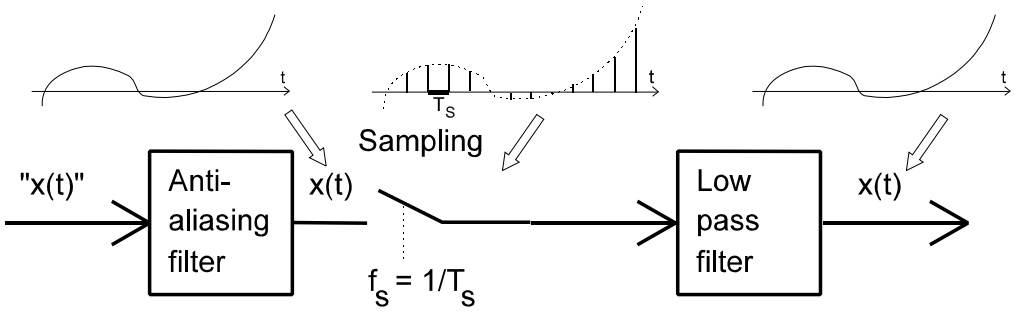

Figure 3.14
Sampling of analog signal followed by its reconstruction.

There is often a need for **changing the sampling frequency** of a discrete time signal. The first sampling of the continuous signal may use a high sampling rate in order to make the filtering less critical, but subsequent stages in the signal processing may be performed at reduced sampling rates.

A stochastic discrete time signal with power spectrum $S_X(\omega)$ may be changed in such a way that only every m'th value is kept, i.e. the distance between sampling points is increased from T to mT. This reduction of sampling rate is often called **decimation** (*decimering*) with a factor m (all m, even though the name may indicate that m has to be 10). The spectrum of the sub-sampled signal, $\tilde{\mathbf{S}}_{\mathbf{x}}'(\boldsymbol{\omega})$, may be expressed as

$$\tilde{S}_X'(\omega) = \sum_{k=0}^{m-1} S_X(\omega - k\omega_S') = \sum_{k=0}^{m-1} S_X(\omega - k\cdot\frac{2\pi}{mT}) \tag{3.30}$$

where the reduced sampling frequency is $\omega_s' = 2\pi/mT$. This result follows from (3.29) when the two different sampling frequencies are used and we remember that $S_X(\omega)$ is periodic with period $2\pi/T$:

$$\tilde{S}_X'(\omega) = \sum_{n=-\infty}^{\infty} S_X(\omega - n\cdot\frac{2\pi}{mT}) = \sum_{p=-\infty}^{\infty}\sum_{k=0}^{m-1} S_X(\omega - mp\cdot\frac{2\pi}{mT} - k\cdot\frac{2\pi}{mT})$$

Looking at the basic period of $2\pi/T$ we have (3.30). Note that the spectrum $S_X(\omega)$ may originate from an inherently discrete time process or it may be a spectrum of a sampled signal as $\tilde{S}_X'(\omega)$ in (3.29) in which case the left hand side of (3.30) could have been "tilded" twice. If the signal is sampled and should be recoverable after the decimation, the original spectrum $S_X(\omega)$ should be very narrow such that there is no overlap in the sum of m terms.

The sampling frequency may be increased, or the position of the sampling points changed by **interpolation**. Often a first stage towards increasing the sampling frequency by m is to insert m−1 zeros between samples (MATLAB function `upsample`). This process increases the bandwidth of the signal by a factor m, but the only change in the power spectrum is a reduction of the power by the same factor as the following calculation where T' = T/m shows

$$\tilde{R}_X(k) = \begin{cases} R_X(n) & k = n\cdot m \\ 0 & k \neq n\cdot m \end{cases}$$

$$\tilde{S}_X(\omega) = \sum_{k=-\infty}^{\infty} T'\tilde{R}_X(k)e^{-j\omega kT'} = \sum_{n=-\infty}^{\infty} \frac{1}{m}T R_X(n)e^{-j\omega nT} = \frac{1}{m}S_X(\omega)$$

Thus we observe m periods of the (periodic) power spectrum $S_X(\omega)$ in the range $-m\pi/T \le \omega \le m\pi/T$. The next stage in the interpolation involves some kind of filtering giving the values for the intermediate samples (where the zeros were inserted). The sampling theorem, [3.6], shows that perfect interpolation is done using a (sin x)/x-type of interpolation function (a lowpass filter if it was the original analog signal we wanted). However, this is

difficult to implement, so other solutions have to be found. For an introduction, see [3.5, 3.7].

Some caution is needed in applying the sampling theorem to communication signals. In Section 3.5 we discussed how a stationary process could be constructed by taking the average over signals with different clock phases. In a digital receiver it is often desirable to use a sampling frequency which is a multiple of the clock frequency (or close to that value). In this case the power spectrum given by (3.29) is the average over all clock phases, but any particular sample signal may have different properties. In this case the safe approach is to take the sample values of the pulse shapes, and then consider the autocorrelation function of the discrete time signal. Sampling of a signal that involves modulation may give rise to similar problems.

3.7 References for Chapter 3

3.1 Jim Pitman: "Probability", Springer-Verlag, 1993, ISBN 3540979743.

3.2 G. Grimmett and D. Stirzaker: "Probability and Random Processes", Oxford University Press, 2001, ISBN 0-19-857222-0.

3.3 J.F. Wakerly: "Digital Design: Principles and Practices (3rd Edition)", Prentice Hall, 2000, ISBN 0-13-089896-1.

3.4 J.G. Proakis & M. Salehi: "Communication Systems Engineering. (2nd Edition)", Prentice Hall, 2002, ISBN 0-13-095007-6.

3.5 J.G.Proakis & D.G. Manolakis: "Digital Signal Processing (4th Edition)", Prentice Hall, 2006, ISBN 0-13-187374-1.

3.6 B.P. Lathi: "Signal Processing and Linear Systems". ISBN 0-19-521917-1, Oxford University Press N.Y., 1998.
(Used in the DTU courses 31605 and 31606: Signals and Linear Systems)

3.7 R.E. Crochiere & L.R. Rabiner: "Multirate Digital Signal Processing". Prentice Hall, 1983.

4. USER SIGNALS

Most natural signals are **analog signals**, i.e. signals which may assume any value in a continuous **signal range** (*udstyringsområde*), and in old times they were transmitted as such, e.g. telephony to up about 1970. As explained before, the signals are now converted into digital form, and then transmitted, and this chapter shall explain this conversion in detail after describing the different signals. We shall describe the analog signals, mainly voice signals, and the requirements for the transmission channel. After that, we shall deal very much with the errors and distortion introduced by conversion, and we shall introduce some more precise measures in the following during discussion of each particular form of signal. The basic conversion and a mean square error criterion, the signal-to-quantization-noise-ratio, are introduced in Sections 4.3 and 4.4, and we shall deal with some more advanced methods in the Sections 4.2 and 4.6. These methods reduce the requirements to the bit rate of the digital signal. The theoretical foundation for some of the methods in Section 4.6 will be given in Section 4.5.

4.1 Analog signals

The most important analog signal source is the human voice (*stemme*) resulting in speech (*tale*) (or singing etc.), but there exist also a number of other sources, see in Section 4.1.2.

4.1.1 Speech signals

The study of the human voice is made difficult by the physiological aspects of sound production, and, in addition, the psychological elements (temper, mood etc.) which influence it. First, we shall describe how speech is produced and (later) perceived.

Speech may be split into a sequence of sound elements, the so-called **phonemes** (*fonemer*), each of which may be assumed quasistationary. The number of phonemes varies with the language. As an example, we may take Danish with 39 phonemes of which 21 are vowel

sounds and 18 consonants [4.1]. The time structure of speech is irregular: Words and phrases are separated by pauses (> 100 ms) which represent about 50% of the time during a monolog and about 75% for each part in a dialog. The rate of the speech varies from person to person, but 80 to 200 words per minute are typical rates.

The most simple phonemes are the **vowels** (*vokaler*). These are generated by the vocal chords (*stemmebånd*) as a periodic sequence of pulses where the repetition frequency is called the **pitch (frequency)** (*grundfrekvensen*). A spectral analysis shows a spectrum with many significant harmonics, i.e. 20-50, each with about the same size. When this signal passes the windpipe (*luftrør*), the throat, and the mouth, the spectrum is shaped so that the many spectral lines now have an envelope characteristic to each vowel. The main characterizing parameters are the placement of the first 4 peaks, the so-called **formants** (*formantfrekvenser*). These are so characteristic that a vowel spoken with another pitch still has the same envelope, but the number of lines under the envelope varies of course with the pitch. The pitch is varied by tightening the vocal chords. The pitch is varied during normal speech to express certain things, e.g a question, and the pitch of course varies from one individual to the next (bass to soprano). Male voices have a pitch of 80 to 200 Hz, female voices from 150 to 400 Hz and children are even higher.

Typical formant frequencies (in Hz) for some Danish vowels spoken by male voices are:

Vowel	f_1	f_2	f_3	f_4
long o	320	640	2400	3500
short a	600	940	2640	3200
long i	280	2200	3100	3500

Despite the statement about the significance of the formants, female and children voices have slightly higher formants due to the smaller cavities shaping the spectrum. For the recognition of the vowels the most important formants are f_1 and f_2. The duration of the vowel sounds in normal speech is about 100 ms.

The **consonant sounds** (*konsonantlydene*) are more complicated than the vowel sounds. For voiced (*stemte*) consonants, e.g. l, m, n, or v, the spectrum is like the vowels, i.e. a line

spectrum. The unvoiced (*ustemte*) consonants like f or s do not use the vocal chords as the source, but noise generated by passage of the air through narrowings possible at a number of places, e.g. between the tongue and the palate, and their characteristic spectrum is continuous rather than a line spectrum. The duration of consonants is smaller than that of the vowels, in average 25-50 ms.

Both the frequency range and the dynamic range for speech are smaller than the normal perception range of the ear. Fortunately, experience has shown that the speech quality is sufficient even if the frequency range is limited, since frequency components above 3400 Hz may be cut off, and the same goes for frequencies below 300 Hz even if it cuts away the pitch frequency. It is still possible for the listener to recognize the pitch since the hearing reconstructs the pitch presumably from the distance between the lines in the rest of the spectrum. The **dynamic range** (*dynamikområdet*), which is defined as the ratio between the faintest and the strongest levels, may also be smaller than in the original speech. It should be emphasized that here the interesting property is the perception of speech, not whether it sounds good. There is no simple objective measure for speech quality, and thus subjective testing has to be done. In a subjective test, a number of persons listen to the speech and present some evaluation of the quality. In Section 4.2 we shall present such measures of speech quality.

As described above, it is the shape of the spectrum that seems to be the most important feature in perception of speech. The hearing is relatively insensitive to the exact timely progress of the signal. This has given significant savings in systems for transmission of speech. To phrase it a bit different, the hearing does not distinguish between two signals having the same amplitude spectrum, but different phase spectra. Thus, the ability of a given system to transmit speech signals is not evaluated only by considering the transfer function or by looking at the output signal waveform compared to the input waveform. The signals may look quite different and still represent the same speech sound.

For use as models later, we give some stochastic description of speech. The distribution of the amplitude is very complex and varies much during time. Analyzing sequences of speech that is bandlimited to the above range and sampled gives the result, [4.2], that the amplitude distribution can be roughly approximated by the Laplacian-density (cf. Example 3.1):

$$f(x) = \frac{1}{\sqrt{2}\sigma} e^{-\frac{\sqrt{2}|x|}{\sigma}}$$

but the variance σ^2 (the power) varies rather much from one person to another. The total emitted power is very small, typically 10 μW in average. The dynamic range is about 40 dB (i.e. 1:100 measured as amplitude). Measurements of the power spectrum show that components are found in the frequency range 80 Hz to 12 000 Hz, but most of the power is found at the pitch frequency and up to the first formant frequency which is in the range from 80 to 400 Hz. The information content, however, is found at somewhat higher frequencies (at the formants, 300 to 2500 Hz) where the power is very limited.

Even if the phase of the signal does not mean much for the perception, it is still a problem that many sounds have a short duration (down to 25 ms), and thus it becomes necessary to limit the phase distortion (i.e. the delay variation) in the system. This is measured by the group delay and requirements for the variation of the group delay are shown below. The term 'group delay' may be known from linear systems, and it will be defined more properly later in this course, but here we can think of group delay as being the delay of a signal element with a certain frequency.

In the table below we summarize the end-to-end requirements from ITU-T [4.3, G.151, G.133, M.1040] for systems transmitting speech of telephony quality:

Bandwidth:	300 Hz - 3400 Hz
Dynamic range:	> 40 dB.
Group delay variation:	< 30 ms for 800 - 3400 Hz
	< 60 ms for 300 - 800 Hz

4.1.2 Other analog signals

Similar to speech signals, but with extended requirements, we find the analog **audio signals**. The requirements are extended in almost all aspects: bandwidth, distortion, dynamic range, and balance between the two channels in a stereo signal, e.g. [4.3, J-series]. For radio transmission of stereo signals, the left and right channel, L and R respectively, are combined together as:

$$S = 0.5R + 0.5L$$
$$D = 0.5R - 0.5L,$$

such that S represents the mono signal. This process is sometimes denoted matrixing.

4.1.3 Video signals

Some very different analog signals are **video signals**. Video signals are "only analog in some dimensions" which is due to the fact that two-dimensional, moving images are usually transmitted as a single time dependent signal obtained by **progressively scanning** the image along a number of lines from top to bottom. The images are transmitted in discrete time (as **frames**), i.e. a number of frames is transmitted each second (25 in Denmark). Each frame has a finite number of lines. These lines may be all lines in the image, or half that number if frames are transmitted alternating between even and odd numbered lines to reduce the flickering. The frame rate is then doubled to achieve the same number of images per second. This is called **interlacing**.

As it may already be known, a color image may be composed of three basic colors: red, green, and blue (**RGB**-signal, additive color mixing). The immediate impression from this is that a color video signal requires 3 times more transmission capacity than a monochrome signal. However, the sensitivity of the eye is not the same for all colors, so it is reasonable to form another signal on basis of the RGB-signal, and this signal describes the **luma** :

$$Y' = 0.30R' + 0.59G' + 0.11B'$$

The primes indicate that R, G, and B are **gamma corrected**, a non-linear operation to account for sensitivity of the human vision. (The name luma is to distinguish it from luminance (*lysstyrke* or *luminans*) based on the linear values of R, G, and B). The Y' signal is also used in monochrome receivers, and the specific linear combination was found after subjective testing. To convey the color information, the **chrominance** (*krominans*), various methods exist, known as different color spaces, [4.4]. They all use two more signals. For analog TV, a popular one is the **Y' UV**-model which uses:

$$U = -0.15R' - 0.29G' + 0.44B'$$

and

$$V = 0.62R' - 0.52G' - 0.10B'.$$

The linear combinations are also found by subjective testing. For digital TV, the chrominance signals use slightly different constants and the model is called **Y'CbCr**, [4.5 (Recommendations 601, 709)]. Since the ability to distinguish color differences is poorer than the ability to distinguish differences in luminance, the bandwidth for the latter two signals does not have be as good as that for the luma.

All systems (**NTSC**, **PAL**, or **SECAM**) for transmission of analog video signals use such three signal, but there are significant differences of how the stream of images is formed, [4.4] and transmitted. In the Western Europe standard (PAL), an image frame has 625 lines and the scanning is interlaced so that 50 half frames are transmitted each second. Thus the duration of a line is 64 μs, but only 52 μs are used for the analog image data. The remaining time is used for a line synchronization pulse such that the start of each line is clearly defined. Besides the line synchronization signal, frame synchronization is also found, and in some lines just after this signal, digital information such as teletext is transmitted. In addition to the image signal, one or two audio channels are found. In the newest stereo sound system (NICAM) the sound is coded as digital signals (see Section 4.4 or [4.6]). It follows from the introduction above that the bandwidth needed for an analog video signal is not so easy to determine, but about 7 MHz is sufficient for a color signal with sound. The luminance signal requires 5.5 MHz, while 1.3 MHz is sufficient for each of the chrominance signals [4.4, 4.5 (Recommendations 567-568)]. This does not necessarily increase the total demand for bandwidth since they are modulated in a clever way. The dynamic range needed is 40 dB.

Modern video production provides image frames that are digitized to **pixels** (picture elements) in the camera. The stream of images is encoded to reduce the amount of data before transmission through a transmission system, **Digital Video Broadcast, DVB**, of which several exist. The possibility to reduce the bandwidth for chrominance signals is often used in these, We shall shortly discuss digitally coded video below.

4.2 Coding of analog signals

By the term **coding** (*kodning*) we here mean representation of the analog signal by a stream of binary symbols. A complete coding system is shown in Figure 4.1. It consists of an **encoder** - sometimes plainly named **coder** (*koder*) - and a corresponding **decoder** (*dekoder*). The purpose of the decoder is to try to reconstruct the original signal in the best possible way based on the signal from the encoder - possibly disturbed by the transmission channel.

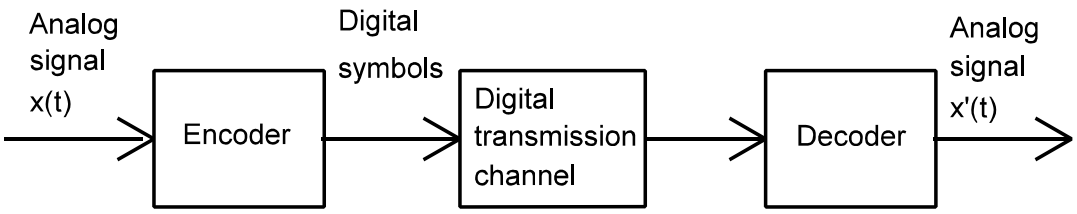

Figure 4.1
Coding of analog signals.

There exist many methods for encoding an analog signal, and the explicit selection of methods for a given coding problem depends on the application. Below we introduce some possible applications:

- Conversion to digital form because digital transmission or storage are basically advantageous (e.g. optical fibers or DVDs), but in addition, digital switching (telephone exchanges or cross connects) may be favorable for economic reasons.

- Reduction of redundancy in the signal to achieve a lower bit rate or a smaller amount of data than the more immediate conversion described above and still maintain a certain quality. In some cases this reduction is the only possible way to implement a system at reasonable costs, e.g. GSM. Methods for redundancy reduction can be classified into **lossless compression** designed to preserve the bit-exact form of the original (digitized) signal, and **lossy compression** that discards some information, but intends to preserve the perceptual similarity with the original content as well as possible.

- **Encryption** (*kryptering*) of the signal, i.e. making the signal secret in order to prevent outsiders from understanding it. Encryption is most effectively implemented by digital means, and this is yet another reason for that modern mobile phones are digital systems.

The entire process from coding the analog signal x(t) to its reconstruction as x'(t) will normally result in that x' deviates from x. The difference, e = x' − x, may be regarded as a **noise** (*støj*) that is added to the original signal x. We shall describe this noise in different systems. We start out in Section 4.3 about quantization by introducing the quantization noise which is a part of the coding process, but other forms of noise are also found. It is necessary to have some form of measure for the noise. In this course we shall use the objective measure **Signal-to-Noise Ratio, SNR** (*signal/støjforhold, SNR*). The SNR is defined as the ratio between the power of the signal and that of the noise, P_x and P_e, respectively:

$$\text{SNR} = \frac{P_x}{P_e} \quad \text{or logarithmic SNR} = 10\log\frac{P_x}{P_e}\ [\text{dB}] \tag{4.1}$$

The signal and the noise are stochastic processes, and if the means for these processes are zero, the power and the variance are identical. A large SNR implies that the original signal and the reconstructed signal are close together, and the users will perceive this as a good reconstruction. On the other hand, the opposite relation is not true, since as we discussed in Section 4.1.1 about speech signals, two signals may look very different (and thus have a poor SNR), but still be perceived as the same sound. This means that for subjective evaluation of the quality other measures may be more relevant. A simple modification of the defined SNR is to weight the power with the sensitivity curve of the human hearing (psophometric weighting, [4.3, O.41 and J.16]), but in the following we shall introduce two subjective quality measures. Reference [4.2, Appendix E and F] gives even more methods.

For digital video signals, **Peak Signal-to-Noise Ratio** (**PSNR**) is often used to estimate the quality. PSNR is computed as:

$$\text{PSNR} = 10\log\frac{\text{MAX}^2}{\text{MSE}}\ [\text{dB}]$$

MAX is the maximum possible value for a pixel. For example, if a pixel of a monochrome image is represented by 8 bits per sample, MAX = $2^8 - 1 = 255$. MSE is the Mean Square-Error between the pixels in the original (digitized) reference image R and the processed image P. For an image frame of size n·m, MSE can be defined as:

$$\mathrm{MSE} = \frac{1}{n \cdot m} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \left(P(i,j) - R(i,j)\right)^2$$

i.e. the mean energy of the difference between the two pictures. For color images, PSNR is often computed from the luminance component only (Y-PSNR). In this case, MSE is computed just from the Y-plane. It is also possible to compute PSNR values for the chrominance planes (U-PSNR and V-PSNR), or compute MSE jointly from all planes to obtain YUV-PSNR.

It is relatively easy to ask a panel of persons to decide which of two systems they judge to be better, or when they judge the two systems to be of equal quality. Using such a panel one may define a **subjective SNR** letting the panel compare the system with a system where the input signal x is corrupted by an artificially generated noise e_K. When the panel evaluates the two systems to be equivalent, the subjective SNR is the ratio between the average powers of x and e_K. Often white noise is used as e_K, but better results are obtained using noise that is correlated with the signal, e.g. $e_K = x \cdot r$, where r is a signal that assume two values $\pm\alpha$ with equal probability, [4.2]. The SNR for the reference system is α^{-2}.

A more direct method for evaluating subjective quality is **MOS**, Mean Opinion Score. In this method, test subjects are asked to grade the quality of the assessed audio or video content according to their subjective opinion. Usually, either a single stimulus or a double stimulus method is used. In the double stimulus method, one of the stimuli acts as a reference, to which the assessed stimulus is compared just as for the subjective SNR above. Different grading scales, either discrete or continuous, can be used. The most commonly used is a 5 step scale, where the steps are numbered from 5 down to 1, and described in words such as "excellent", "good", "fair", "poor", and "unsatisfactory". When the judgement has been given by the panel, the average and the standard deviation are calculated after assignment of numbers from 5 down to 1. The results of such experiments are clearly difficult to reproduce, but an indication of the reliability of the average result is of course

found by looking at the standard deviation. No system gives MOS = 5, but the much used 64 kbit/s system to be described in Section 4.4 gives a fairly good value of 4.53 with standard deviation 0.57, [4.2].

The partly subjective methods are not well suited to automated supervision of transmission quality, and standardization efforts are going on for specification of objective measurement methods. At the transmitter side, quality can be measured by comparing the processed digital content against the original digital content using Full Reference (FR) metrics. ITU-T has published recommendations for Perceptual Evaluation of Speech Quality (PESQ) and Perceptual Evaluation of Audio Quality (PEAQ) [4.7, P.862, P.862.2, BS.1387]. For video signals, PSNR is still the most widely used FR metric, but more accurate (and complex) metrics have been developed and even standardized in ITU-T Recommendation J.247 [4.8, J.247]. At the receiver side, quality monitoring is more challenging, since the original signal is not available. For this purpose, several so-called No Reference (NR) metrics have been proposed. They are intended to estimate features that are typically considered disturbing, such as white noise in audio or blockiness and blurriness in video. However, the performance of NR metrics is in general not as good as of FR metrics, and the development is still going on. Parametric quality models, such as E-Model, are popular especially for modeling networked speech quality. They use network parameters, such as delay and errors, together with information of the compressed audio (or video) quality to generate a quality estimate [4.7, 4.8].

As shown in Figure 4.1 there is often a transmission channel between encoder and decoder. The channel may be some ordinary transmission channel (cf. Section 1.4), maybe even the result of a large network, or it may be some storage media, e.g. a magnetic disk. In this model, the channel part includes error-correction etc. that might be necessary to obtain enough reliability in the transmission to provide a reasonable quality at the analog output. Note that no transmission channel is completely error-free. Using metallic or optical lines the error rate is low, but radio channels as in mobile phones have many errors. As described above, the quality is a complicated, subjective function which makes it even more difficult to select error-correcting methods for the channel. It is an important characteristic of a particular coding system how sensitive it is to the errors in the transmission and there are significant possibilities for a joint optimization of the coding system and the channel.

Basicly, two types of coding are found:

- **Waveform coding** (*kurveformskodning*) which may use the statistical properties of the source, but without a complex model of it. The decoder aims at reconstructing the waveform for the coded signal as good as possible, i.e. maximizing SNR. In Section 4.4 we introduced the most important method, PCM, and we shall present extensions of PCM in Section 4.6.

- **Source coding** (*kildekodning*) uses a very complex model for the signal source. The first success for such methods was obtained in speech coding, but in the last 10 years or so, much improvement has been done on image coding methods, e.g. **MPEG** coding for video signals, cf. [4.4].

The reference [4.2] gives an extensive treatment of methods for waveform coding of analog signals, both speech and video signals. As it may be seen from the courses in Data Compression [4.9] and Digital Video Technology [4.4] there has been a strong development of source coding methods such that they surpass the waveform methods far measured in bit rates and still maintain a reasonable quality.

Later in Section 4.6.3 we shall present the most advanced method for waveform coding of speech. Typical bit rates of waveform encoders for speech are from 16 kbit/s to 100 kbit/s. But the strong development of the source coding area has resulted in the possibility of transmitting understandable speech with bit rates from 2.0 to 10 kbit/s. These rates are achieved using very complex terminal equipment and cause some reduction of the quality. Users of GSM may compare the quality for the 13 kbit/s used in the mobile phone with the usual waveform encoded 64 kbit/s PCM in the fixed network. The quality of the source coder depends of course on the model used and the development has provided better and better models from the linear prediction coding to complicated models that use the so-called analysis-by-synthesis approach, Figure 4.2.

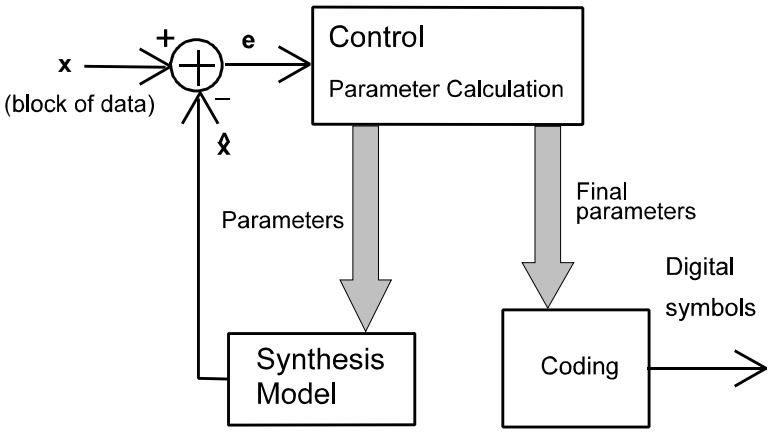

Figure 4.2
Analysis-by-synthesis for source coding.

In the analysis by synthesis approach, the model tries to synthesize the speech from a small number of parameters and then use some criterion to estimate the error between the synthesized signal, $\hat{\mathbf{x}}$, and the signal to be encoded. Normally a whole block of signal data is used as input, **x**, e.g. 20 msec speech. Adjusting the parameters and repeating this loop several times gives a minimum error and a parameter set which transmitted to the receiver synthesizes the signal there. In the range from 7 to 10 kbit/s such systems provide results close to **telephony quality** which is only slightly inferior to the quality of 64 kbit/s PCM. If the quality is allowed to be further reduced (**communication quality**), such models may even allow speech communication in the range 300-2000 bit/s, but the quality is bad (like "robot speech"), and it is hard to recognize the speaker and sometimes difficult to understand all phrases in all languages.

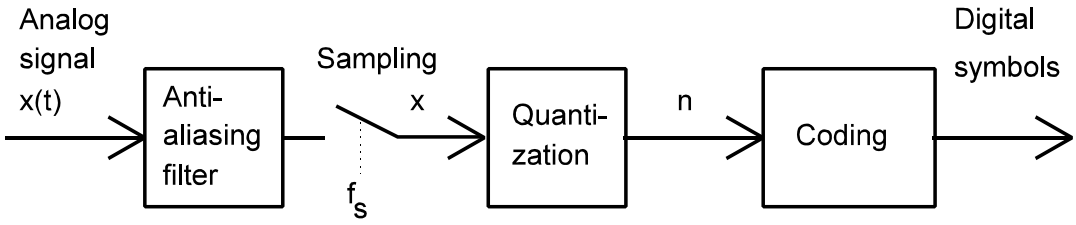

Figure 4.3
Coding of analog signals separated into subprocesses.

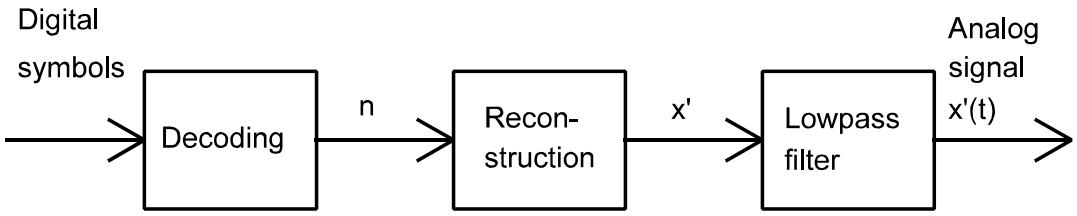

Figure 4.4
Decoding for reconstruction of analog signal.

It is practical to analyze the coding, i.e. the conversion from analog to digital form, as more processes each treated independently. As it is seen from Figure 4.3, it is only the last process that is explicitly known as coding, but we shall continue also to use this name for the whole process as done above. The conversion always involves sampling which we described in Section 3.6. In the figure we show explicitly the anti-aliasing lowpass filter before the sampling which is used to eliminate frequencies that result in aliasing. The corresponding decoder is shown in Figure 4.4 which also includes the lowpass filter for reconstructing the sampled signal. After the sampling, the values should be represented in digital form, i.e. a **quantization** (*kvantisering*) should be performed and the result of the quantization should be represented in digital (binary) symbols, **coding** (*kodning*). The quantization is treated in Section 4.3, while coding is found in Sections 4.4 and 4.6 where the first is a simple realization and the latter presents a more complicated system based on the theory presented in Section 4.5. It may be difficult in a system to see the difference between quantization and coding which is often implemented as one process, e.g. in a signal processor with an A/D-converter at the input. However, valuable insight is gained from treating the two processes independently as we shall see in the following sections.

4.3 Quantization

The purpose of quantization is to represent values in a continuous range from an analog source by only a finite number of values which then later may be mapped onto binary numbers in the coding process. Quantization introduces an irreversible distortion of the signal and characterization of this distortion is the main issue of this section.

Quantization schemes can be classified as **(scalar) quantization** (*kvantisering*) and **vector quantization** (*vektorkvantisering*). Scalar quantization treats one signal value at a time while vector quantization processes a whole block of signal values. We are not interested in vector quantization in this course, but this form of quantization is now very much used in speech and image coding [4.9].

The most simple form of quantization is **uniform quantization** (*lineær kvantisering*). The signal range $[a_0;a_M]$ is divided into M **quantization intervals** (*kvantiseringsinterval*) of uniform size which are numbered n = 0, 1, ..., M−1. Ordinary **A/D-conversion** is an example of uniform quantization followed by a mapping of the quantization intervals into binary numbers. In Figure 4.5 we illustrate quantization into M = 8 levels.

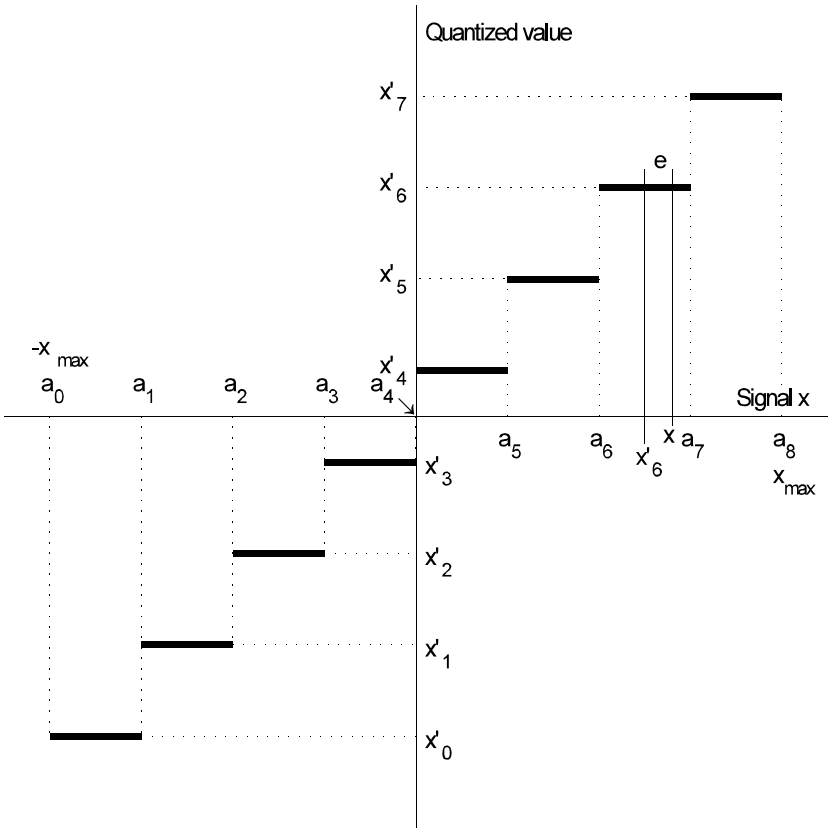

Figure 4.5
Example of uniform quantization with M = 8. The quantization is to be understood as a mapping from the x-axis to the quantized values shown at the y-axis. Quantization of a specific signal, x, in the interval 6 is illustrated including the quantization error, e.

If the signal x is in a particular interval $[a_n;a_{n+1}[$, only the number n is known for the **reconstruction** of the signal, and our problem is now to find a single value x_n', the **quantization level** or **quantized value** (*kvantiseret værdi*) that represents all signals in the

interval in the best possible way. As a criterion for the best possible value we select the value that minimizes the power of the **quantization noise** (*kvantiseringsstøj*), also known as the **quantization error** (*kvantiseringsfejlen*), $e = x - x'$. The power is sometimes known as the **mean-square (quantization) error** (*middelkvadratfejl*). We have already introduced this criterion around Eq. (4.1) where signal-to-noise ratio was discussed. We will now calculate the optimal x':

When the boundary values a_n for the intervals are given, the mean-square quantization error is

$$D = \sum_{n=0}^{M-1} \int_{a_n}^{a_{n+1}} (z - x_n')^2 f_X(z)\,dz \tag{4.3}$$

where $f_X(z)$ is the density function of X. D is minimized when

$$\frac{\partial D}{\partial x_n'} = \int_{a_n}^{a_{n+1}} 2(z - x_n')(-1) f_X(z)\,dz = 2x_n' \int_{a_n}^{a_{n+1}} f_X(z)\,dz - 2\int_{a_n}^{a_{n+1}} z f_X(z)\,dz = 0$$

which gives the optimal x':

$$x_n' = \frac{\int_{a_n}^{a_{n+1}} z f_X(z)\,dz}{\int_{a_n}^{a_{n+1}} f_X(z)\,dz} = \int_{a_n}^{a_{n+1}} z f_X(z \mid z \in [a_n; a_{n+1}[)\,dz \quad (\text{for } n = 0,\ldots,M-1) \tag{4.4}$$

This value, the mean in each interval, is known as the centroid of the interval. Note that this equation is also valid, if the signal range is not finite (i.e. $a_0 = -\infty$ and/or $a_M = \infty$) even though the outermost intervals do not have the same size as the uniform intervals. If the signal is uniformly distributed in the interval $[a_n, a_{n+1}]$ we have $f_X(z \mid z \in [a_n, a_{n+1}]) = 1/(a_{n+1} - a_n)$ and then

$$x_n' = \int_{a_n}^{a_{n+1}} z f_X(z \mid z \in [a_n; a_{n+1}[)\,dz = \frac{1}{a_{n+1} - a_n} \int_{a_n}^{a_{n+1}} z\,dz = \frac{a_n + a_{n+1}}{2} \tag{4.5}$$

which is the midpoint. If the quantization is fine enough, the signal will almost always be uniformly distributed in each interval, and only very rarely the quantized value x'_n is chosen different from the midpoint.

If the probability of being in the interval $[a_n;a_{n+1}[$ of length $\Delta = a_{n+1} - a_n$ is denoted p_n (p_n is the denominator in (4.4)), and we still assume a uniform distribution in the interval, we have $f_X(z) = p_n f_X(z \mid z \in [a_n;a_{n+1}[) = p_n/\Delta$, and the mean-square quantization error, D, becomes

$$\begin{aligned} D &= \sum_{n=0}^{M-1} \int_{a_n}^{a_{n+1}} (z - x_n')^2 f_X(z)\,dz = \sum_{n=0}^{M-1} \frac{p_n}{\Delta} \int_{a_n}^{a_{n+1}} (z - x_n')^2\,dz \\ &= \sum_{n=0}^{M-1} \frac{p_n}{\Delta} \int_{-\Delta/2}^{\Delta/2} e^2\,de = \sum_{n=0}^{M-1} p_n \frac{\Delta^2}{12} = \frac{\Delta^2}{12} \end{aligned} \tag{4.6}$$

The ratio between the signal power $E[X^2]$ and the power of the quantization noise may then be calculated assuming that the signal is uniformly distributed in a maximum signal range $[-x_{max};x_{max}]$ since $\Delta = 2x_{max}/M$:

$$\begin{aligned} SNR &= 10 \cdot \log \frac{E[X^2]}{D}\,[dB] = 10 \cdot \log \frac{x_{max}^2/3}{\Delta^2/12}\,[dB] \\ &= 10 \cdot \log \left(\frac{2x_{max}}{\Delta} \right)^2 [dB] = 20 \cdot \log M\,[dB] = 6v\,[dB] \end{aligned} \tag{4.7}$$

where the last expression is found in the common situation where M is represented with υ bits, i.e. $M = 2^{\upsilon}$. If the signal is not uniformly distributed in $[-x_{max};x_{max}]$ any value of SNR is possible. This is illustrated in Figure 4.6, where each dot represents the SNR at a given value of x.

In the figure, **overload** (*overstyring*) is also seen where the quantization error grows quickly since the signal is reconstructed as the last quantization value which is a little smaller than x_{max}. As it can be seen from the figure, the SNR for small signals is poor.

If x is uniformly distributed in only a part, u, of the maximum signal range, i.e. $x \in [-u \cdot x_{max}; u \cdot x_{max}]$, a calculation as (4.7) gives a reduced ($u < 1$) signal-to-noise-ratio

$$SNR = 10 \cdot \log \frac{(u \cdot x_{max})^2/3}{\Delta^2/12}\,[dB] = 20 \cdot \log M - 20 \cdot \log \frac{1}{u}\,[dB] = 6v - 20 \cdot \log \frac{1}{u}\,[dB]$$

which is shown in the figure as a solid line.

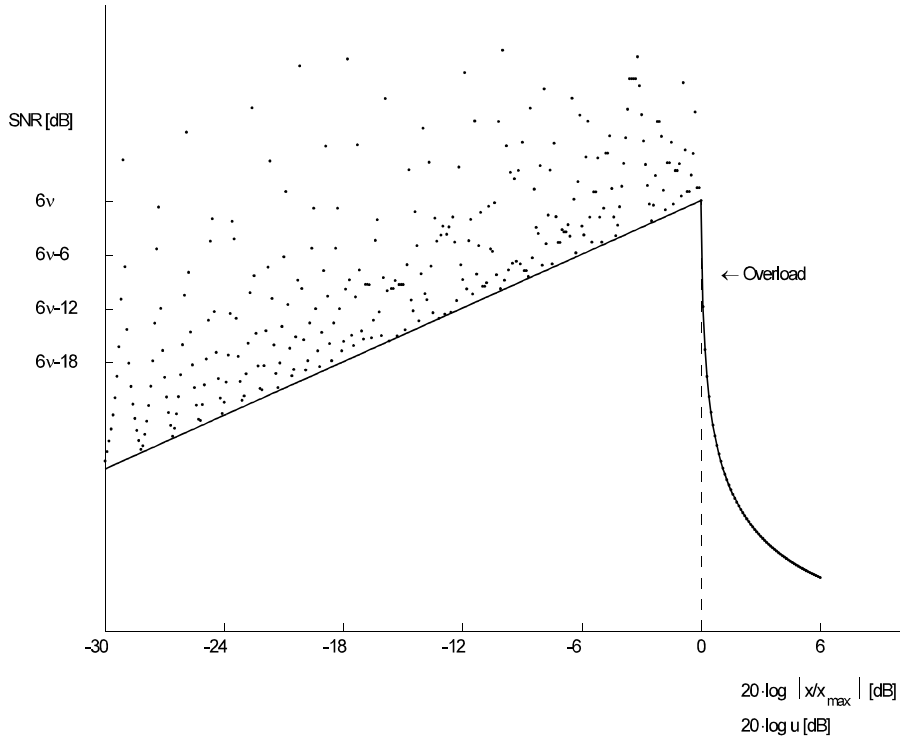

Figure 4.6
Signal-to-quantization-noise ratio for uniform quantization as function of the signal values (x/x_{max}, dots) and the used proportion, u, of the maximum signal range (solid line).

Generally, the phenomenon illustrated with the reduced signal range is an example of the quantization not being matched to the signal. This would require more knowledge of the signal, e.g. the distribution of signal amplitude such as the distribution for speech signals. With this knowledge it is possible to provide a better quantization for a signal, a **nonuniform quantization** (*ulineær kvantisering*) improving the signal-to-quantization-noise-ratio. The signal range $[a_0;a_M]$ is still divided into M quantization intervals, but now with arbitrary size. They are numbered n = 0, 1, ..., M − 1. The size of interval n is denoted Δ_n, and we shall see how to determine the set of decision values or boundary values $a_1,...,a_{M-1}$ separating the intervals. The optimal boundary value a_n for a given set of quantization values $\{x_n'\}$ is found from D in (4.3):

$$\frac{\partial D}{\partial a_n} = \frac{\partial}{\partial a_n}\left(\int_{a_{n-1}}^{a_n} (z-x_{n-1}')^2 f_X(z)\,dz + \int_{a_n}^{a_{n+1}} (z-x_n')^2 f_X(z)\,dz \right)$$
$$= (a_n-x_{n-1}')^2 f_X(a_n) - (a_n-x_n')^2 f_X(a_n) = 0$$

which gives the optimal solution:

$$a_n = \frac{x'_{n-1} + x'_n}{2} \quad (\text{for } n = 1,...,M-1) \tag{4.8}$$

The equations (4.4) and (4.8) are known as the **Lloyd-Max conditions**. As it may be seen, they cannot be calculated right away since each of them assumes knowledge of the other. However, it is possible to iterate towards a solution using the following algorithm:

1. Pick at random a set of quantization levels $x_0',...,x_{M-1}'$, e.g. distributed uniformly in the signal range $[a_0;a_M]$.
2. Use (4.8) to calculate a set of boundary values $a_1,...,a_{M-1}$ (a_0 and a_M are fixed).
3. Use (4.4) to calculate a new set of of quantization levels $x_0',...,x_{M-1}'$.
4. Calculate the mean-square error using (4.3).
5. Stop, if the reduction of the error from the previous step is very small, else go to Step 2.

Lloyd-Max quantization is mostly used if very few quantization intervals are found, but normally a finer quantization is used together with some postprocessing (in the coding subprocess) of the outcome of the quantization. The conditions require complete knowledge of the probability density function for X, and such knowledge is very seldom found in telecommunication applications since both the type of distribution and even such a characteristic as (instantaneous) power are unknown or only roughly known. These facts result in requirements for a robust quantization scheme. This could of course be found with uniform quantization but the required number of levels would be too large for the costs to be reasonable. Therefore it is more economical to apply the robust system described in the following.

Assuming that the signal X is uniformly distributed in each interval (of size Δ_n), the signal is reconstructed, x_n', as the midpoint of the interval, (4.5), and the contribution to the mean-square error from the interval becomes $\Delta_n^2/12$, (4.6). Denoting the probability of X being in the interval p_n the signal-to-quantization-noise ratio becomes

$$SNR = 10\cdot\log\frac{E[X^2]}{D} \approx 10\cdot\log\frac{\sum_{n=0}^{M-1} p_n x_n'^2}{\sum_{n=0}^{M-1} p_n \frac{\Delta_n^2}{12}} = 10\cdot\log 12 + 10\cdot\log\frac{\sum_{n=0}^{M-1} p_n x_n'^2}{\sum_{n=0}^{M-1} p_n \Delta_n^2} \text{ [dB]}$$

Now it is easily seen, that the SNR becomes almost independent of the signal distribution if one chooses $\Delta_n = k\cdot|x_n'|$, where k is a constant. This robust quantization is sometimes called **ideal quantization**. The resulting signal-to-noise ratio becomes

$$SNR \approx 10\cdot\log 12 - 10\cdot\log k^2 = 10.8 - 20\cdot\log k \text{ [dB]} \quad (4.9)$$

The nonuniform quantization just defined may be seen as a mapping of the variable x to another continuous variable y which is then uniformly quantized with N regions of equal size Δ_y. This mapping y = c(x) is called **compression** (*kompression*), and the method is often named **companding** (*kompandering*), which is merged from compression and the reverse **expansion** $x = c^{o-1}(y)$. Figure 4.7 shows such a compressor function c(x) and a rough nonuniform quantization to illustrate the principle.

For interval n we have $\Delta_y/\Delta_n = \Delta_y/(k\cdot|x_n'|)$, which for the differentiable function y = c(x) with the condition $x_{max} = y_{max}$ gives

$$\frac{dy}{dx} \approx \frac{\Delta_y}{k}\cdot\frac{1}{|x|} = \frac{2y_{max}/M}{k\cdot|x|} = \frac{2}{k\cdot M}\cdot\frac{1}{|x/x_{max}|}$$

which by integration gives the **ideal compressor characteristic** (*ideelle kompressorkarakteristik*) also named **logarithmic compressor characteristic** (*logaritmisk kompressorkarakteristik*):

$$\frac{y}{y_{max}} = \frac{sgn(x)}{y_{max}}\left(\frac{2}{Mk}\cdot x_{max}\cdot\ln\left|\frac{x}{x_{max}}\right| + c'\right) = sgn(x)\frac{1}{\ln c}\cdot\ln\left(c\left|\frac{x}{x_{max}}\right|\right)$$

since the condition $x_{max} = y_{max}$ gives $y_{max} = c'$ and k may be replaced by $k = (2\cdot\ln c)/M$. Figure 4.7 shows a coarse version of such a characteristic. As it may be seen, the characteristic does not pass (0,0), which means that minor modifications have to be done such as moving (μ-law, see Section 4.4) or connection with a straight line (A-law, also in Section 4.4). Calculation of SNR by (4.9) gives

$$SNR \approx 20 \cdot \log M + 10 \cdot \log \frac{3}{\ln^2 c} \approx 6 \cdot \nu + 10 \cdot \log \frac{3}{\ln^2 c} \qquad (4.10)$$

The last term is negative so the resulting SNR is reduced.

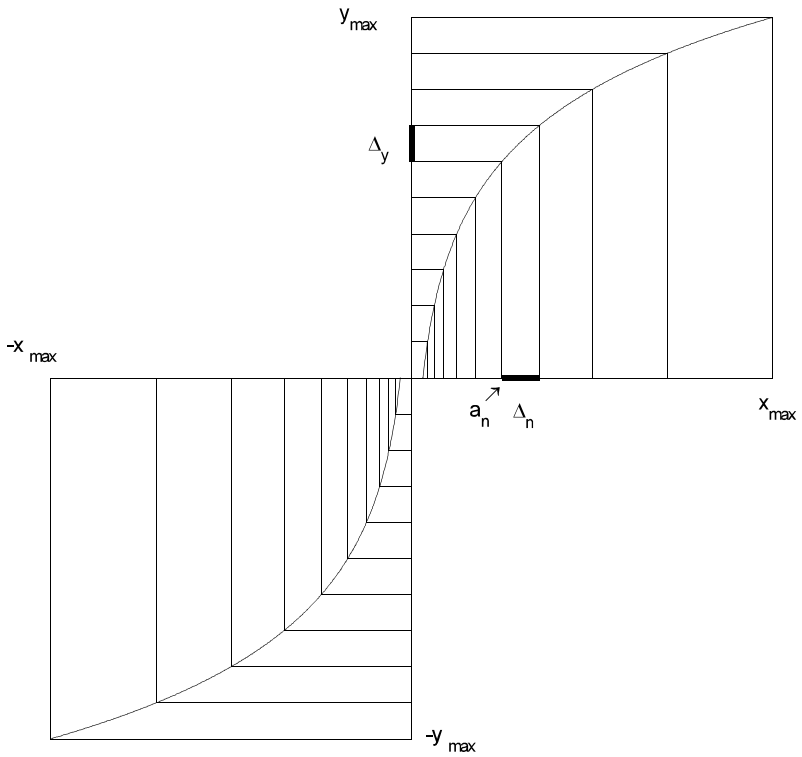

Figure 4.7
Ideal (or logarithmic) compressor function.

Normally, the compressor function is implemented as a uniform quantization with much more than M intervals, and then a digital conversion to the M intervals. A graph for the SNR for such a system is shown in Figure 4.10 and further description is given in the next section.

In Figures 4.8 and 4.9 we show the three quantization methods described, i.e. uniform, logarithmic, and Lloyd-Max, applied to a particular distribution for X. The comparison shows as expected better performance for Lloyd-Max, and it shows that the logarithmic quantization is robust to a change in the distribution in contrast to the uniform quantization.

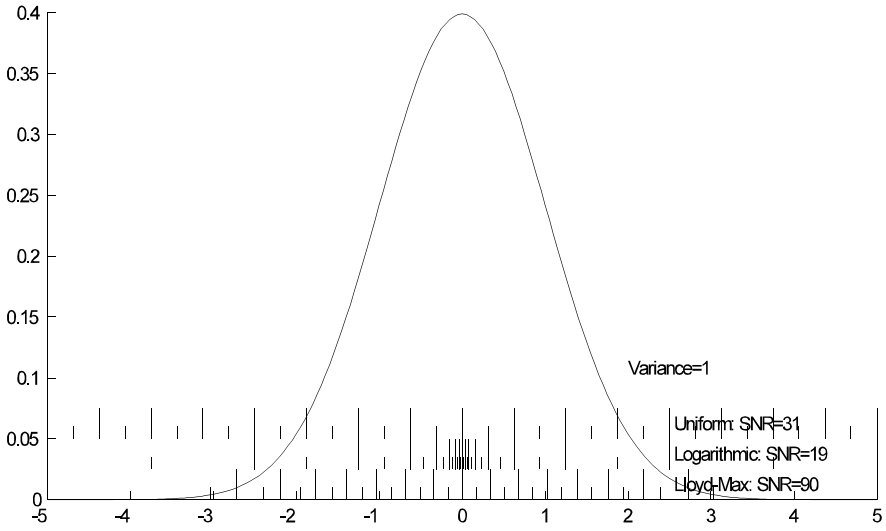

Figure 4.8
Comparison of three quantization methods for a normal distributed signal with variance 1. For each of the methods, the quantization intervals and values are given, and the resulting signal-to-noise ratios (in real values, not dB) are shown.

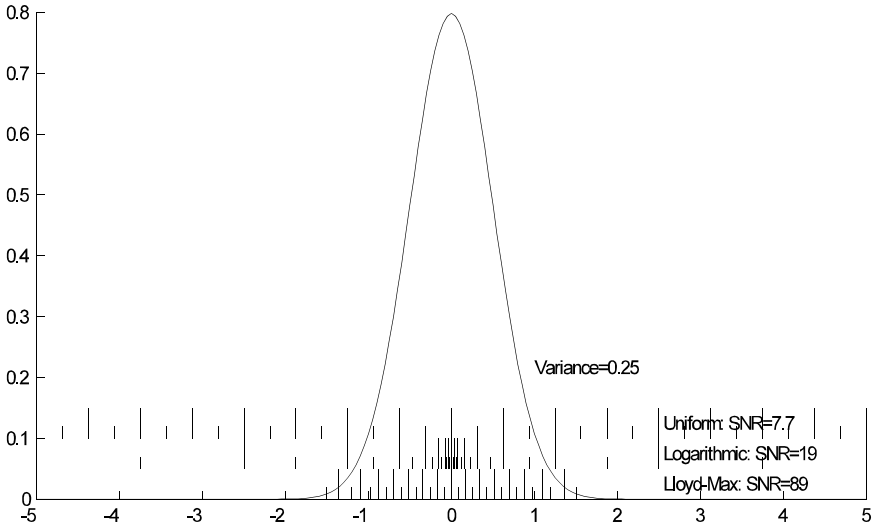

Figure 4.9
Comparison of three quantization methods for a normal distributed signal with variance 0.25. The results may be compared with Figure 4.8, and it is seen that the uniform quantization now is more unsuited to the signal, while the other two schemes give the same results.

4.4 Pulse Code Modulation, PCM

As we shall see below, a number of analog signals are transmitted or stored by conversion to digital signals employing sampling and quantizing. Traditionally the term for this operation has been **Pulse Code Modulation, PCM** (*pulskodemodulation, PCM*). Both uniform and nonuniform quantization are used.

4.4.1 Uniform PCM

Uniform quantization is mostly used for storage purposes. The main example is audio signals stored in CDs. The sampling frequency is 44.1 kHz and uniform quantization with 16 bits is used [4.10]. From (4.7) the signal-to-noise ratio becomes 96 dB, and even in faint passages the quantization noise is judged to be fairly low which has eliminated the need for nonuniform quantization and made the equipment a bit simpler. The capacity of the disc is still sufficient. For reproduction of music, the linearity of the quantizer is very important. Hard disk recorders use similar systems, but with higher sampling frequencies and maybe also more quantization bits. For transmission of high quality audio signals in the telecommunication network, similar solutions exist − often with the same characteristics as the CD.

Modern video production provides image frames which are PCM coded in the camera, and they are now brought to the receivers in digital form, **digital video** or **digital television (DVB)**, [4.4]. As described in Section 4.1.2, a video signal may be seen as a luma signal and two chrominance signals. If these are seen as time continuous signals, the luma signal is sampled with 13.5 MHz (i.e. 864 points, 720 active ones, per line for PAL, 576 active lines), [4.11], followed by uniform quantization with 8 bits for each pixel - since the luma component is gamma corrected the net result is a non-uniform quantization. A slightly modified version of the chrominance signals is sampled with 6.75 MHz (each), also followed by uniform quantization with 8 bits. Modern cameras produce pixel based signals directly, e.g. a standard resolution TV camera with 576 lines of 720 pixels. With 8 bits for each of the three signals this gives around 250 Mbit/s. The possibility to reduce the bandwidth for chrominance signals is often used, e.g. the **4:2:0** which uses half resolution in both directions for the chrominance signals. The name comes from a 4×2 rectangle of 8 luma samples for which two values of Cb and Cr describe the chrominance (in the first line) and none are found in the second line. **4:4:4** thus is the name of a scheme with full

resolution of all components. 4:2:0 reduces the bit rate to the half of the 4:4:4 systems, but still that data compression methods have to be applied to reduce the rate. Simple reduction methods will be described in Section 4.6 and more complicated methods (**MPEG**) are described in the courses on data compression and digital video, [4.9, 4.4].

4.4.2 Nonuniform PCM

The main example of nonuniform PCM is coding of telephone channels. In Section 4.1.1 we argued that the frequency range of such a signal is 300 Hz - 3400 Hz. We shall describe the recommendations from ITU-T [4.3, G.711, G.712]. In this standard, the sampling frequency is chosen to be 8000 Hz (±0.4 Hz) since the lowpass filters employed before sampling and for reconstruction cannot be made ideal. The dynamic signal range for speech is very large and the number of bits available for transmission of each sample is restricted. Therefore nonuniform quantization is employed with 256 intervals (8 bits). For historical reasons, ITU-T was forced to standardize two nonuniform quantization schemes.

In North America the **μ-law nonlinearity** (*μ-lov karakteristik*) is applied:

$$\frac{y}{y_{max}} = \operatorname{sgn}(x)\frac{\ln\left(1+\mu\left|\frac{x}{x_{max}}\right|\right)}{\ln(1+\mu)} \qquad \mu = 255$$

In the rest of the world (especially Europe) the **A-law nonlinearity** (*A-lov karakteristik*) is used:

$$\left.\begin{aligned} &\frac{y}{y_{max}} = \frac{A}{1+\ln A}\cdot\frac{x}{x_{max}} && \left|\frac{x}{x_{max}}\right| < \frac{1}{A} \\ &\frac{y}{y_{max}} = \operatorname{sgn}(x)\frac{1+\ln\left(A\left|\frac{x}{x_{max}}\right|\right)}{1+\ln A} && 1 \geq \left|\frac{x}{x_{max}}\right| \geq \frac{1}{A} \end{aligned}\right\} \quad A = 87.6$$

The transition between the straight line segment and the logarithmic segment is chosen such that the straight line is the tangent of the logarithmic segment. Using (4.10) we have $c = eA$ and SNR $\approx 6\cdot\nu - 10$ dB = 38 dB. In practice, the A-law is implemented as uniform quantization with 12 bits plus sign (i.e. [−4095; 4095]) followed by a piecewise linear approximation to the A-law. The approximation is divided into a number of segments (7

for positive signals and 7 for the negative part) each having uniform quantization (normally 16 intervals, 32 in segment 1). A similar digital implementation of the μ-law exists with 15 segments. For the A-law, the segments, the endpoints for these, number of intervals and their size are shown in Table 4.1.

The coding to 8 bits (numbered from left as 1 - 8) is shown in the third column and the code is composed of the sign (bit 1), the segment number (0 - 7, 3 bits (2-4), segment 1 has double size, so both 0 and 1 are applied, 00y in the table), and the number of the interval with the uniform coded segment (bits 5 - 8, xxxx in table). Following the coding, the **even numbered bits are inverted** before transmission. This is done to avoid a long run of zeros if the signal is 0. The maximum voltage quantized (corresponding to 4096) is 1.5725 V which is the power 3.14 dBm in the nominal input impedance 600 Ω (dBm is a very much used logarithmic unit for power, and it means the power, P, in relation to 1 mW, i.e. P[dBm] = 10·log(P/1 mW)).

The reconstruction of analog signals quantized into one interval is done as the midpoint, i.e. with the notation of (4.5):

$$x_n' = \frac{a_n + a_{n+1}}{2} = a_n + \frac{\Delta_n}{2}, \quad n = 0, \ldots, 255$$

This is also the reason for specifying 12 bits which are not needed for the quantization since the minimum $\Delta_n = 2$, but the quantized values for segment 1 are odd such that all 12 bits are needed.

Segment	Endpoint (a_n)	Coding	# of intervals	Size of interval
7	4096 2048	1 111 xxxx	16	128
6	1024	1 110 xxxx	16	64
5	512	1 101 xxxx	16	32
4	256	1 100 xxxx	16	16
3	128	1 011 xxxx	16	8
2	64	1 010 xxxx	16	4
1		1 00y xxxx	32	2
1		0 00y xxxx	32	2
2	−64	0 010 xxxx	16	4
3	−128	0 011 xxxx	16	8
4	−256	0 100 xxxx	16	16
5	−512	0 101 xxxx	16	32
6	−1024	0 110 xxxx	16	64
7	−2048 −4096	0 111 xxxx	16	128

Table 4.1
A-law coding (exclusive inverting) from [4.3, G.711].

Example 4.1 - Coding and reconstruction of analog signal

Assume that we sample 0.25 V. The uniformly quantized value becomes

$$x = \left\lfloor \frac{0.25\,\mathrm{V}}{1.5725\,\mathrm{V}} \cdot 4096 \right\rfloor = 651$$

Note that the rounding is done towards 0 since that interval endpoint is assumed to belong to the interval. Since 651 is in segment 5 (101) and 651 = 512 + 4·32 + 11, the decision value a_n = 512 + 4·32. Thus xxxx = 0100 and the coding is 1 101 0100, which after inverting even bits becomes 10000001. The signal is reconstructed as

$$x_n' = \left(512 + \frac{4+5}{2} \cdot 32\right) \cdot \frac{1.5725}{4096}\,\mathrm{V} = 656 \cdot \frac{1.5725}{4096}\,\mathrm{V} = 0.2518\,\mathrm{V}$$

and the quantization error is (0.25 − 0.2518) V = −1.8 mV.

As an example of a negative voltage we use −0.017 V which uniformly quantized gives

$$x = -\left\lfloor \frac{0.017\,\mathrm{V}}{1.5725\,\mathrm{V}} \cdot 4096 \right\rfloor = -44$$

again rounded towards 0. Since −44 is in segment 1 and 44 = (16+6)·2, we get yxxxx = 10110, and the coding 0 001 0110 which after inverting even bits becomes 01000011.

The signal is reconstructed as

$$x_n' = -\left(0 + \frac{22+23}{2} \cdot 2\right) \cdot \frac{1.5725}{4096}\,\mathrm{V} = -45 \cdot \frac{1.5725}{4096}\,\mathrm{V} = -0.01728\,\mathrm{V}$$

and the quantization error is now (−0.017 − (−0.01728)) V = 0.28 mV.

It is easily seen that k from (4.9) for this system 2^{-4} in the lower end of segments 2 to 7 (e.g. 128/2048 in segment 7) and 2^{-5} in the higher end (e.g. 128/ 4096). Applying that in (4.9) shows that the signal-to-noise-ratio in these intervals varies between 35 dB and 41 dB. This is shown in Figure 4.10. The smallest signal where SNR still is at least 35 dB is found as the middle of segment 1, i.e. signal power (3.14 − 7·6) dBm = −38.9 dBm. Thus the dynamic range for maintaining SNR = 35 dB is 42 dB.

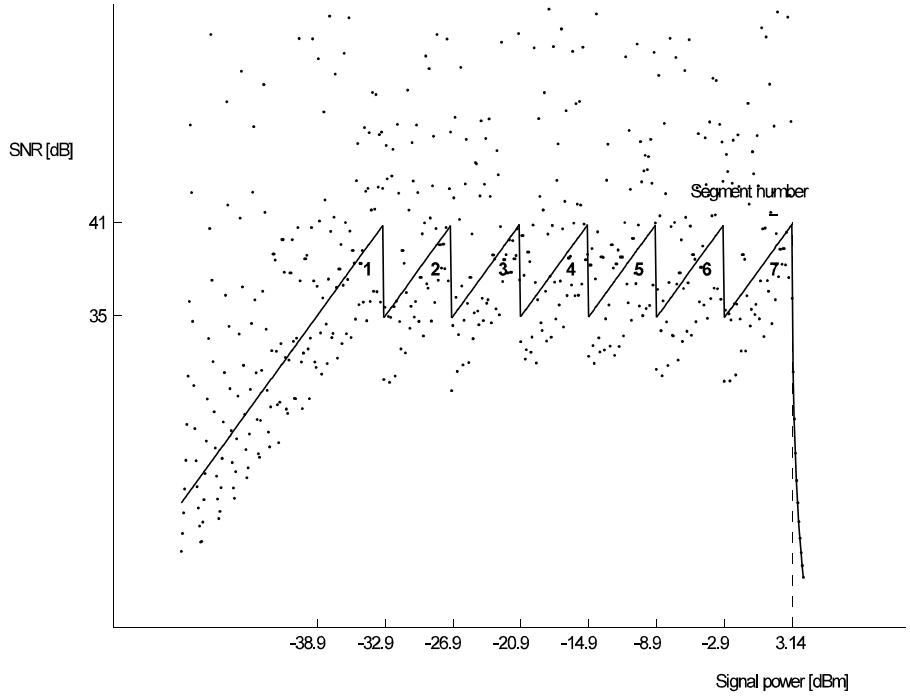

Figure 4.10
Signal-to-quantization-noise ratio for the nonuniform quantization from Table 4.1 as functions of a constant voltage (dots) and the average over each quantization interval (solid line).

The difference in performance between the μ-law and the A-law is negligible. In addition to the sampling and quantization described, ITU-T also specifies requirements for filters used and delay of the system. This is done in ITU-T recommendation [4.3, G.712], which describes requirements for the analog channel through a PCM coder followed by a decoder. The most important problem is specification of the lowpass filters that are to be used at the input before sampling in order to keep the aliasing small, and to be used for the reconstruction at the output, cf. Figures 4.3 and 4.4. It is difficult to implement lowpass filters and

therefore the sampling frequency is more than 2·3400 Hz, namely 8000 Hz. It is then possible to keep the aliasing at an acceptable level by a transfer function whose magnitude in the range 3400 - 4600 Hz is shown in Figure 4.11. The transfer function shown is a result of both lowpass filters.

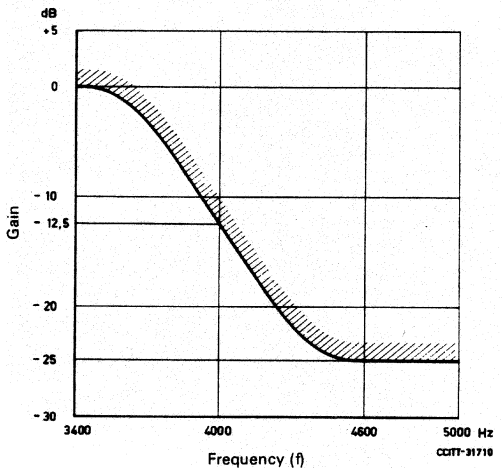

Figure 4.11
Gain outside the pass frequency range 300 - 3400 Hz for
an analog channel through a PCM-system, [4.3, G.712].

ITU-T has also standardised a system for audio signals with bandwidth 15 kHz, which are sampled with 32 kHz and compressed from 14 bits for the uniform quantization to 11 bits [4.3, J.41]. The compression may be done with the A-law but with 2·6 segments. This gives $k = 2^{-6} - 2^{-7}$ and (4.9) shows that the signal-to-noise-ratio varies between 47 dB and 53 dB over a large signal range. When overhead for frame synchronization etc. is added the bit rate becomes 384 kbit/s = 6·64 kbit/s. In the same standard, another compression method is described, where the coding is determined by 32 samples as a block. This coding is slightly more powerful (10 bits/sample), but gives a small delay, which is why it is named **Near Instantaneous Companding**. **NICAM** (Near Instantaneous Companding Audio Multiplexed) for TV-stereo sound in analog TV uses this method (728 kbit/s for two channels incl. overhead, see [4.6]). If the transmission capacity is smaller, [4.3, J.42] for 7 kHz audio channels is found. These are sampled with 16 kHz and compressed as in J.41. Two such channels add up to 384 kbit/s.

4.5 Linear estimation, prediction, and interpolation

As a theoretical basis for the methods of waveform coding in the following section and for other applications, we shall consider stochastic signals with known power spectra (and autocorrelation functions). First we shall present a very general principle of **linear estimation** of real variables. This principle will then be applied to filtering of signals in noise, **interpolation**, and **extrapolation** (**prediction** (*prædiktion*)) of signals. Another application - equalization - will be shown in Section 5.6.

Estimation is a very important operation in communication systems. Here we treat only (linear) estimation of variables given some quality requirement, but in many other applications, parameters of a stochastic process are to be estimated instead of the variables as such. We give no theoretical treatment of this estimation problem in the present course, but in Chapter 5 we show examples, e.g. how to determine the optimal sampling time for a received signal.

We begin our treatment of linear estimation with the following simple example.

Example 4.2 - Linear prediction based on the previous value

We observe a stationary process, X, with autocorrelation $R_X(k)$, zero mean and variance σ_x^2. We want to estimate the variable X_n before it can be observed, but using the observation of the previous value, x_{n-1}. Consider the estimator

$$\hat{x}_n = h_1 x_{n-1}$$

The error is $e = x_n - \hat{x}_n$. The variance of the error is (since $E[x_n] = 0$)

$$\sigma_e^2 = E[(x_n - \hat{x}_n)^2] = E[(x_n - h_1 x_{n-1})^2] = (1 + h_1^2) R_X(0) - 2h_1 R_X(1)$$

The optimal value of h_1 which minimizes the variance of the error may be found from

$$\frac{d\sigma_e^2}{dh_1} = 2h_1 R_X(0) - 2R_X(1) = 0$$

which gives the optimal value $h_1 = R_X(1)/R_X(0)$ and

$$\sigma_e^2 = R_X(0) - R_X(1)^2/R_X(0).$$

In the next section we shall generalize this approach to several observations.

4.5.1 Linear estimation with minimal mean square error

Assume that we have made m observations $y_1, y_2, \ldots, y_m$ of the stochastic variables $Y_1, Y_2, \ldots, Y_m$, and that we want to estimate the variable X, which cannot be observed, i.e. we want some function f - an **estimator** - such that $\hat{x} = f(y_1,y_2,\ldots,y_m)$ gives our best guess for X. If we restrict the function f to be linear, we form the **linear estimator**

$$\hat{x} = \sum_{k=1}^{m} h_k y_k \tag{4.11}$$

which is also a stochastic variable. An outcome of $\hat{x}$ is called an **estimate** (*estimat*) of the outcome of X. The number, m, of observations in the estimator is known as the **order** (*orden*) of the estimator.

We would like to determine the estimator which minimizes the mean square error $E[(x-\hat{x})^2]$. This may not always be the relevant criterion, but it is the most important one, and, as it will be seen, treatable. First we prove an important result about such optimal estimators, known as the projection lemma

$$E[(x-\hat{x})y_j] = 0 \quad \text{for all } j \tag{4.12}$$

Proof: For the optimal point we have

$$\frac{\partial}{\partial h_j} E[(x-\hat{x})^2] = 2E[(x-\hat{x})\cdot\frac{\partial(x-\hat{x})}{\partial h_j}] = -2E[(x-\hat{x})y_j] = 0$$

Since $\hat{x}$ is a linear combination of the observations y_k, we may apply (4.12) to obtain

$$E[\hat{x}(x-\hat{x})] = 0 \quad \text{or} \quad E[x\hat{x}] = E[\hat{x}^2] \tag{4.13}$$

It is intuitively reasonable that if the error were correlated with one of the observations, it would be possible to improve the estimator. The projection lemma may be given a geometric interpretation in the following way: Let each variable y_j be represented as a vector $\mathbf{y}_j$, and introduce the scalar product: $\mathbf{y}_j \cdot \mathbf{y}_k = E[y_j \cdot y_k]$. In particular $|\mathbf{y}_j|^2 = \sigma_y^2$. The

error should be orthogonal to the space spanned by the observations, i.e. the estimate is obtained by projecting **x** on the observed space. This is shown in Figure 4.12 for an estimator of order 2.

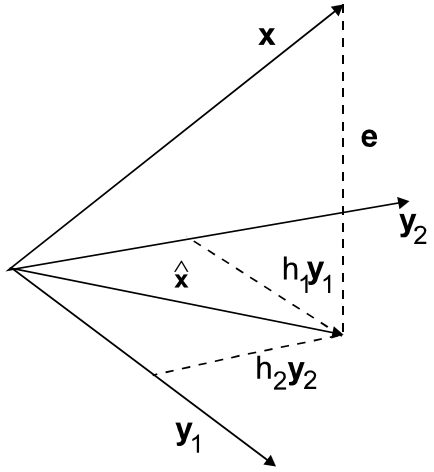

Figure 4.12
Geometric illustration of the projection lemma, (4.12) and (4.13).

Inserting (4.11) into (4.12) we obtain m equations in the coefficients h_k :

$$\sum_{k=1}^{m} E[Y_j Y_k] h_k = E[XY_j], \quad 1 \le j \le m \tag{4.14}$$

This system of m equations is called the **Wiener-Hopf equation** (*Wiener-Hopf ligningen*).

Using (4.13) we may find the variance of the error $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$ of the best estimate as

$$\sigma_e^2 = E[(x-\hat{x})^2] = E[(x-\hat{x})(x-\hat{x})] = E[x(x-\hat{x})] = \sigma_x^2 - E[x\hat{x}] \Rightarrow$$

$$\sigma_e^2 = \sigma_x^2 - \sigma_{\hat{x}}^2 \tag{4.15}$$

Here we have assumed that the mean of X is zero which implies that $E[\hat{x}] = 0$. The variance of the estimate is also obtained from (4.13)

$$\sigma_{\hat{x}}^2 = E[\hat{x}^2] = E[x\hat{x}] = \sum_{j=1}^{m} h_j E[XY_j] \tag{4.16}$$

It is easiest to remember this result by noticing that when **h** satisfies (4.14), the variance of the estimate is the scalar product of **h** and the righthand side of this equation.

This expression may be used to indicate the uncertainty of the estimate, and it is also useful for evaluating the potential advantage of using an estimate derived from a larger Wiener-Hopf equation, or perhaps the advantage of this approach compared to a simpler type of signal processing. It is a common experience that it rarely pays off to use a very complicated estimate for problems of this type.

Example 4.3 - Linear interpolation

Assume that the outcome of a stochastic, stationary signal, Y, is observed with the exception of a missing value at time 0, y_0. We shall find the best linear interpolation from the observed surrounding values:

$$\hat{y}_0 = \sum_{k\neq 0} h_k y_k$$

From (4.14) we get

$$\sum_{k\neq 0} E[Y_j Y_k] h_k = E[Y_0 Y_j], \quad j\neq 0$$

which due to the assumption of stationarity may be expressed by the autocorrelation function

$$\sum_{k\neq 0} R_Y(j-k) h_k = R_Y(-j), \quad j\neq 0$$

or since the autocorrelation is an even function

$$\sum_{k\neq 0} R_Y(|j-k|) h_k = R_Y(|j|), \quad j\neq 0$$

For a signal with $R_Y(0) = 4$, $R_Y(1) = 2$ and $R_Y(2) = -1$, we may create an interpolator of order 2 between y_{-1} and y_1:

$$\begin{bmatrix} R_Y(0) & R_Y(2) \\ R_Y(2) & R_Y(0) \end{bmatrix} \begin{bmatrix} h_{-1} \\ h_1 \end{bmatrix} = \begin{bmatrix} R_Y(1) \\ R_Y(1) \end{bmatrix}$$

$$\begin{bmatrix} 4 & -1 \\ -1 & 4 \end{bmatrix} \begin{bmatrix} h_{-1} \\ h_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

Thus $h_{-1} = h_1 = 2/3$. The estimate becomes

$$\hat{y}_0 = \frac{2}{3}(y_{-1} + y_1)$$

The variance of the estimate may be found from (4.16)

$$\sigma_{\hat{y}}^2 = \frac{2}{3} \cdot 2 + \frac{2}{3} \cdot 2 = \frac{8}{3}$$

and the variance of the error is then found from (4.15)

$$\sigma_e^2 = 4 - \frac{8}{3} = \frac{4}{3}.$$

The result could be compared with the variance of the error for the obvious interpolator

$$\hat{y}_0 = \frac{1}{2}y_{-1} + \frac{1}{2}y_1.$$

for which the variance of the error is found as

$$\begin{aligned}\sigma_e^2 &= E[(Y - \hat{Y})^2] = E[(y_0 - \hat{y}_0)^2] = E[(y_0 - \tfrac{1}{2}y_{-1} - \tfrac{1}{2}y_1)^2] \\ &= E[y_0^2] + \tfrac{1}{4}E[y_{-1}^2] + \tfrac{1}{4}E[y_1^2] - E[y_0 y_{-1}] - E[y_0 y_1] + \tfrac{1}{2}E[y_{-1} y_1] \\ &= \tfrac{3}{2}R_Y(0) - 2R_Y(1) + \tfrac{1}{2}R_Y(2) = \tfrac{3}{2}\end{aligned}$$

where we have used the stationarity several times. The resulting variance is of course more than that for the optimal interpolation.

We may reduce the error by using more signal values in the interpolation. Using 2 values on both side of y_0 and taking $R_Y(3) = -2$ and $R_Y(4) = -3/2$, we get

$$\begin{bmatrix} 4 & 2 & -2 & -3/2 \\ 2 & 4 & -1 & -2 \\ -2 & -1 & 4 & 2 \\ -3/2 & -2 & 2 & 4 \end{bmatrix} \begin{bmatrix} h_{-2} \\ h_{-1} \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 2 \\ -1 \end{bmatrix}$$

Thus

$$h_2 = h_{-2} = -\frac{2}{5},\ h_1 = h_{-1} = \frac{2}{3},$$
$$\sigma_{\hat{y}}^2 = \left(-\frac{2}{5}\right)\cdot(-1-1) + \frac{2}{3}\cdot(2+2) = 3\frac{7}{15},\ \sigma_e^2 = 4 - 3\frac{7}{15} = \frac{8}{15}.$$

and we obtain a clearly better result compared to the first estimate. The estimator is

$$\hat{y}_0 = -\frac{2}{5}(y_{-2} + y_2) + \frac{2}{3}(y_{-1} + y_1)$$

The fact that h_1 has the same value in the two estimates is a coincidence.

4.5.2 Optimal filtering of stationary signals

Although the Wiener-Hopf equation (4.14) is valid for general stochastic variables with some relation, we restricted ourselves in the two examples to consider observations taken from a stochastic process, so we shall take a closer look at this case. The previous Example 4.3 and the following Examples 4.4 and 4.5 are very important special cases of applications of the following Wiener-Hopf equation for stationary signals, (4.18).

If we observe a stationary process, Y, with autocorrelation $R_Y(k)$, and we know the cross correlation $R_{XY}(k)$ between the observed process and another process, X, which we wish to estimate, then we may write the linear estimate as

$$\hat{x}_n = \sum_{k\in K} h_k y_{n-k} \tag{4.17}$$

where K is some index set with m elements (m is known as the order). The index set is chosen appropriately for the application, e.g. $K = \{1\}$ in Example 4.2 and $K = \{-2,-1,1,2\}$ in the last part of Example 4.3. The coefficients in the estimator may be obtained from the m equations, which result from using stationarity to replace the mean values in (4.14) with correlations

$$\sum_{k\in K} R_Y(j-k)h_k = R_{XY}(j) \qquad j \in K = \{j_1, j_2, \ldots, j_m\} \tag{4.18}$$

The lefthand side is rather obvious from (4.14), and the righthand side is found as $E[x_n y_{n-j}] = R_{XY}(n-(n-j)) = R_{XY}(j)$. (4.18) may be written as a matrix equation:

$$\begin{bmatrix} R_Y(0) & R_Y(j_1-j_2) & \dots & R_Y(j_1-j_m) \\ R_Y(j_2-j_1) & R_Y(0) & \dots & R_Y(j_2-j_m) \\ - & - & \dots & - \\ R_Y(j_m-j_1) & R_Y(j_m-j_2) & \dots & R_Y(0) \end{bmatrix} \begin{bmatrix} h_{j_1} \\ h_{j_2} \\ - \\ h_{j_m} \end{bmatrix} = \begin{bmatrix} R_{XY}(j_1) \\ R_{XY}(j_2) \\ - \\ R_{XY}(j_m) \end{bmatrix}$$

This system of m equations is called the **Wiener-Hopf equation** for stationary signals which is the most common assumption. If observations arbitrarily far back may be used, we may let the set K in (4.18) be infinite towards ∞. If a delay is allowed in the estimate, it is possible to include some negative values of j. In the case of off-line signal processing the set K may contain both ±∞. In the last case one can apply the Fourier transform to the Wiener-Hopf equation to obtain

$$H(\omega)S_Y(\omega) = S_{XY}(\omega) \tag{4.19}$$

which is exactly the relation between input and output if the output were X rather than $\hat{X}$, compare (3.17). Calculating the variance of the estimate by (4.16) and inserting into (4.15) for calculation of the variance of the estimation error we get

$$\sigma_e^2 = \sigma_x^2 - \sigma_{\hat{x}}^2 = R_X(0) - \sum_{k\in K} h_k R_{XY}(k) \tag{4.20}$$

with the usual assumption of zero mean for X.

Example 4.4 - Filtering of signals plus noise

The observations are $y_i = x_i + n_i$, $y_{i-1} = x_{i-1} + n_{i-1}$, . . ., $y_{i-(m-1)} = x_{i-(m-1)} + n_{i-(m-1)}$, where x_j has zero mean, and n_j is white noise with zero mean and variance σ_n^2. Further we assume that the noise is uncorrelated with the signal x_j. One wishes to estimate x_i by a linear filter (estimator). Since n_j and x_j are uncorrelated, the correlations become

$$R_Y(k) = \begin{cases} E[(x_i+n_i)(x_i+n_i)] = E[x_i^2] + E[n_i^2] + 2E[x_i n_i] = R_X(0) + \sigma_n^2 + 0, & k = 0 \\ E[(x_{i+k}+n_{i+k})(x_i+n_i)] = R_X(k) + R_N(k) + 0 + 0 = R_X(k), & \text{otherwise} \end{cases}$$

$$R_{XY}(k) = E[x_{i+k}(x_i+n_i)] = E[x_{i+k}x_i] + E[x_{i+k}n_i] = R_X(k) + 0$$

The coefficients of the filter giving the best estimate of x_k are determined by solving the system of equations (4.18) ($K = \{0,1,...,m-1\}$), where we use the fact that the autocorrelation function is even

$$\begin{bmatrix} R_X(0)+\sigma_n^2 & R_X(1) & \dots & R_X(m-1) \\ R_X(1) & R_X(0)+\sigma_n^2 & \dots & R_X(m-2) \\ - & - & \dots & - \\ R_X(m-1) & R_X(m-2) & \dots & R_X(0)+\sigma_n^2 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ - \\ h_{m-1} \end{bmatrix} = \begin{bmatrix} R_X(0) \\ R_X(1) \\ - \\ R_X(m-1) \end{bmatrix}$$

We shall now in an example investigate the behavior of the estimation error for different orders m of the system. Let the autocorrelation function be

$$R_X(k) = \begin{cases} 1 & \text{for } k = 0 \\ \tfrac{1}{2} & \text{for } k = \pm 1 \\ 0 & \text{otherwise} \end{cases}$$

and let $\sigma_n^2 = \tfrac{1}{4}$ in the rest of the example. Thus for m = 2 we get

$$\begin{bmatrix} \frac{5}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{5}{4} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

with solution $h_0 = 16/21$, $h_1 = 2/21$. The estimate

$$\hat{x}_i = \frac{16}{21} y_i + \frac{2}{21} y_{i-1}$$

has variance $\sigma_{\hat{x}}^2 = 16/21 + 1/21 = 17/21$ and error $\sigma_e^2 = 1 - 17/21 = 4/21 = 0.190$.

Allowing a delay of one sample period, y_{i+1} may be used for estimating x_i. With y_{i+1}, y_i, and y_{i-1} we have the index set $K = \{-1,0,1\}$ and we get

$$\begin{bmatrix} R_X(0)+\sigma_n^2 & R_X(1) & R_X(2) \\ R_X(1) & R_X(0)+\sigma_n^2 & R_X(1) \\ R_X(2) & R_X(1) & R_X(0)+\sigma_n^2 \end{bmatrix} \begin{bmatrix} h_{-1} \\ h_0 \\ h_1 \end{bmatrix} = \begin{bmatrix} R_X(1) \\ R_X(0) \\ R_X(1) \end{bmatrix}$$

$$\begin{bmatrix} \frac{5}{4} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{5}{4} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{5}{4} \end{bmatrix} \begin{bmatrix} h_{-1} \\ h_0 \\ h_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 1 \\ \frac{1}{2} \end{bmatrix}$$

Thus $h_0 = 12/17$ and $h_1 = h_{-1} = 2/17$. The estimate

$$\hat{x}_i = \frac{12}{17} y_i + \frac{2}{17}(y_{i+1} + y_{i-1})$$

has variance

$$\sigma^2_{\hat{x}} = \frac{12}{17} + \frac{2}{17} = \frac{14}{17}$$

and the error variance $\sigma_e^2 = 1 - 14/17 = 3/17 = 0.176$ is slightly better than before.

If the estimation may be based on all signal values up to time i, one gets an infinite system of equations

$$\begin{bmatrix} \frac{5}{4} & \frac{1}{2} & 0 & \dots \\ \frac{1}{2} & \frac{5}{4} & \frac{1}{2} & \dots \\ 0 & \frac{1}{2} & \frac{5}{4} & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \dots \\ \dots \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \\ \dots \\ \dots \end{bmatrix}$$

For $j \geq 1$ we see from the third row and onwards that

$$\frac{1}{2} h_j + \frac{5}{4} h_{j+1} + \frac{1}{2} h_{j+2} = 0.$$

This difference equation may be solved as $h_j = a_1 (-1/2)^j + a_2 (-2)^j$, but since the filter coefficients cannot increase without limit, $a_2 = 0$. Thus we take $h_2 = -1/2\, h_1$ and solve the first two equations to obtain $h_0 = 3/4$ and $h_1 = 1/8$. The variance of the estimate becomes $\sigma^2_{\hat{x}} = 3/4 + 1/16 = 13/16$ and the error $\sigma_e^2 = 1 - 13/16 = 3/16 = 0.186$ which is actually slightly worse than the situation where we were allowed to use the following value. By writing the estimates $\hat{x}_i$ and $\hat{x}_{i-1}$

$$\hat{x}_i = \frac{3}{4}y_i + \frac{1}{8}y_{i-1} - \frac{1}{16}y_{i-2} \cdots$$

$$\hat{x}_{i-1} = \frac{3}{4}y_{i-1} + \frac{1}{8}y_{i-2} - \frac{1}{16}y_{i-3} \cdots$$

we see that in the expression for $\hat{x}_i + \frac{1}{2}\hat{x}_{i-1}$ the terms for $i-2$ and earlier cancel. Thus the filter may be expressed by the difference equation

$$\hat{x}_i = -\frac{1}{2}\hat{x}_{i-1} + \frac{3}{4}y_i + \frac{1}{2}y_{i-1}$$

and it has transfer function (cf. (2.16)):

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

If finally we allow the estimate to depend on the entire received signal, it is easiest to get the solution from (4.19):

$$H(\omega) = \frac{S_{XY}}{S_Y} = \frac{S_X(\omega) + 0}{S_X(\omega) + \sigma_n^2} = \frac{1 + \cos\omega T}{\frac{5}{4} + \cos\omega T}$$

The filter coefficients may be found from the transform that was used from Equation (3.19) to (3.20). With p = 1 and q = ½ both expressions have to scaled with ½, and one then finds $h_0 = 2/3$, $h_k = 1/6(-1/2)^{|k|-1}$ for $|k| \geq 1$.

The variance of the estimate becomes $\sigma_{\hat{x}}^2 = 2/3 + 1/6 = 5/6$, and the smallest possible error variance is thus $\sigma_e^2 = 1 - 5/6 = 1/6 = 0.167$.

In Section 5.6 we shall use a similar method to estimate *a* subject to a linear operation, *x = f(a)*, and noise.

Example 4.5 - Linear prediction (Yule-Walker equations)

The observations from a stationary process Y are $y_{n-1}, y_{n-2}, ..., y_{n-m}$ and one wishes to predict y_n with the linear predictor (4.17):

$$\hat{x}_n = \hat{y}_n = \sum_{k=1}^{m} h_k y_{n-k}$$

From (4.18) we get, using $R_{XY}(j) = E[y_n y_{n-j}] = R_Y(j)$

$$\begin{bmatrix} R_Y(0) & R_Y(1) & ... & R_Y(m-1) \\ R_Y(1) & R_Y(0) & ... & R_Y(m-2) \\ - & - & ... & - \\ R_Y(m-1) & R_Y(m-2) & ... & R_Y(0) \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ . \\ h_m \end{bmatrix} = \begin{bmatrix} R_Y(1) \\ R_Y(2) \\ . \\ R_Y(m) \end{bmatrix} \qquad (4.21)$$

This special case of (4.14) and (4.18) is known as the **Yule-Walker equations** (*Yule-Walker ligningerne*) or the **normal equations** (*normalligningerne*).

As an example, consider a signal created by a first order linear system as described in Eq. (3.18) with p = 0, i.e. $R_Y(k) = (-q)^{|k|}/(1-q^2)$. After multiplication with $1-q^2$ the equations become

$$\begin{bmatrix} 1 & -q & ... & (-q)^{m-1} \\ -q & 1 & ... & (-q)^{m-2} \\ - & - & ... & - \\ (-q)^{m-1} & (-q)^{m-2} & ... & 1 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ . \\ h_m \end{bmatrix} = \begin{bmatrix} -q \\ q^2 \\ . \\ (-q)^m \end{bmatrix}$$

and the solution is always $h_1 = -q$, $h_k = 0$ for $k > 1$.

Thus the prediction is

$$\hat{y}_n = -q \cdot y_{n-1}.$$

The error variance is given by (4.20)

$$\sigma_e^2 = R_Y(0) - \sum_{k=1}^{m} h_k R_Y(k) = \frac{1}{1-q^2} - (-q)\frac{-q}{1-q^2} = 1.$$

This result is consistent with the result from Example 4.2:

$$\sigma_e^2 = R_Y(0) - R_Y(1)^2/R_Y(0).$$

Note that for $|q| \neq 0$, which implies $R_Y(1) \neq 0$, the error variance σ_e^2 is less than the variance (σ_y^2) of the signal.

As it was stated before, the signal may be generated by the following first order difference equation as described in Eq. (3.18):

$$y_n = -qy_{n-1} + x_n$$

and it may be seen that the prediction is as one would expect, and that the error is exactly the quantity x_n, which has variance 1. Such prediction methods are used in various systems for control and data compression, and examples will be discussed in the following section.

For a digitized image scanned line by line, $-q = 0.95$ could be a typical value. This would lead to a significant reduction of the variance

$$\frac{\sigma_e^2}{\sigma_y^2} = \frac{1}{(1-q^2)^{-1}} = 0.1$$

and the conversion of the difference signal will require fewer bits.

Example 4.6 - Linear prediction of (partly) unknown systems

If we want to use a linear predictor like in Example 4.5, the optimal solution for selection of $\mathbf{h} = [h_k]$ is known from (4.21), but these equations require knowledge of the autocorrelation values for the process which is assumed to be stationary. This is not always known in advance and the values have to be estimated from some data, and after that the linear equations have to be solved, i.e. a matrix has to be inverted. This may not be a simple job in a signal processing system, and other methods will have to be used. Here we shall present an **adaptive method** that always keeps the coefficients updated to a good solution sufficiently close to the optimum. The method requires that some method exists for determination of the error in the prediction, and it is often the case when the real value of the

predicted variable may be at hand after the prediction. First we review the variance of the estimation error for any linear predictor:

$$\begin{aligned}\sigma_e^2 &= E[(y-\hat{y})^2] = E[y^2] - 2E[y\hat{y}] + E[\hat{y}^2] \\ &= E[y^2] - 2\sum_{k=1}^{m} h_k E[y_n y_{n-k}] + \sum_{k=1}^{m} h_k \sum_{j=1}^{m} h_j E[y_{n-k} y_{n-j}] \\ &= R_Y(0) - 2\sum_{k=1}^{m} h_k R_Y(k) + \sum_{k=1}^{m} h_k \sum_{j=1}^{m} h_j R_Y(j-k)\end{aligned} \tag{4.22}$$

The last reformulation requires stationarity, but prediction would otherwise also be difficult. The error variance is minimized when $\mathbf{h} = \mathbf{h}_{opt}$ is a solution to the Yule-Walker equations (4.21) which in matrix notation is:

$$\mathbf{R}_Y \mathbf{h}_{opt} = \mathbf{r}_Y \tag{4.23}$$

Instead of solving this equation, we hope for the prediction to be inferred from the quantized error signal through an adaptive algorithm. If we consider the error variance from (4.22), we see that it is a quadratic function of the vector $\mathbf{h}$ and it has minimum at $\mathbf{h}_{opt}$ from (4.23). The optimum is of course unknown if $\mathbf{R}_Y$ is unknown, as it is normally the case, and this is the reason for using an adaptive method. The adaptive method is iterative since it tries to get a better guess for $\mathbf{h}$ in each iteration. For an $\mathbf{h}^{(n)}$ in step n, the adaption algorithm shall use the difference signal e_n and $\mathbf{h}^{(n)}$ to provide a better $\mathbf{h}^{(n+1)}$ in the next step, i.e. an $\mathbf{h}$ closer to the optimum. The direction to change $\mathbf{h}$ must be opposite to the partial derivatives which are calculated from (4.22) with insertion of the optimal $h_{j,opt}$ from (4.23):

$$\begin{aligned}\frac{\partial \sigma_e^2}{\partial h_k} &= 0 - 2\cdot R_Y(k) + \sum_{j=1}^{m} h_j R_Y(j-k) + \sum_{k'=1}^{m} h_{k'} R_Y(k-k') \\ &= 2\cdot\left(\sum_{j=1}^{m} h_j R_Y(j-k) - \sum_{j=1}^{m} h_{j,opt} R_Y(j-k) \right) \\ &= 2\cdot\sum_{j=1}^{m} (h_j - h_{j,opt}) R_Y(j-k)\end{aligned}$$

In vector notation the derivatives with respect to $\mathbf{h}$ are called the gradient and denoted ∇_h which is a vector with m elements:

$$\nabla_{\mathbf{h}} \sigma_e^2 = 2\mathbf{R}_Y(\mathbf{h}-\mathbf{h}_{opt})$$

Moving towards a minimum is moving opposite to the gradient. This process is called **gradient search** (*gradientsøgning*). The problem is to determine the gradient not knowing $\mathbf{R}_Y$ and $\mathbf{h}_{opt}$. To calculate the autocorrelation, averaging over time has to be done, and for non-stationary signals the averages will change such that the solution moves. It is simpler to avoid the averaging by minimizing the current error power instead: e^2. Such an algorithm is known as **LMS, Least Mean Square**, [4.2, Section 6.5]:

$$\nabla_{\mathbf{h}}(e_n^2) = 2e_n \cdot \nabla_{\mathbf{h}} e_n = 2e_n \cdot \nabla_{\mathbf{h}}(y_n - \hat{y}_n) = 2e_n \cdot \nabla_{\mathbf{h}}(y_n - \sum_{k=1}^{m} h_k y_{n-k}) = -2e_n \cdot \mathbf{y}^{(n)}$$

$$\text{where } \mathbf{y}^{(n)} = (y_{n-1}, y_{n-2}, \cdots, y_{n-m})$$

since only $\hat{y}_n$ is dependent of $\mathbf{h}$. We may now state the **LMS algorithm** for updating $\mathbf{h}$:

1. Make an intial guess $\mathbf{h}^{(m)}$ (we assume that first step has n = m)
2. Establish the vector $\mathbf{y}^{(n)} = (y_{n-1}, y_{n-2}, ..., y_{n-m})$, the previous m observed values.
3. Calculate the prediction $\hat{y}_n$ and the error

   $$e_n = y_n - \hat{y}_n = y_n - \sum_{k=1}^{m} h_k y_{n-k} = y_n - \mathbf{h}^{(n)} \cdot \mathbf{y}^{(n)}$$

   from the observed value y_n.
4. Calculate

   $$\mathbf{h}^{(n+1)} = \mathbf{h}^{(n)} + \alpha \cdot e_n \cdot \mathbf{y}^{(n)}$$

   which moves $\mathbf{h}$ in the opposite direction of the gradient. The coefficient α is introduced to vary the adaption speed. Too large an α causes instability, and $0 \le \alpha < 1/(2\lambda_{max})$ is needed (λ_{max} is the largest eigenvalue of $\mathbf{R}_Y$).
5. Set n = n + 1 and goto step 2.

A further simplification of the LMS algorithm is to take into account only the signs of e and $\mathbf{y}^{(n)}$:

$$\mathbf{h}^{(n+1)} = \mathbf{h}^{(n)} + \beta \cdot \mathrm{sgn}(e_n) \cdot \mathrm{sgn}(\mathbf{y}^{(n)}).$$

Some practical applications of adaptive algorithms for prediction will be shown in Section 4.6.

4.6 Differential waveform coding

The bit rates from conventional PCM (e.g. 64 kbit/s for telephone speech) as described in Section 4.4 are quite high compared with the observation that often there is only a minor difference between two neighbor samples in speech or images. This is also seen from the power spectrum since most power is found at the low frequencies, but on the other hand, the information in the signal is found at higher frequencies so you cannot simply reduce the sampling rate. Thus you may expect to gain something by letting the coding be dependent on the previous sample(s), but you have to prepare for sudden changes carrying the information in the signal.

The very advanced parametric coding schemes of Section 4.2 make extensive use of knowledge of the structure of the information source and the signal to be encoded, but also waveform coding may make use of some knowledge of the signal. The most important waveform coding schemes are the **differential** methods (*differentielle metoder*), where the encoding uses the difference, e, between the current sample, x, and a predicted value, $\hat{x}$. In Figure 4.13 we show a general model for differential encoding, and in Figure 4.14 we show the corresponding decoding.

The prediction may seem to be done in an awkward way since it might seem more precise to use the values x_n directly for the prediction, but these are not found at the decoder, so the decoder has to calculate the basis for the prediction as shown. The quantization might then cause the encoder and decoder to disagree on the prediction and prediction errors will add up. The prediction errors have mean 0, but the encoder and decoder might drift quite much apart which is prevented (in case of no transmission errors) in the **backward prediction** scheme (*baglæns prædiktion*) shown in Figure 4.13.

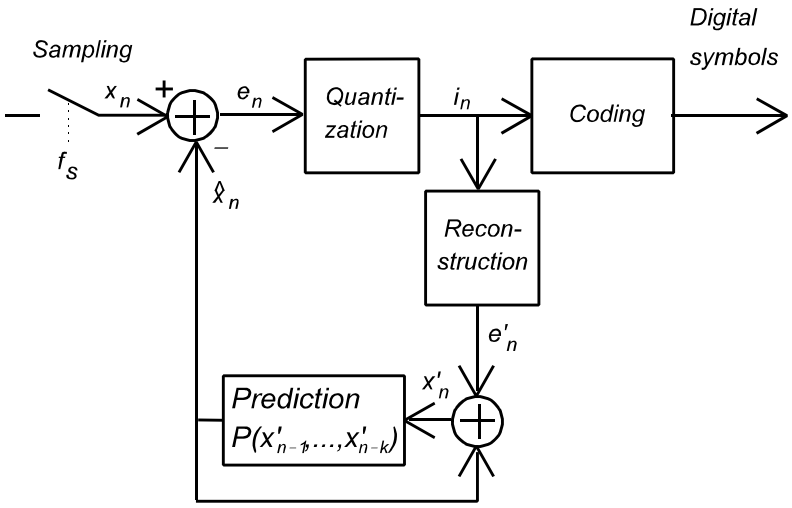

Figure 4.13
Differential coding of analog signals.
A lowpass filter at the input has been omitted.

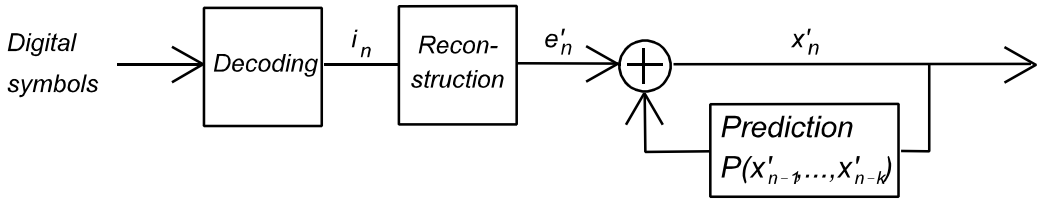

Figure 4.14
Decoding of the differential coding from Figure 4.13.
The lowpass filter at the output has been omitted.

The system is of order k if the prediction depends on the k preceding samples. A special case is conventional PCM from Section 4.4 which has order 0 since $\hat{\mathbf{x}} = 0$ is kept constant. Differential waveform coding is named **DPCM** (differential pulse code modulation). The differential methods reduce the bit rate if the difference signal *e* may be encoded with fewer bits. This is the case when the power (variance) of the difference is minimum. In most applications, the predictor is linear as in Example 4.5:

$$\hat{\mathbf{x}}_n = \sum_{j=1}^{k} \mathbf{h}_j \mathbf{x}_{n-j}$$

In the realization shown in Figure 4.13 we see the quantized values x' as input to the predictor. If we assume a sufficiently fine quantization this will not make any difference, and we neglect this fact in the analysis.

The optimal solution for selection of $\mathbf{h} = [h_i]$ is known from Example 4.5: The error variance is minimized when $\mathbf{h} = \mathbf{h}_{opt}$ is a solution to the Yule-Walker equations (4.21). For $k = 1$ the solution to the Yule-Walker equation is $h_{1,opt} = R_X(1)/R_X(0)$ and the variance is reduced from the signal variance $R_X(0)$ if just some correlation is found (i.e. $R_X(1) \neq 0$) as we have shown earlier in Examples 4.2 and 4.5.

In Figure 4.14 we notice that x' is produced by a filtering of e'. Using the theory of linear discrete systems [4.12] we see that the transfer function contains poles. An unfortunate property for such **autoregressive** systems where the preceding x' are included in the next x' is sensitivity to transmission errors, i.e. e' in Figure 4.14 is not equal to e' in Figure 4.13. Such an error influences the output signal forever after the error. To prevent this, many applications use predictors that cancel out the effect of an transmission error after some time. Such a predictor is

$$\hat{x}_n = \sum_{i=1}^{k} h_i x_{n-i} + \sum_{i=1}^{\ell} g_i e_{n-i}.$$

The variance of e is slightly increased but the last operation (**moving average**) gives some information about errors in e which in that case is moved away from average 0, and a reasonable choice of g_i may cancel the error out. We have added zeros to the transfer function and as we will see in Section 4.6.3 we usually find more zeros than poles.

4.6.1 DPCM

Conventional DPCM has a fixed prediction function and the coefficients in the prediction are chosen as a compromise, e.g. from analysis of a large set of typical signals, or they are chosen to simplify the realization.

The most simple DPCM system is of order 1 and the prediction is the preceding sample, i.e. $\hat{x}_n = x'_{n-1}$. From (4.22) we get

$$\sigma_e^2 = R_X(0) - 2\cdot 1\cdot R_X(1) + 1\cdot R_X(0)\cdot 1 = 2R_X(0)\left(1 - \frac{R_X(1)}{R_X(0)}\right)$$

Reduction of the error variance compared to the signal variance is thus found if $R_X(1) > R_X(0)/2$, i.e. the sample values are much correlated.

The difference signal has to be quantized and the most simple solution is to quantize in only two intervals: $e \geq 0$ and $e < 0$. This system (with the simple prediction, x'_{n-1}) is known as **delta modulation, DM**, (*deltamodulation*) which has its name from the reconstructed signal $e' = \pm\Delta$. This very coarse quantization requires a larger sampling frequency than otherwise needed for a bandlimited signal. This is due to the phenomenon called **slope overload** (*hældningsoverstyring*), since the maximum slope of a signal x(t) which may be followed by a DM system with the sampling frequency f_s is determined by

$$\frac{dx}{dt} \leq \Delta\cdot f_s$$

To see how the sampling rate depends on the signal frequency we may look at the sine signal $x(t) = G\cdot\sin 2\pi ft$ with maximal $dx/dt = 2\pi fG$. If we quantize as fine as we did for the faintest signal in PCM A-law companding, i.e. $\Delta/G = 2^{-11}$ (cf. Section 4.4), and use $f = 3.4$ kHz we get $f_s = 43.7$ MHz. This is clearly too much which is due to the logarithmic companding where the small quantizing steps are only used over a small fraction (2^{-6}) of G. Slope overload may be found in all differential coding schemes.

We shall now calculate the signal-to-noise-ratio for the quantization noise in DM. If we assume no slope overload, the quantization error is restricted by $-\Delta \leq e \leq \Delta$ and if we assume a uniform distribution within this interval we have $\sigma_e^2 = \Delta^2/3$. The spectrum of the noise is very broad and almost uniform up to f_s, i.e. the power spectral density is $S(f) = \Delta^2/(3\cdot 2f_s)$ for $|f| < f_s$. The reconstruction of the signal is done with a lowpass filter with limiting frequency f chosen such that no slope overload is found, i.e. $2\pi fG = \Delta\cdot f_s$ or $f = \Delta f_s/(2\pi G)$ which gives the noise power at the output as

$$\frac{\Delta^2}{6f_s}\cdot 2\frac{\Delta f_s}{2\pi G} = \frac{\Delta^3}{6\pi G},$$

and since the power of a sine signal is $G^2/2$, we obtain

$$SNR = \frac{G^2}{2} \cdot \frac{6\pi G}{\Delta^3} = 3\pi \left(\frac{G}{\Delta} \right)^3 = \frac{3}{8\pi^2} \left(\frac{f_s}{f} \right)^3$$

Requiring SNR = 30 dB gives $f_s = 30 \cdot f$ or about 100 kHz for telephone speech, still more than conventional logarithmic PCM.

DM is not sensitive to transmission errors since the signal is just translated 2Δ by an error and this translation is removed using a bandpass filter for reconstruction, e.g. 300 - 3400 Hz, or it is cancelled out when the maximal signal amplitude is reached. DM does not require any kind of frame synchronization since all binary symbols are of the same value in contrast to e.g. PCM where 8 different interpretations are possible. Thus DM is a robust system which has found military applications where the robustness is preferred to the speech quality.

Another example of a DPCM system is [4.3, J.43] for coding of 15 kHz audio signals. The signal is sampled as in J.41 with 32 kHz and then uniformly quantized with 14 bits. Every other sample is compressed from 14 bits to 10 bits using Near Instantaneous Companding (Section 4.4) and the 10 bits are transmitted. For the samples in between, the difference to the average of the two surrounding samples is calculated:

$$e_n = x_n - \frac{x_{n-1} + x_{n+1}}{2}$$

The difference is compressed to 9 bits (Near Instantaneous Companding over 16 samples) resulting in an average of 9.5 bit per sample, a modest gain. The same principle is used with 16 kHz sampling in [4.3, J.44].

4.6.2 ADPCM

In the previous section we used a fixed prediction and a fixed quantization, both chosen as some compromise for all the possible signals. It may be expected that a significant gain could be obtained by trying to adapt parameters of the DPCM system to the signal found: **adaptive PCM, ADPCM**.

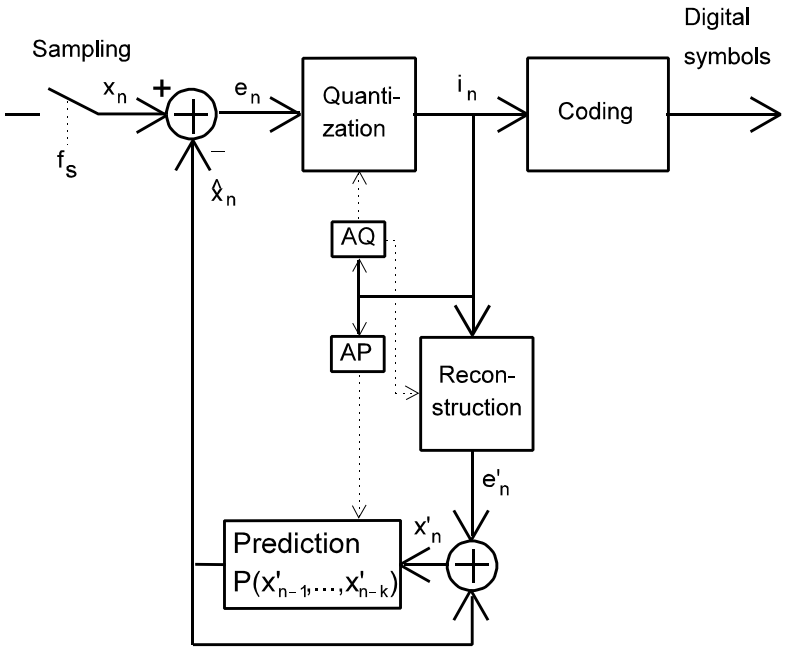

Figure 4.15
Adaptive differential coding of analog signals.
A low pass filter at the input has been omitted.

As stated (and shown in Figure 4.15) two characteristics of the system may be adapted: adaptive quantization, **ADPCM-AQ**, and adaptive prediction, **ADPCM-AP**. The corresponding decoder is shown in Figure 4.16. In the next section we give an example of a combined ADPCM-AQ-AP (the ITU-T recommendation for speech coding) which uses methods similar to those introduced here. First, we shall treat methods for **adaptive quantization**. For the **adaptive prediction** the algorithm from Example 4.6 may be used.

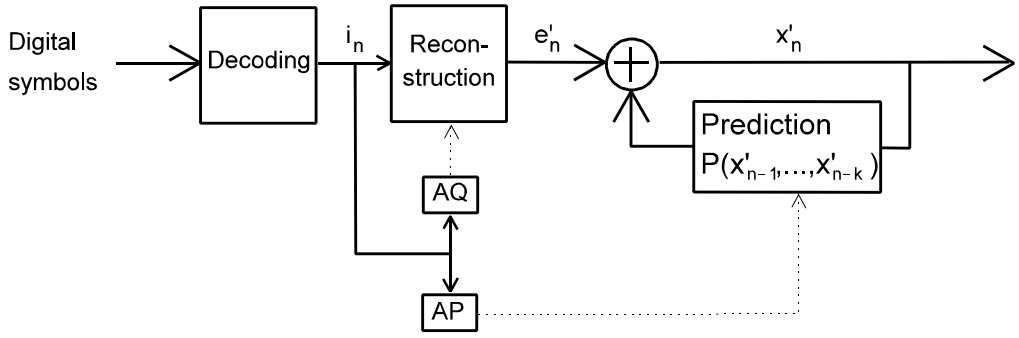

Figure 4.16
Decoding of the adaptive differential coding from Figure 4.15.
The low pass filter at the output has been omitted.

From the discussion above connecting slope overload with sampling frequency we see that the sampling frequency might be reduced if the quantization step Δ could be adjusted to the slope of the signal such that large steps are found for a large slope and small steps where the signal is slowly varying or very small. Consider the following example of **adaptive delta modulation** where the quantization steps Δ are calculated as:

Last 3 e'	New Δ
$+\Delta$ $+\Delta$ $+\Delta$	2Δ
$+\Delta$ $+\Delta$ $-\Delta$	Δ
$+\Delta$ $-\Delta$ $+\Delta$	$\Delta/2$
$+\Delta$ $-\Delta$ $-\Delta$	Δ
$-\Delta$ $+\Delta$ $+\Delta$	Δ
$-\Delta$ $+\Delta$ $-\Delta$	$\Delta/2$
$-\Delta$ $-\Delta$ $+\Delta$	Δ
$-\Delta$ $-\Delta$ $-\Delta$	2Δ

The quantization step is increased (2Δ) at steep slopes (3 differences to the same side in a row) and it is decreased ($\Delta/2$) when differences oscillate between positive and negative values, a possible indication of a too coarse quantization. When the signal has a (local) minimum or maximum the stepsize is kept since there is no explicit indication that it should be wrong. With such a system one obtains a reasonable speech quality even with f_s = 30 - 40 kHz. The adaption system is close to being self synchronized since differences in step size are removed when the maximal or minimal step size is reached.

A similar adaptive quantization algorithm may be used for quantization with more intervals [4.2, Section 4.10]. Assume that the size of quantization intervals is controlled by some parameter, D, e.g. the size of the smallest quantization interval in a logarithmic compressor characteristic. Now we let D be updated by the following function

$$D_n = D_{n-1}^{\beta} \cdot M(i_{n-1}, i_{n-2}, \ldots).$$

If β is chosen slightly less than 1, errors in the quantization intervals at the decoder will disappear after some time. The factor $M(i_{n-1}, i_{n-2}, \ldots)$ depends only of a limited part of the error history and it is the main adapting factor which increases for large errors and

decreases for small errors as we saw in the example with adaptive delta modulation. In the next section, we shall present the ITU-T standard for ADPCM which uses a similar method.

Here we have only treated one-dimensional time dependent signals for which the predictor just goes back in time. For image and video signals there are more neighbors to a sample point, i.e. around the point, especially to the left and above for pictures which are progressively scanned (for video known as **intraframe prediction**), and for video also points in the preceding picture (**interframe prediction**). The reference [4.2] gives many details about waveform coding of video signals with ADPCM methods.

4.6.3 The ITU-T recommendation for ADPCM of telephone channels

A telephone channel is not only used for transmission of speech signals, but also for data transmission which is performed by modulating (in a modem) signals that sound like tones. Other tone signals are also found such as signaling of digits from a push button telephone. Therefore it is necessary to take a number of different signal characteristics into account when designing a waveform coder for a general telephone channel which the ITU-T recommendation G.726 [4.13] is meant for. G.726 describes ADPCM for bit rates 40, 32, 24, and 16 kbit/s, and it replaces (and is compatible with) some previous recommendations G.721 (32 kbit/s) and G.723 (24 and 40 kbit/s) found in [4.3]. 32 kbit/s is thought as the general application for coding of speech signals (e.g. used in DECT) and modem signals (up to 4800 bit/s) while the purpose of 40, 24, and 16 kbit/s is to provide more channels in overload situations in the so-called Digital Circuit Multiplication Equipment, DCME. Especially 40 kbit/s allows replacement of 64 kbit/s without sacrificing too much of the speech quality and allowing data transmission with up to 12 kbit/s. The low bit rates are only suited for speech signals.

Figure 4.17 shows the block diagram of the encoder and Figure 4.18 is the corresponding decoder. Now we shall explain some of the signals shown and some of the algorithms used. The system is thought as a **transcoder** that translates signals coded with PCM A-law (or μ-law) (8000 samples per sec., Section 4.4) to 40, 32, 24, or 16 kbit/s signals with 5, 4, 3, or 2 bits per sample, respectively, and the decoder performs the opposite process. This explains the blocks at the input and output with the additional remark that synchronous

coding adjustment is introduced such that a sequence of several transcoders and ordinary PCM systems serially connected does not introduce reconstruction errors after each transcoder.

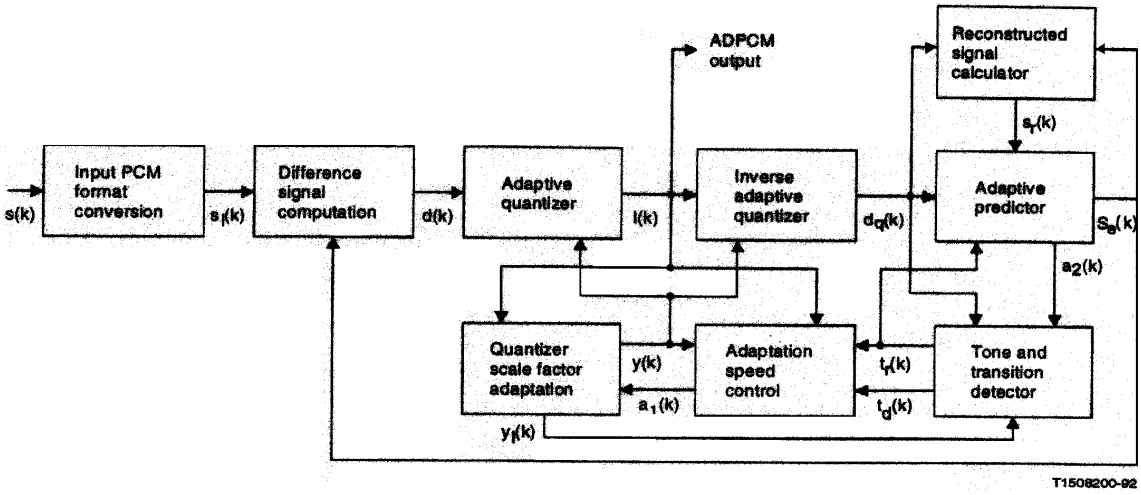

Figure 4.17
Encoder for ITU-T G.726 ADPCM.

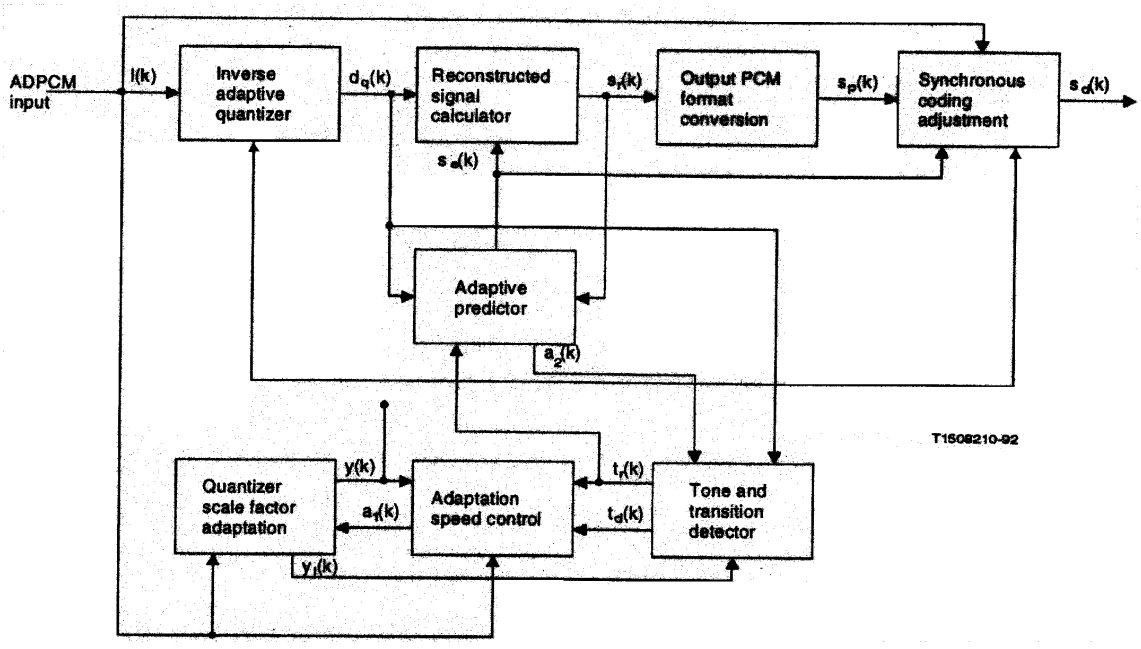

Figure 4.18
Decoder for ITU-T G.726 ADPCM.

First we treat the adaptive quantization. The difference signal d (*e* in the preceding section) is scaled with 2^y and then quantized with a fixed logarithmic compressor characteristic into 32, 16, 8, or 4 intervals respectively for the four bit rates. This is equivalent to the scaling of D in the preceding section. The signal y is calculated as

$$y = \alpha y^{(u)} + (1-\alpha)y^{(l)}$$

where $y^{(u)}$ (**u**nlocked) is meant for fast amplitude variations as they are found in speech, while $y^{(l)}$ (**l**ocked) is meant for other signals where the amplitude varies more slowly (e.g. tone signals and modem signals) and the quantization is relatively constant. The factor α is calculated from some time averages of the signals resulting in that α tends towards 1 for speech and towards 0 for the other signals. The details are found in G.726 [4.13]. The unlocked y-signal is calculated as

$$y_n^{(u)} = \beta y_{n-1} + (1-\beta)W(I_{n-1})$$

where β = 31/32, and W is a non-linear function of the coded differences I_n. This corresponds to the updating introduced in Section 4.6.2 since y is the logarithm of the size of the quantization intervals. The slow, locked signal is a lowpass filtered version of $y^{(u)}$:

$$y_n^{(l)} = \frac{63}{64}y_{n-1}^{(l)} + \frac{1}{64}y_n^{(u)}.$$

The prediction is also adaptive. Generally one would expect the prediction of a signal to become better with higher order, but as stated in Section 4.5.1, the gain for higher orders is small, and a high order increases the possibility of instability in case of transmission errors. Thus the order is quite modest, i.e. 2. As described in Section 4.6, zeros in the transfer function improve the performance in case of transmission errors. There are 6 zeros:

$$\hat{x}_n = s_e = \sum_{i=1}^{2} a_i x_{n-i} + \sum_{i=1}^{6} b_i d_{n-i}$$

The coefficients are determined by a gradient search where the gradient is estimated in much the same way as described in Example 4.6. For details, please refer to G.726 [4.13]. A thorough analysis and references is found in [4.2].

The subjective speech quality of the 32 kbit/s system is close to the quality of the conventional 64 kbit/s PCM, and the 40 kbit/s system is comparable with 64 kbit/s. The

quality for data transmission is not so good since the 64 kbit/s PCM system transmits 33.4 kbit/s which is not possible for the ADPCM systems. The low speeds have a somewhat inferior quality.

The ITU-T has also standardized a 64 kbit/s high quality ADPCM system [4.3, G.722]. The system uses 16 kHz sampling frequency allowing the analog signal to have bandwidth 7 kHz. The input signal is split into two bands, 0 - 4 kHz and 4 - 8 kHz. The sampling frequency is then reduced in each of the bands to 8 kHz by deleting every second sample (decimation with factor 2, cf. Section 3.6). Each of the decimated signals are then encoded with an ADPCM algorithm similar to the G.726 recommendation. The lower band is coded with 6 bits per sample, while only 2 bits per sample are used for the upper band. At the decoder, interpolation (cf. Section 3.6) recreates the sampling frequency of 16 kHz and by filtering the two bands are joined together to the 7 kHz signal.

4.7 References for Chapter 4

4.1 Poul Levin: "Dansk fonetik", Nyt Nordisk Forlag Arnold Busk, København 1974.

4.2 Nuggehally S. Jayant and Peter Noll: "Digital Coding of Waveforms". ISBN 0-13-211913-7, Prentice-Hall Inc. 1984.

4.3 ITU-T: "Recommendations from IXth Plenary Assembly, Melbourne 1988, Blue Book", International Telecommunication Union, Geneva 1989, ISBN 92-61-03xxx-x.

4.4 Yun Q. Shi and Huifang Sun: "Image and Video Compressing for Multimedia Engineering, Second Edition". ISBN 0-8493-7364-6, CRC Press 2008.
(Used in the DTU course 34241 Digital Video Technology).

4.5 ITU-R: "CCIR Recommendations", International Telecommunication Union, Geneva.

4.6 Bent Thøgersen: "NICAM. Digital stereolyd i TV", Teleteknik 1990 nr. 4, side 266-272.

4.7 A. Rix, "Perceptual Speech Quality Assessment – A Review", Proc. of ICASSP'10, Montreal, Canada, May 2004.

4.8 S. Winkler, and P. Mohandas, "The Evolution of Video Quality Measurement: From PSNR to Hybrid Models ", IEEE Trans. Broadcasting, Vol. 54(3), pp. 660-668, Sep. 2008.

4.9 K. Sayood: "Introduction to Data Compression". Morgan Kaufmann Publishers, 1996, ISBN 1-55860-346-8.
(Used in the DTU course 34240 Data Compression).

4.10 J.B.H. Peek: "Communications Aspects of the Compact Disc Digital Audio System", IEEE Communications Magazine, Vol. 23 No. 2, February 1985, pp. 7-15.

4.11 ITU-R: "Recommendation BT.601-5: Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios", International Telecommunication Union, Geneva 1998.

4.12 B.P. Lathi: “Signal Processing and Linear Systems”. ISBN 0-19-521917-1, Oxford University Press N.Y., 1998.
(Used in the DTU courses 31605 and 31606: Signals and Linear Systems)

4.13 ITU-T: "40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM), Recommendation G.726", International Telecommunication Union, Geneva 1990.

5. BASEBAND TRANSMISSION

5.1 Introduction

The connections in transmission systems are of many types, but they may be divided into

- Baseband systems, where signals are transmitted without any change in frequency or with only minor adjustments of the frequency range. An example is metallic transmission lines, and the characteristics of these will be explained in Section 7.2. Metallic transmission lines are well known in daily life, e.g. telephone subscriber lines or lines in private communication systems e.g. local area networks or systems for traffic control. Optical fibers used with direct detection is another example, see Section 7.3.
- Modulated systems, where the signal is modulated to a much higher frequency band. We shall discuss modulation in Chapter 6. Modulated systems are primarily found in wireless communication, but sometimes modulation is also used in wired systems, e.g. optical fibers may be used for so-called coherent transmission.

We shall call transmission of the first type **baseband transmission** (*basisbåndstransmission*). This chapter presents an analysis of such systems and some examples. It is also of importance to modulated systems, since the analysis is similar and much of the signal processing in these systems may take place in the baseband. Many existing systems transmit voice, video, or broadcast radio programs in analog form. However, it is consistent with current technology to assume that all new systems will use digital transmission, and this course will treat only digital systems.

In digital communication we treat signals of the form

$$v(t) = \sum_{k=-\infty}^{\infty} a_k g(t - kT)$$

where the pulses, g(t), are displaced by integer multiples of T and scaled by factors a_k which contain the information that we want to transmit. The duration of each pulse may be greater than T, and the overlapping parts are then added. As explained before in Section 3.5, this form of transmission with pulses is also known as **Pulse Amplitude Modulation** or **PAM**.

Since one symbol a_k is transmitted in each time period T, we define a rate for this. The rate is called **symbol rate** (*symbolhastighed*) or more out-of-date **modulation rate** (*modulationshastighed*). The unit for symbol rate is **baud**:

$$\text{symbol rate} = \frac{1}{T} \text{ baud} \tag{5.1}$$

The alphabet (*alfabetet*) for the digital symbols contains a number, M, of symbols, i.e. $a_k \in \{b_1, b_2, ..., b_M\}$. Since M symbols may be described with $\log_2 M$ binary symbols (**bit**) we define a rate that allows us to compare various systems on a common basis, the **bit rate** (*signaleringshastighed*), R, as

$$\text{bit rate } R = \frac{\log_2 M}{T} \text{ bit/s.} \tag{5.2}$$

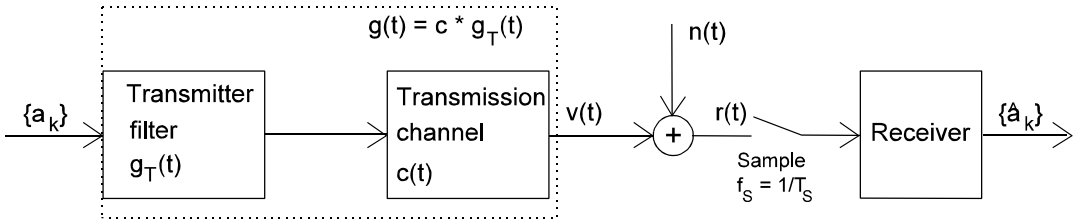

Figure 5.1
Baseband transmission system, PAM.

The pulse shape used in the transmission is basically created in the **transmitter filter** (*sendefilter*). In modern implementations, this often consists of a digital filter with impulse response $\mathbf{g_T}$ followed by a (low-pass) filter which creates the waveform $g_T(t)$ for the transmission channel. The digital filter uses a sampling rate high enough to specify the spectrum in the appropriate frequency range for the transmission channel, and the sampling

rate $1/T_S$ is often chosen as a multiple of 1/T (oversampling, introduced in Section 3.5). A higher sampling rate (higher **oversampling factor** T/T_S) may help controlling the spectrum of the transmitted signal without too many demands on the (low-pass) filter as we saw in Example 3.12.

When the PAM-signal is transmitted through the **transmission channel** (*transmissionskanal*) with response c(t) (cf. Figure 5.1, transfer function $C(\omega)$) it results in

$$v(t) = c * \sum_{k=-\infty}^{\infty} a_k g_T(t-kT) = \sum_{k=-\infty}^{\infty} a_k c * g_T(t-kT) = \sum_{k=-\infty}^{\infty} a_k g(t-kT),$$

where we have introduced $g(t) = c*g_T(t)$ (or transformed $G(\omega) = G_T(\omega)C(\omega)$), and neglected that the signals are sampled. The signal is seen to be a PAM signal still, but the pulse shape may have changed.

The oversampling at the **receiver** should be chosen to take into account all relevant information, i.e. all relevant frequencies in the received signal. We assume that some kind of low pass filtering has been done before the sampling such that the sampling theorem is fulfilled. We shall discuss the choice of oversampling factor further in Sections 5.3 and 5.7. The discussion of the transmitter and Figure 5.1 seem to suggest that the same oversampling factor has to be chosen in the transmitter and in the receiver. This is not necessarily true since the oversampling factor in the transmitter may be chosen for the reasons given above whereas the oversampling factor in the receiver is more concerned with performance of the receiver and implementation problems. In this course we shall not distinguish between the two factors since we are mostly analyzing receiver performance. Thus the received waveform is sampled also with $1/T_S$ as shown in the figure and the system may be seen as a time discrete system where the transmitted and received signals are both sampled with $1/T_S$ and so are also the pulses $g_T(t)$ and g(t). However, this setup ignores an important problem about synchronization between transmitter and receiver. For the following theory to work, it is necessary that the pulse shape g(t) is the same for all symbols in the signal v(t), but if the sampling at the receiver is not synchronized with the symbols, the pulse samples change. It is often the case that the sampling at receiver is free-running, and such an implementation requires an interpolation between the sampled values to find the correct values of the received signal, i.e. the values synchronized with the pulse. We shall return to this important problem in Section 5.7, but for the moment we neglect it, not to

complicate matters too much. For high sampling rates, the problem of interpolation may be neglected, but usually the oversampling factor is a moderate number.

Two problems disturb the reception of $\{a_k\}$:

- The **distortion** in the transmission channel may introduce a dependency between the symbols. In Sections 5.3 and 5.5 we formulate conditions for the function $G(\omega)$ such that the detection of the symbols may be done independently of the neighbor symbols or where the dependency is known and used in the detection. In Chapter 7 we shall return to a description of various transmission channels, and we shall also describe the so-called channel equalization that may be used by the receiver to counteract the influence of the channel.
- The **noise**, normally **additive**, as shown in Figure 5.1. We shall assume that noise is modeled as **Gaussian noise** (*gaussisk støj*) whose amplitude is normal distributed with density function (as introduced in Section 3.1):

$$f_N(n) = \frac{1}{\sqrt{2\pi}\,\sigma_N} e^{-n^2/2\sigma_N^2}$$

In most of the applications we shall describe, the power spectrum of the noise is **white** (*hvidt*), i.e. the power spectrum is constant, and the constant is usually known as:

$$S_N(\omega) = N_o/2, \; -\infty < \omega < \infty.$$

This is of course not found in practice, but constant $S_N(\omega)$ in the appropriate frequency range is enough. The frequency range for the noise is limited with the filter before sampling mentioned above. Note that this filtering may introduce a dependency between the noise samples. However, if the noise is bandlimited to exactly half of the sampling frequency, it is possible to prove that the noise samples are independent, so such an effect is not considered in the following derivations of an optimal receiver. The physical unit for $N_o/2$ is W/Hz, i.e. energy (Joule). Thermal noise is an example of physical noise being **AWGN**, **Additive White Gaussian Noise**.

With the assumption just done, the noise samples are independent and have zero mean, such that the autocorrelation becomes $R_N(0) = \sigma_N^2$ and 0 for other shifts. We may then relate the noise variance at each sample to the spectral density by

$$S_N(\omega) = \sum_{k=-\infty}^{\infty} T_S R_N(k) e^{-j\omega k T_S} = T_S \sigma_N^2 = \frac{N_o}{2} \tag{5.3}$$

The noise variance plays an important role in the detection as we shall see. It may seem a little strange that the variance of the sampled noise increases with a decreasing T_S for a given constant N_o. However, decreasing T_S increases the frequency range for the receiver, and thus more noise is to be taken into account since all noise in the range $-\pi/T_S \le \omega \le \pi/T_S$ is seen by the receiver, so the received noise power becomes

$$\frac{1}{2\pi}\int_{-\pi/T_S}^{\pi/T_S} S_N(\omega)\,d\omega = \frac{1}{2\pi}\int_{-\pi/T_S}^{\pi/T_S} \frac{N_o}{2}\,d\omega = \frac{N_o}{2T_S} = \sigma_N^2$$

The received signal is r(t) = v(t) + n(t), and the task of the optimal receiver is to find an estimated sequence $\{\hat{a}_k\}$ which deviates as little as possible from $\{a_k\}$, i.e. with a minimum **error probability** (*fejlsandsynlighed*). The process is known as **detection** (*detektion*), and it is described in the next section where we at first consider a single, sampled pulse, $a_0 g(jT_S)$, $a_0 \in \{b_1,.., b_M\}$ with no interaction between the pulses. We shall base our discussion on this discrete time (sampled) version of the signal. By choosing the sampling frequency (the oversampling factor, T/T_S) high enough we can include all information about the signal, and the results will be valid also in time continuous form as it is found in the literature. The noise is also bandlimited to $|\omega| \le 2\pi/(2T_S)$. In Section 5.3 we return to the composite PAM-signal where interaction between the pulses may be found. The pulse g(t) is assumed to have finite energy (as before without impedance reference)

$$E = \int_{-\infty}^{\infty} g^2(t)\,dt \tag{5.4}$$

In a practical implementation, the pulse g(t) has to be of finite duration, i.e. g(t) = 0 outside some finite time interval. Most of the time we shall not be concerned with the time continuous transmission channel, and we shall treat transmission of some pulse $\mathbf{g} = (g_0, g_1, ..., g_{N-1})$ which is a (finite duration) sampled version of g(t), i.e. $g_j = g(jT_S)$. Note that the pulse may be longer in time than T, so the transmitted pulses in a signal may be overlapping. In the next section we shall see that the energy of the time discrete pulse

$$E = \sum_{j=0}^{N-1} T_S g^2(jT_S) = \sum_{j=0}^{N-1} T_S g_j^2 \tag{5.5}$$

plays an important role for the detection of the sampled, received signal as does the variance of the noise.

Thus we shall treat transmission of some pulse $\mathbf{g} = (g_0, g_1, ..., g_{N-1})$ of finite duration disturbed by added noise with a given variance σ_N^2 (or a given spectral density for the noise, $N_o/2$). At first we do not consider synchronization problems so we assume that the time instants of the transmitter and receiver are the same and thus that all symbols are transmitted with the same pulse shape.

The receiver is discussed in detail in Section 5.2, and the selection of a proper pulse shape is treated in Sections 5.3 and 5.5. In Section 5.4 we discuss the efficiency of communication schemes. In Section 5.6 we treat how to mitigate the effects of the physical transmission channel (c(t) in Figure 5.1) through a process called equalization. After this, the discussion of receiver implementation will be resumed with the problems of synchronization (Section 5.7), and we continue with a way to improve the relation between the transmission channel and the PAM-signal since we present methods to create symbol sequences with certain properties, e.g. suitable power spectra or certain temporal properties in Section 5.8, and the chapter ends with a description of digital subscriber lines where most of the methods in this chapter are applied.

5.2 Optimal receiver for digital signals

5.2.1 MAP and ML optimal receivers

Assume first that we observe a single stochastic variable

$$y = x + n$$

where x assumes the discrete values A and 0 with probabilities q and $1-q$, and n is a noise sample with known density function f_N. We seek a decision function, d(y), that assigns a value of x to each observation, and we want to **minimize the probability of error**. As it is seen from Figure 5.2, the contribution to the error probability for each observed value, y', will be one of the terms

$$qf_N(y' - A) \quad \text{or} \quad (1 - q)f_N(y')$$

The expected error is minimized by taking the smaller term in each case. The decision function, which depends on the probabilities of the signal values and the noise density is known as a **MAP, maximum a posteriori probability** rule. We minimize the error probability by selecting

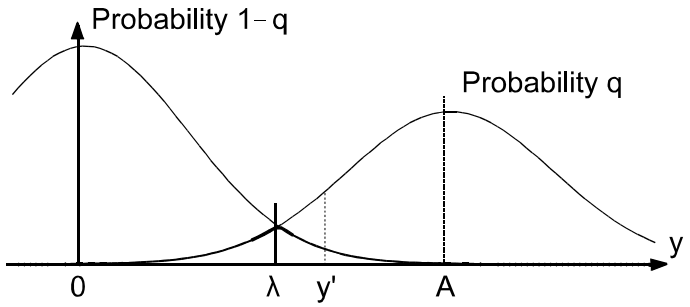

Figure 5.2
MAP, Maximum a posteriori decision rule

$$d(y) = A \quad \text{if } L(y) = \frac{f_N(y-A)}{f_N(y)} > \frac{1-q}{q} \tag{5.6}$$

where L(y) is known as a **likelihood ratio**. It is often practical to take the logarithm of this ratio (**log likelihood**), and introducing Gaussian noise with mean 0 and standard deviation σ_N we get

$$\ln L(y) = \ln f_N(y-A) - \ln f_N(y) = \ln e^{-(y-A)^2/2\sigma_N^2} - \ln e^{-y^2/2\sigma_N^2} = \frac{A \cdot y - A^2/2}{\sigma_N^2}$$

and the **(MAP) decision rule** (*beslutningsfunktion*) follows from (5.6)

$$d(y) = \begin{cases} A \text{ for } y > \lambda \\ 0 \text{ otherwise} \end{cases} \qquad \lambda = \frac{A}{2} + \frac{\sigma_N^2}{A} \ln \frac{1-q}{q} \tag{5.7}$$

For q = 1/2 the result is obvious. The decision rule is illustrated in Figure 5.2 which shows that the **threshold**, λ, (*tærskelværdi*) is determined as the point where the two contributions to the density function for y intersect. It should be noted that the two types of error (deciding d = A when x = 0 and d = 0 for x = A), can have very different probabilities. Depending on the application the wrong decision may have very different consequences, and other criteria than minimum error probability may be preferred.

In digital communication applications the signal values are often assumed to be equally likely, and if that cannot be assumed to be a good approximation we may use scrambling (cf. Section 2.6) to get a better distribution. When the symbols are equally likely, the thresholds of (5.7) and the forthcoming (5.8) all become much simpler: The MAP rule above maximizes the probability of x given the observation of y, i.e. P(x|y), but using the relation

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

and that P(x) is constant for equally likely symbols, we may instead maximize the conditional probability, $P(y|x)$, of the received symbol, i.e. we always select the symbol closest to the one that is transmitted as the threshold in (5.7) describes for q = ½. Such a decision rule is known as **ML, maximum likelihood**.

5.2.2 Matched filter detection

We are now in a position to describe an **optimal receiver** in the case of a signal with known pulse shape. Besides in digital communication, the task of detecting a known pulse shape with some unknown parameters, usually amplitude and time of arrival, in a background of Gaussian noise is found in other applications as well. These include radar and medical imaging.

Assume that the observation of the received signal in Figure 5.1 is

$$\mathbf{r} = (r_0, r_1, \ldots, r_{N-1}) = (v_0 + n_0, v_1 + n_1, \ldots, v_{N-1} + n_{N-1})$$

Again n_i is white Gaussian noise with standard deviation σ_N and the vector, **v**, is either **0** or the pulse $\mathbf{g} = (g_0, g_1, g_2, \ldots, g_{N-1})$ corresponding to transmission of symbol a = 0 or a = 1, respectively. We obtain the decision function by applying (5.6) for each value in the pulse **g** (i.e. N times) and noting that the corresponding N noise samples are independent (note the remark above around the noise being filtered) such that their density is found as a product of the individual densities:

$$L(\mathbf{r}) = \frac{\prod_{j=0}^{N-1} f_N(r_j - g_j)}{\prod_{j=0}^{N-1} f_N(r_j)} > \frac{1-q}{q}$$

and the corresponding **(MAP) decision rule** calculated like in (5.7) becomes

$$\mathbf{d(r)} = \begin{cases} \mathbf{g} & \text{for } s = \sum_{j=0}^{N-1} T_S g_j r_j > \lambda \\ \mathbf{0} & \text{for } s \le \lambda \end{cases} \tag{5.8}$$

with the **threshold**

$$\lambda = \frac{1}{2}\sum_{j=0}^{N-1} T_S g_j^2 + T_S \sigma_N^2 \ln\frac{1-q}{q} = \frac{E}{2} + \frac{N_o}{2}\ln\frac{1-q}{q}$$

where we have inserted (5.5) and (5.3). The decision function in the optimal receiver for a pulse (a symbol) consists of two steps:

- First we collect the relevant information in a single number, s, which in the literature is know as a **statistic**.
- Next this number is compared to a threshold, λ, and the most likely transmitted pulse (symbol) is decided from this comparison.

In the usual literature, [5.13, 5.3, 5.12], the same procedure is described, but the pulses are treated in continuous time which means that the sums are replaced with integrals.

From the derivation above we may conclude that the statistic may be computed as a **correlation** of the received signal and the known pulse. By constructing a linear filter with impulse response

$$\mathbf{g_R} = a\frac{1}{\sqrt{E}}(g_{N-1},\ldots,g_0) \tag{5.9}$$

we get a value proportional to the statistic from (5.8) as the output at time $(N-1)T_S$. The constant *a* may be chosen freely as we shall see below, and the constants are used to provide the same physical dimension for input and output. Thus the constants are not important, the only important characteristic is: the impulse response is obtained as the pulse **g** with the time **reversed**. The filter is known as a **matched filter** (*tilpasset filter*) and the optimal receiver with such a filter is shown in Figure 5.3.

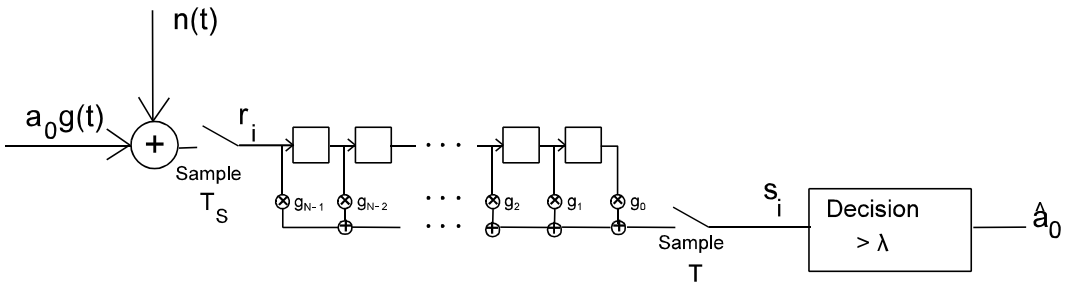

Figure 5.3
Matched filter receiver (without any scaling factor)
for an on-off signaling system (a_0 = 0 or 1).

The correct operation of the impulse response in (5.9) is easily proven by the calculation of the output at time $(N-1)T_S$:

$$\mathbf{g_R} * \mathbf{r}[(N-1)T_S] = \sum_{j=0}^{N-1} T_S r_j a \frac{1}{\sqrt{E}} g_{N-1-j}^{(R)} = \frac{a}{\sqrt{E}} \sum_{j=0}^{N-1} T_S r_j g_j = \frac{a}{\sqrt{E}} s$$

($g^{(R)}$ are the coefficients of $\mathbf{g_R}$). Note that if $\mathbf{r} = \mathbf{g}$ (pulse received without noise) the output becomes $a\sqrt{E}$ which has the same dimension as the transmitted signal $\mathbf{g}$, if a is chosen properly.

We have proven the decision rule (5.8) and use of a matched filter directly by minimizing the probability of error, but sometimes in the literature the optimization is done differently: Which linear filter maximizes the so-called signal-to-noise ratio, E/N_o, at the receiver? The solution to this problem is of course also the matched filter since as we shall see in the next section, the error probability depends on the signal-to-noise ratio.

Example 5.1 Matched filter receiver

In Example 2.11 we treated Barker sequences as examples of pulses with good correlation properties. Let

$$\mathbf{g} = (-1, 1, -1, -1, 1, 1, 1)$$

be the Barker sequence of length 7. The receiver for this is shown in the figure.

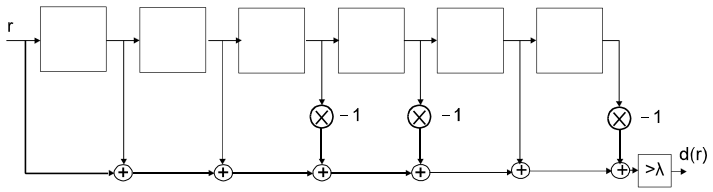

Figure 5.4
Receiver (without any scaling factor)
for a length 7 Barker sequence

When **g** is input to the filter, the response is

$$-1,\ 0,\ -1,\ 0,\ -1,\ 0,\ 7,\ 0,\ -1,\ 0,\ -1,\ 0,\ -1$$

which is the autocorrelation function of the pulse as expected. Note that the receiver calculates the correlation as described in Example 2.11. The pulse spreads over a relatively large bandwidth and thus it is not relevant in most communication systems, but there are cases where spreading out the bandwidth makes the system less susceptible to narrowband noise. The Barker pulses are used in radars where the pulse is detected in noise, and the detection time allows calculation of distances. We illustrate the operation of the matched filter receiver in the following figure generated by the following program where the noise variance is 1:

```
Ts=1;
g=[-1 1 -1 -1 1 1 1];
E=sum(g.^2)*Ts;
sigma=sqrt(1);
q=0.1;
lambda=E/2+Ts*sigma^2*log((1-q)/q);
% Generate sequence with one pulse and noise
v=[zeros(1,15),g,zeros(1,15)];
r=v+sigma*randn(1,length(v));
plot(r)
hold on
% Matched filter is g reversed
gR=fliplr(g);
out=Ts*conv(r,gR);
stem(out,'r')
```

If the threshold is passed at wrong times, a false echo is found at the radar, and if the threshold is not passed at the right time, the echo is not seen. We try to simulate these two situations below, and the results in a particular run are the probabilities $P_{false} = 2.0\%$ and $P_{miss} = 32.0\%$. The miss rate may seem high, but the pulse is repeated often (we have set q = 0.1), so it may not be so important for a radar. In the next section, we shall demonstrate how to calculate such probabilities.

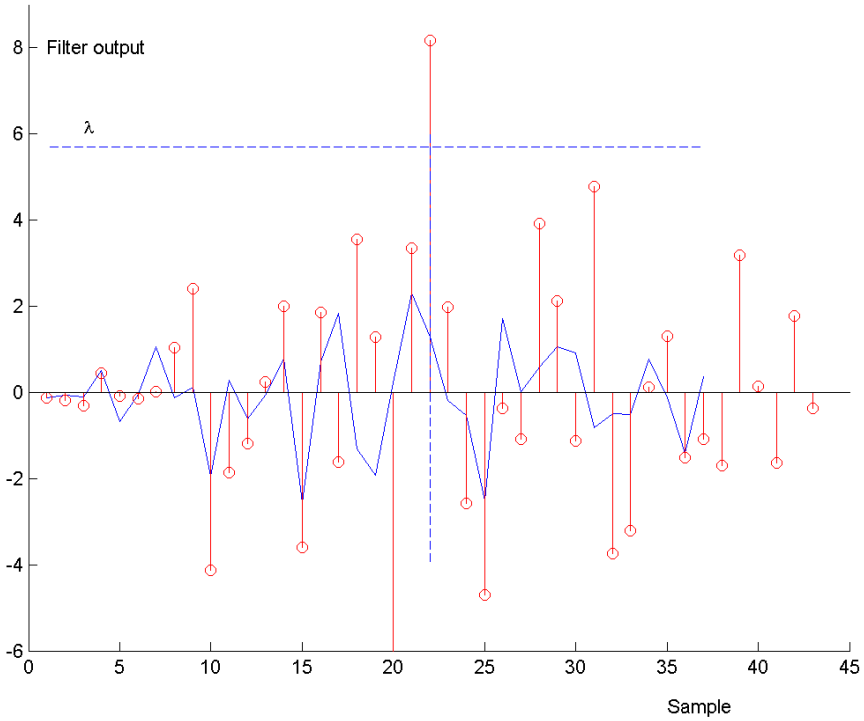

Figure 5.5
Signal with Barker pulse (at sample 16 - 22) and noise ($\sigma_N^2 = 1$) added. Output from matched filter (without scaling factor $a/\sqrt{E}$) shown as stems together with threshold λ for q = 0.1.

```
Nexp=1000;
L=length(g);
error1=0;
error0=0;
for experiment=1:Nexp
   r_withpulse=g+randn(1,L)*sigma;
   % Count # of missed pulses using (5.8) detection:
   out=Ts*conv(r_withpulse,gR);
   if out(length(g))<=lambda
      error1=error1+1;
   end
   r_withoutpulse=randn(1,L)*sigma;
   out=Ts*conv(r_withoutpulse,gR);
   % Count # of falsely detected pulses:
   if out(length(g))>lambda
      error0=error0+1;
   end
end
Pfalse=error0/Nexp
Pmiss=error1/Nexp
```

5.2.3 Error probability for on-off signaling

So far we have talked about the detection problem in general terms. As a communication problem, what we have discussed here may be called '**on-off signaling**' or '**on-off keying, OOK**', which is commonly used with optical fibers with direct detection (Section 7.3).

Expressed in the terminology from the beginning of this chapter, the alphabet is {0, 1} with probabilities 1 − q and q, respectively.

At the output of the matched filter we have the following value at the time of the sampling for decision

$$z = \frac{a}{\sqrt{E}} \sum_{j=0}^{N-1} T_S g_j r_j = \frac{a}{\sqrt{E}} \sum_{j=0}^{N-1} T_S g_j v_j + \frac{a}{\sqrt{E}} \sum_{j=0}^{N-1} T_S g_j n_j$$

which is seen to contain a signal term and a noise term. The signal term equals

$$\begin{cases} \dfrac{a}{\sqrt{E}} \displaystyle\sum_{j=0}^{N-1} T_S g_j^2 = \frac{a}{\sqrt{E}} E = a\sqrt{E} & \text{with probability } q \\ 0 & \text{with probability } 1 - q \end{cases}$$

Since a sum of Gaussian variables is still Gaussian, the noise term is Gaussian. The noise contributions are independent, each with variance σ_N^2, so the variances may be added to give the variance

$$\left(\frac{a}{\sqrt{E}}\right)^2 \sum_{j=0}^{N-1} T_S^2 g_j^2 \sigma_N^2 = \left(\frac{a}{\sqrt{E}}\right)^2 T_S E \sigma_N^2 = a^2 T_S \sigma_N^2 = a^2 \frac{N_o}{2}$$

where we have used (5.3). The standard deviation of this term becomes $a\sigma_N\sqrt{T_S} = a\sqrt{N_o/2}$. It may be noted that $a\lambda/\sqrt{E}$ is the amplitude where the distributions for signal and no signal intersect. The tails of the two distributions indicate the probabilities of wrong decisions. The situation is like the situation described in Figure 5.2 with a different abscissa. Thus the **error probability** (*fejlsandsynligheden*) may be calculated as

$$\begin{aligned} P_e &= (1-q)\cdot P(s > \lambda \mid \text{signal off (0)}) + q\cdot P(s < E-\lambda \mid \text{signal on (1)}) \\ &= (1-q)\cdot P(z > a\lambda/\sqrt{E} \mid \text{signal off (0)}) \\ &\quad + q\cdot P(z < a(E-\lambda)/\sqrt{E} \mid \text{signal on (1)}) \\ &= (1-q)\, Q\left(\frac{a\lambda}{\sqrt{E}a\sqrt{N_o/2}}\right) + qQ\left(\frac{a(E-\lambda)}{\sqrt{E}a\sqrt{N_o/2}}\right) \\ &= (1-q)\, Q\left(\frac{\lambda/\sqrt{E}}{\sqrt{N_o/2}}\right) + qQ\left(\frac{\sqrt{E}-\lambda/\sqrt{E}}{\sqrt{N_o/2}}\right) \end{aligned} \tag{5.10}$$

where we have used the Q-function from Eq. (3.3). Q(x) gives the probability for a Gaussian variable with mean 0 and variance 1 to exceed x, and the probability for this with another variance σ^2 is calculated as $Q(x/\sigma)$. Note that *a* cancelled out.

For a **maximum likelihood receiver** we have q = ½. For comparison with other systems we introduce the **average energy** $\mathbf{E_{av}} = E/2$. With this assumption, the error probability for a maximum likelihood receiver is calculated from (5.10) as

$$P_e = (1-\tfrac{1}{2})\,Q\left(\frac{E/2/\sqrt{E}}{\sqrt{N_o/2}}\right) + \tfrac{1}{2}\,Q\left(\frac{\sqrt{E}-E/2/\sqrt{E}}{\sqrt{N_o/2}}\right)$$

$$= 2\cdot\tfrac{1}{2}\,Q\left(\frac{\frac{1}{2}\sqrt{E}}{\sqrt{N_o/2}}\right) = Q\left(\sqrt{\frac{E}{2N_o}}\right) = Q\left(\sqrt{\frac{E_{av}}{N_o}}\right) \qquad (5.11)$$

P_e as a function of the signal-to-noise ratio E_{av}/N_o is shown in Figure 5.7. In the first expression in the last line, we may interpret the numerator as **half of the distance** between the two possible values of the output from the matched filter: 0 and $\sqrt{E}$, and the denominator is the standard deviation for the noise. This interpretation is much used the following section and later in Chapter 6 about modulation where two-dimensional signal constellations are introduced

5.2.4 Optimal receiver for other one-dimensional signal constellations

If we return to our original detection problem for a PAM-signal, i.e. detection of a single, sampled pulse, $a_0 g(jT_S)$, $a_0 \in \{b_1,.., b_M\}$, with no interaction between the pulses, we may have the impression from the preceding section, that since M signals are found, we need M matched filters. This is not the case, since each filter will have the same shape, so a single filter is enough, but the output from the filter may assume several values, $b_j\cdot\sqrt{E}$, plus the noise. Thus the optimal receiver is just as the one shown in Figure 5.3 with a modified decision function, often named a **slicer**, which we illustrate in the following. With the Gaussian noise as before we have the following density function at the output given that the transmitted symbol is $a_0 = b_j$

$$f_R(z\,|\,a_0 = b_j) = \frac{1}{\sqrt{2\pi}\,\sigma_N\sqrt{T_S}}\,e^{-(z-b_j\sqrt{E})^2/2\sigma_N^2 T_S} = \frac{1}{\sqrt{\pi N_o}}\,e^{-(z-b_j\sqrt{E})^2/N_o}$$

An example is shown in Figure 5.6.

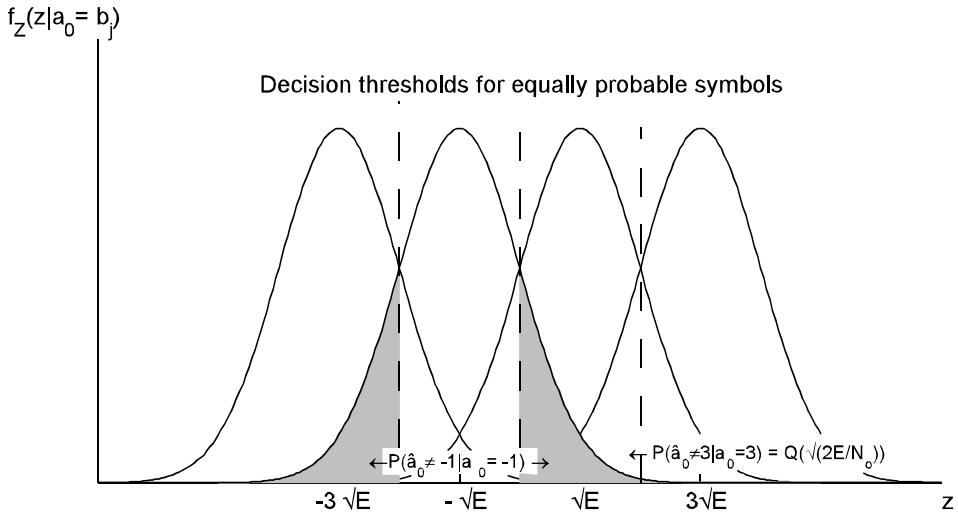

Figure 5.6
Probability densities for the output, z, from a matched filter in an example with M = 4 equally spaced antipodal symbols with equal probabilities

Usually we consider **maximum likelihood receivers**, i.e. $P(b_j) = 1/M$, where the decision thresholds are the **midpoints** between the values $b_j \cdot \sqrt{E}$. Even if there is a small gain of letting the two outermost values be slightly deviating, it is common practice that the b_j are equally spaced. Further, it is advantageous to have mean 0 and thereby low average energy, and the signals are placed symmetrical around zero, also known as **antipodal signaling**, which is obtained by $b_j = 2j - 1 - M$. The distance between the different (noiseless) outputs from the matched filter then becomes $2\sqrt{E}$ as shown in Figure 5.6. Now we may find the probability P_e for a wrong decision â:

$$
\begin{aligned}
P_e &= \sum_{j=1}^{M} P(\hat{a}_0 \neq b_j \,|\, a_0 = b_j) P(a_0 = b_j) \\
&= 2 \cdot Q\left(\frac{\sqrt{E}}{\sqrt{N_o/2}} \right) \cdot \frac{1}{M} + (M-2) \cdot 2 Q\left(\frac{\sqrt{E}}{\sqrt{N_o/2}} \right) \cdot \frac{1}{M} \\
&= \frac{2M-2}{M} Q\left(\frac{\sqrt{E}}{\sqrt{N_o/2}} \right) = \frac{2M-2}{M} Q\left(\sqrt{\frac{2E}{N_o}} \right)
\end{aligned}
\qquad (5.12)
$$

Note again that the numerator in the expression before the last one may be interpreted as half of the distance between the values $b_j \cdot \sqrt{E}$. As before the error probability depends on the relation between pulse energy and noise, and letting M grow makes it possible to transfer as much information in one symbol with as little P_e as you like, but it may require

much energy, and comparison of different systems must be based on a fixed average energy, which for M antipodal signals is

$$E_{av} = \sum_{j=1}^{M} \frac{1}{M} \cdot b_j^2 E = \sum_{j=1}^{M} \frac{(2j-1-M)^2}{M} \cdot E = \frac{M^2-1}{3} \cdot E$$

As for on-off signaling we may then calculate the symbol error probability as a function of E_{av} and the one-sided spectral density N_o of white, Gaussian noise:

$$P_e = \frac{2M-2}{M} Q\left(\sqrt{\frac{6E_{av}}{(M^2-1)N_o}} \right) \tag{5.13}$$

For M = 2 this may be compared with the on-off case from Equation (5.11). The signal-to-noise ratio E_{av}/N_o has to be doubled to obtain the same performance for an on-off signaling system. This shows the advantage of the antipodal signal. The error probability is shown in Figure 5.7 for various $M = 2^{\nu}$ and for the on-off system. Doubling M in the antipodal system gives one binary symbol more, i.e. $\nu + 1$, and the cost is approximately 6 dB in E_{av}/N_o. If E_{av}/N_o on the other hand is kept constant, a doubling of M increases the error probability significantly.

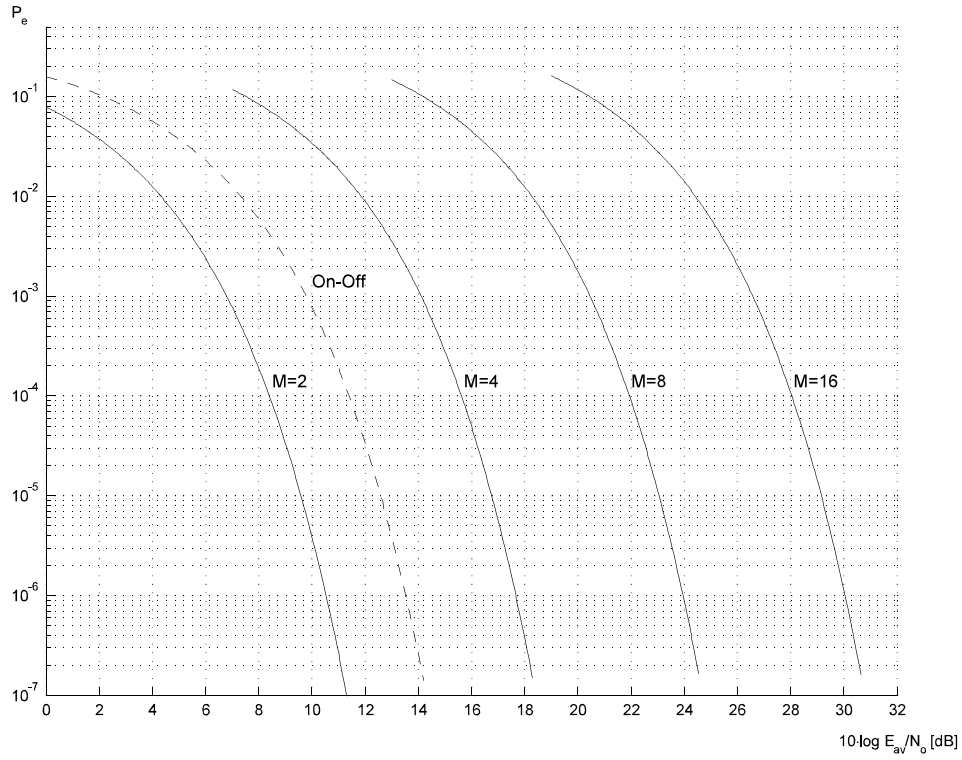

Figure 5.7
Symbol error probability P_e as a function of signal-to-noise ratio for PAM with M antipodal symbols (solid lines) and on-off signals (dashed line).

Equation (5.13) and similar equations give the **symbol error probability**, but normally a symbol originates from a number of binary symbols, e.g. $M = 2^\nu$ corresponds to ν bits, and to calculate the **bit error probability** one needs the explicit mapping between the bits and the symbols. An appropriate choice of this mapping is **Gray-coding** where the most likely symbol error to the neighbor symbol produces exactly one bit error. An example of a Gray code for M = 8 is shown in Table 5.1. The Gray code is easily extended to larger M using the symmetry which may for example be seen in the extension from M = 4 (two bits of the first 4 rows) to M = 8. Using Gray-coding, the average bit error probability becomes $P_b = P_e/\nu$ for $M = 2^\nu$, but note that it varies with the bit position, e.g the three positions of the M = 8 code have bit error probabilities of p/4, p/2 and p, respectively, where p is the Q-part of (5.12), giving the overall bit error probability of $(p/4 + p/2 + p)/3 = 7/12p = P_e/3$ with P_e from (5.12).

Symbol	Gray code
−7	000
−5	001
−3	011
−1	010
1	110
3	111
5	101
7	100

Table 5.1
Example of Gray coding.

To make a reasonable comparison of the binary performance of systems with different M, we introduce the **energy per information bit**, E_b, instead of the previously used average energy per symbol, E_{av}. The relation between these is of course:

$$E_b = \frac{E_{av}}{\log_2 M} \tag{5.14}$$

Thus for $M = 2^\nu$ equally likely and equidistant symbols encoded with a Gray code we get from (5.13) as shown in Figure 5.8 for various $M = 2^\nu$ and for the on-off system:

$$P_b = \frac{2\cdot 2^\nu - 2}{\nu 2^\nu} Q\left(\sqrt{\frac{6E_{av}}{(2^{2\nu}-1)N_o}}\right) = \frac{2\cdot 2^\nu - 2}{\nu 2^\nu} Q\left(\sqrt{\frac{6\nu}{(2^{2\nu}-1)}\cdot\frac{E_b}{N_o}}\right)$$

Note that even if the abscissae in Figures 5.7 and 5.8 are both for signal-to-noise ratios measured in dB, there is a difference in the definition. **Be careful when using dB!**

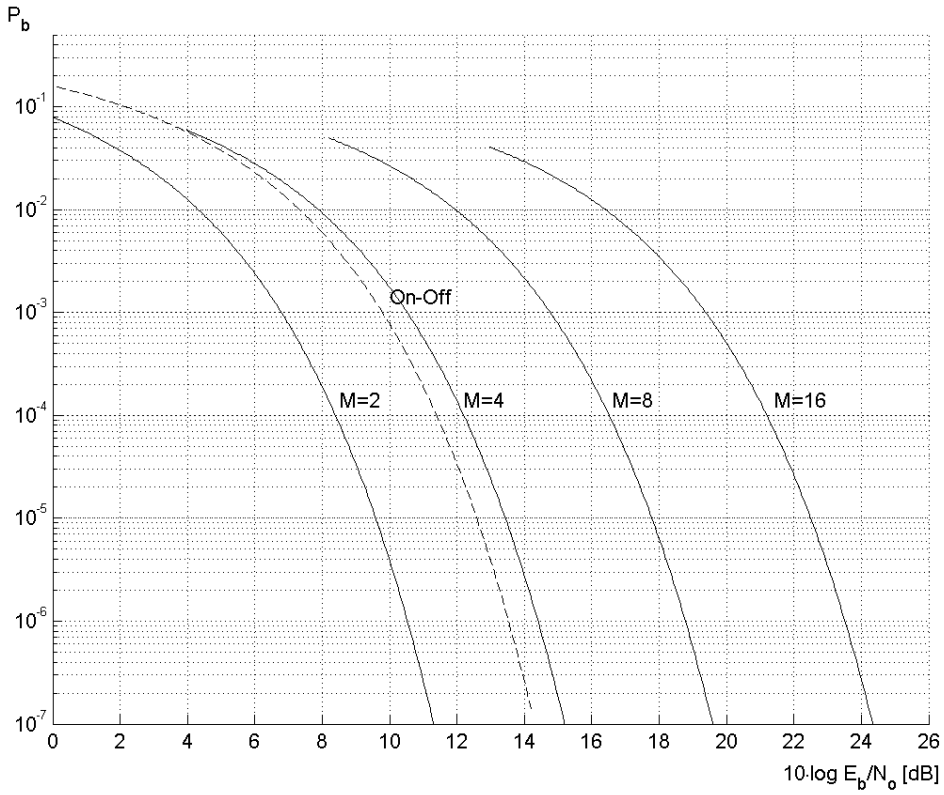

Figure 5.8
Bit error probability P_b as a function of signal-to-noise ratio per bit for PAM with M antipodal symbols (solid lines) and on-off signals (dashed line).

In this section we have described signal constellations where all possible transmitted signals are scalings of one basic form, one-dimensional signals. Other constellations are also found, and we shall return to this in Chapter 6.

Example 5.2 - Simulation of detection of PAM signal with a matched filter

The following MATLAB script makes decisions on 10000 samples of a 4-PAM signal ($a_j = \pm 1, \pm 3$) received with a noise density $N_o = 0.3$. The pulse used in the PAM-signal is triangular and sampled $m = 4$ times. 10 symbols of the signal with noise is shown in Figure 5.9.

```
T=1;
N0=0.3;                          %Spectral density of noise
N=10000;
m=4;
T_s=T/m;
Variance=N0/2/T_s;      % (5.3)
```

```
pulse=(1:m)/m;          %Triangular pulse
% Generate 4-PAM signal oversampled with factor m
a=2*sign(0.5-rand(1,N))+sign(0.5-rand(1,N));
aover=upsample(a,m);
v=conv(aover,pulse);
% Display 10*T of signal with no noise
l=stem(0:10*Over-1,v(1:10*m));
hold on
% Add noise
rnoise=v(1:m*N)+sqrt(Variance)*randn(1,m*N);
stem(0:10*m-1,rnoise(1:10*m),':');
```

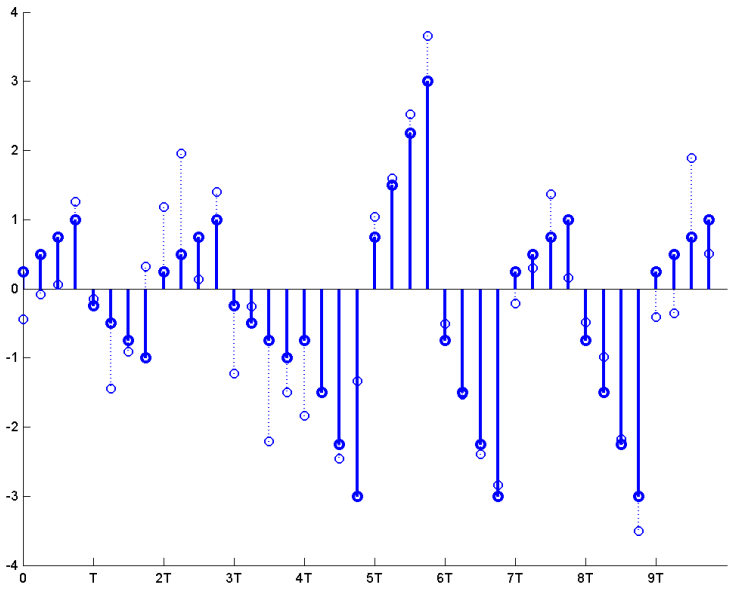

Figure 5.9
Example of 10 time-discrete pulses (solid lines)
in an M = 4 PAM signal oversampled 4 times
and noise added (dashed lines).

Below we show the detection with a matched filter. Since the pulse and the corresponding matched filter has length equal to T, there is no interaction between various symbols, and the output from the filter may be used for detection just by comparing it to the thresholds $\pm 2\sqrt{E}$ and 0 (at the correct times k·T). The estimated number from (5.12) of errors is 578, and various runs of the script produce from 537 to 613 errors.

```
E=sum(pulse.^2)*T_s;          %Energy of time discrete pulse
% The matched filter is the reversed pulse scaled properly
match=1/sqrt(E)*fliplr(pulse);
% The received signal is passed through the matched filter
filterOut=T_s*conv(rnoise,match);
% The output is sampled at the correct times (i*m*T_s)
```

```
sampledOut=filterOut(m:m:m*N);
% and the decisions are made with boundaries -2√E, 0, +2√E
d=sign(sampledOut).*(2*(abs(sampledOut)>=2*sqrt(E))+1);
% Count errors and compare with estimate from (5.12), M = 4
errors=sum(d~=a)
estimate=round(N*(2*4-2)/4*qfunc(sqrt(2*E/N0)))
```

5.3 Intersymbol interference

In a real communication system, the channel has certain properties due to physical mechanisms, e.g. a bandwidth limitation. Models for transmission channels will be given in Chapter 7. The inherent properties mean that g(t) cannot be selected without constraints. Some of the constraints may be mitigated by the use of channel equalization, and this is also a subject of Chapter 7. Here we shall treat conditions for the total impulse response

$$x(t) = g_R * c * g_T(t),$$

where $g_R(t)$ is the receiver filter. The system with filters is illustrated in Figure 5.10

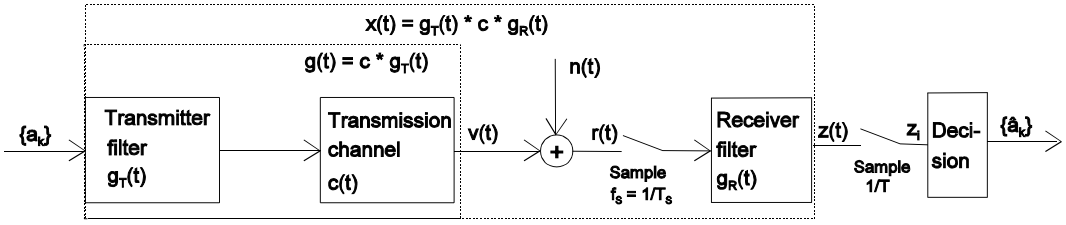

Figure 5.10
Baseband transmission system with total impulse response.

At the output of this filter we have

$$z(pT_S) = \sum_{k=-\infty}^{\infty} a_k\, x(pT_S - kT) + \sum_{k=-\infty}^{\infty} g_R * n(pT_S - kT)$$

As stated in Section 5.2.2 the decisions in the receiver are based on the values of this signal at certain times separated by the time interval T. How these times are related to the T_S is determined by the symbol synchronization which will be introduced in Section 5.7. Not to complicate the notation we just state the properly sampled values of z as

$$z_i = \sum_{k=-\infty}^{\infty} a_k\, x_{i-k} + n_i = a_i x_0 + \sum_{k\neq i}^{\infty} a_k\, x_{i-k} + n_i$$

We see that the symbol to be detected at time iT, a_i, is influenced by the neighbor symbols through the weights $x_{i-k} = x((i-k)T)$ and then comes the added noise, n_i. The influence from the neighbor symbols is called **intersymbol interference**, **ISI** (*intersymbolinterferens*).

Based upon the filter output values $\{z_\ell\}$ we shall estimate a_i. One method could be to do the detection symbolwise only on basis of z_i as in the previous section, but then we have to **eliminate the ISI** which presents the following condition on x (or X(ω)):

$$x_n = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{jn\omega T}d\omega = \begin{cases} c & \text{for } n = 0 \\ 0 & \text{for } n \neq 0 \end{cases} \tag{5.15}$$

where we have introduced a constant c for x_0. This condition for a system without ISI was formulated in 1928 by H. Nyquist [5.1], and it is known as the **(first) Nyquist criterion**. The constant $c = x_0$ is not found in most literature, but it is used here to indicate that there is a difference in dimension between input (dimensionless) and output (squareroot of energy). The Nyquist criterion is of course fulfilled by any pulse x(t) limited to the range $0 \le t < T$ as in Example 5.2 since the pulses then do not overlap and ISI is not present. However, such pulses occupy in principle an infinite range of frequencies, and for most channels some kind of bandwidth limitation exists, so we will examine the Nyquist criterion in that case. Many functions X(ω) could be the solution to the Nyquist criterion since the sampling theorem tells that too few samples are known for bandwidths larger than π/T to define a unique function. To characterize the criterion in the frequency domain we may calculate

$$\int_{-\infty}^{\infty} X(\omega)e^{jn\omega T}d\omega = \sum_k \int_{(2k-1)\pi/T}^{(2k+1)\pi/T} X(\omega)e^{jn\omega T}d\omega = \sum_k \int_{-\pi/T}^{\pi/T} X(\omega + k\cdot\frac{2\pi}{T})e^{jn\omega T}d\omega$$

$$= \int_{-\pi/T}^{\pi/T} \sum_k X\left(\omega + k\cdot\frac{2\pi}{T}\right)e^{jn\omega T}d\omega = \int_{-\pi/T}^{\pi/T} X_{eq}(\omega)e^{jn\omega T}d\omega$$

where we have introduced

$$X_{eq}(\omega) = \begin{cases} \sum_k X\left(\omega + k\cdot\frac{2\pi}{T}\right) & \text{for } |\omega| \le \pi/T \\ 0 & \text{for } |\omega| > \pi/T \end{cases} \tag{5.16}$$

i.e. $X_{eq}(\omega)$ is built from segments of X(ω) placed on top of each other in the interval $-\pi/T \le \omega \le \pi/T$. This is shown in Figure 5.11.

Since $X_{eq}(\omega)$ is bandlimited, $-\pi/T \le \omega \le \pi/T$, and sampled with $x_n = x(nT)$ from (5.15) as values, the sampling theorem states that X_{eq} is unique, and

$$X_{eq}(\omega) = \begin{cases} cT & \text{for } |\omega| \le \pi/T \\ 0 & \text{for } |\omega| > \pi/T \end{cases} \tag{5.17}$$

This gives the **Nyquist criterion** expressed in the frequency domain. The frequency π/T (1/(2T) Hz) is known as the **Nyquist frequency** (*Nyquist-båndbredde*).

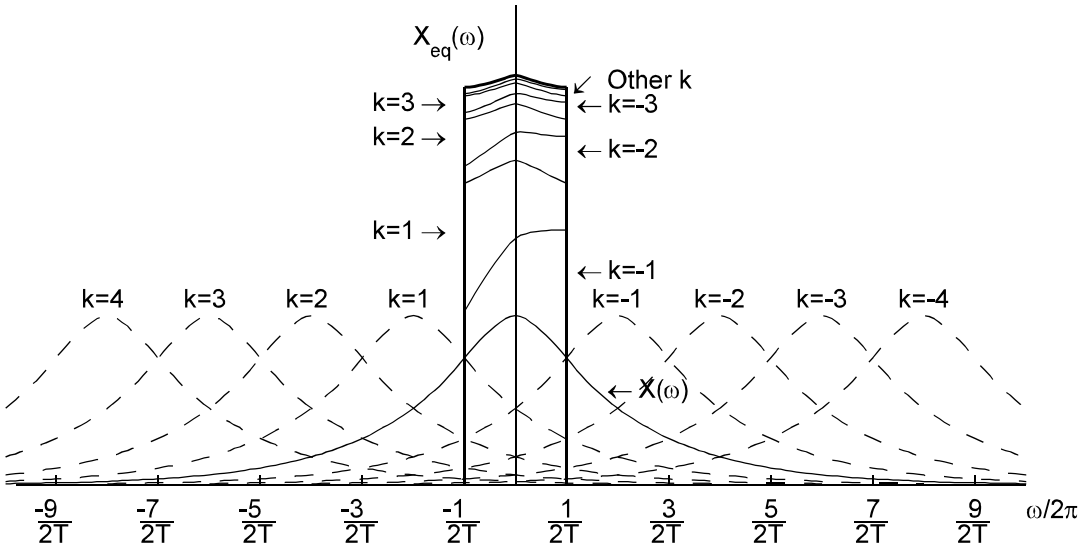

Figure 5.11
Construction of $X_{eq}(\omega)$ from a given $X(\omega)$. Note that this $X(\omega)$ does **not** fullfil the Nyquist criterion.

Now the problem is: Which $X(\omega)$ fulfills the Nyquist criterion? An $X(\omega)$ narrower than the Nyquist frequency, i.e. a "smooth" pulse in the time domain due to predominant low frequencies, cannot fulfill the criterion, which means that ISI cannot be removed with such an $X(\omega)$ for the given symbol rate 1/T, but possibly for a lower rate. If $X(\omega)$ is exactly confined to the Nyquist interval from $-\pi/T$ to π/T, i.e. we have $X(\omega) = X_{eq}(\omega)$ and constant in the interval, the criterion is automatically fulfilled. In the time domain, the pulse is

$$x(t) = c\frac{\sin \pi t/T}{\pi t/T} = c\,\mathrm{sinc}\left(\frac{t}{T}\right) \tag{5.18}$$

which is c for t = 0 and contains the proper zero crossings (nT, n ≠ 0). The pulse is shown in Figure 5.13 in time and frequency domains. Unfortunately, the pulse starts at $-\infty$ such

that implementations have to cut the outer parts off. If it was directly realizable, a symbol rate of 1/T would require a bandwidth of 1/2T Hz. Another problem is that ISI would add up if the sampling instants are not perfectly in synchronism, since a small difference in timing, δ, results in ISI since $\Sigma x(\delta + nT)$ does not converge (x(t) behaves like 1/t). These problems may be solved for a pulse with X(ω) wider than the Nyquist interval, but not for any shape as Figure 5.11 demonstrates.

However, there are many functions being wider than the Nyquist interval that fulfill the criterion. These are known as Nyquist characteristics. The synchronization is less problematic if X(ω) is chosen to be more smooth.

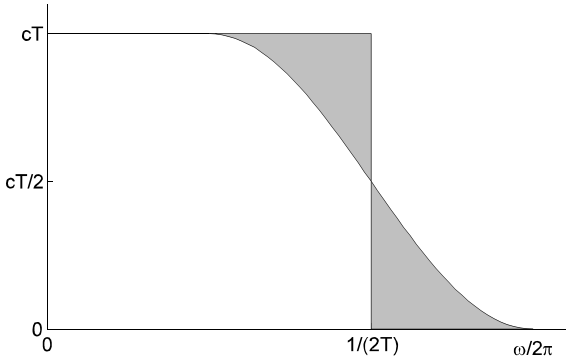

Figure 5.12
Vestigial symmetry.
The two grey areas are symmetrical around (π/T,cT/2).

A simple variation is frequency characteristics with an odd symmetry around (π/T, cT/2) since the piece outside π/T just fills the "hole" below π/T when X_{eq} is made (see Figure 5.12). This is known as **vestigial symmetry** and it was also found out by Nyquist [5.1]. The most important example of such a characteristic is the **cosine roll-off** (*cosinus roll-off*) pulse, also known as **raised cosine** (*hævet cosinus*). The definition contains an α, the roll-off factor, which gives the extra bandwidth used in addition to the Nyquist frequency ($0 \le \alpha \le 1$). The functions x(t) and X(ω) for α = 0.25, 0.5, and 1 are shown at Figure 5.13 (together with α = 0 which is identical to (5.18)).

The definition of the cosine roll-off characteristic is

$$X(\omega) = \begin{cases} cT & 0 \le |\omega| \le (1-\alpha)\frac{\pi}{T} \\ c\frac{T}{2}\left(1 - \sin\frac{|\omega|T - \pi}{2\alpha}\right) & (1-\alpha)\frac{\pi}{T} < |\omega| \le (1+\alpha)\frac{\pi}{T} \\ 0 & |\omega| > (1+\alpha)\frac{\pi}{T} \end{cases} \qquad (5.19)$$

Transforming to the time domain gives

$$x(t) = c\frac{\sin\pi t/T}{\pi t/T}\frac{\cos\alpha\pi t/T}{1-4\alpha^2t^2/T^2} = c\,\mathrm{sinc}\left(\frac{t}{T}\right)\frac{\cos\alpha\pi t/T}{1-4\alpha^2t^2/T^2}$$

The maximum roll-off is found at $\alpha = 1$ giving the double bandwidth. The parameter α is sometimes given as a percentage. Again the x(t) starts at $-\infty$ but the decrease is faster such that an approximation through a cut-off is easier found.

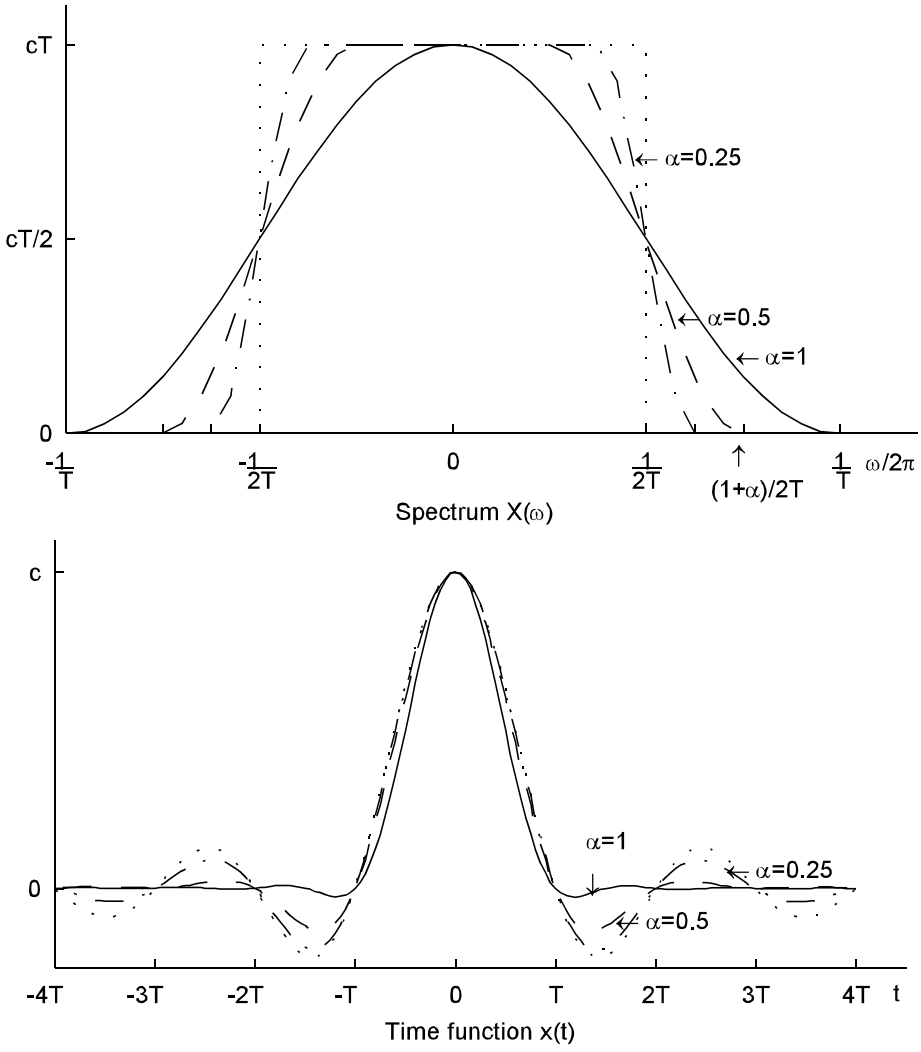

Figure 5.13
Cosine roll-off time function x(t) and spectrum X(ω).
The Nyquist pulse of (5.18) is also shown (dotted lines).

As stated in Section 5.2 the optimal receiver has a matched filter as receiver filter. The transfer function for this filter with N taps is $G_R(\omega) = G^*(\omega)\cdot e^{-j\omega(N-1)T_s}$ (without scaling factors) and the transfer function from input to output becomes

$$X(\omega) = G(\omega)\cdot G^*(\omega)e^{-j\omega(N-1)T_S}$$

The last factor just describes the delay, $(N-1)T_S$, in the decision, and apart from that we have

$$X(\omega) = |G(\omega)|^2 \tag{5.20}$$

which is real and ≥ 0. Thus the minimum error probability is obtained by distributing the total $X(\omega)$ between

$$|G_T(\omega)C(\omega)| = |G(\omega)| = \sqrt{X(\omega)}$$

and the receiver filter

$$|G_R(\omega)| = |G^*(\omega)| = \sqrt{X(\omega)}$$

An arbitrary scaling factor may be applied to the receiver filter since it does not change performance as we demonstrated in Section 5.2.3. A scaling factor in the transmitter filter controls the pulse energy and thereby it changes performance which is dependent on the ratio between pulse energy and noise as we shall see in the next section.

Example 5.3 - Implementation of transmitter and receiver filters

The cosine roll-off characteristic is never found in any filter in a communication system, since as we see from Equation (5.20) what we need is the squareroot of $X(\omega)$ such that the combined effect of the transmitter (incl. channel) and receiver filter gives $X(\omega)$. Using some trigonometric identities we find

$$\sqrt{X(\omega)} = \begin{cases} \sqrt{cT} & 0 \leq |\omega| \leq (1-\alpha)\dfrac{\pi}{T} \\ \sqrt{cT}\cos\left(\dfrac{|\omega|T-\pi}{4\alpha}+\dfrac{\pi}{4}\right) & (1-\alpha)\dfrac{\pi}{T} < |\omega| \leq (1+\alpha)\dfrac{\pi}{T} \\ 0 & |\omega| > (1+\alpha)\dfrac{\pi}{T} \end{cases}$$

As discussed before, different scaling factors may be applied to give the actual transmitter filter (affects performance) and receiver filter (without affecting

performance). The figure shows such a pulse in the time domain (for positive time) as it is implemented in a filter without any scaling and with an oversampling factor of 4, i.e. $T = 4T_S$.

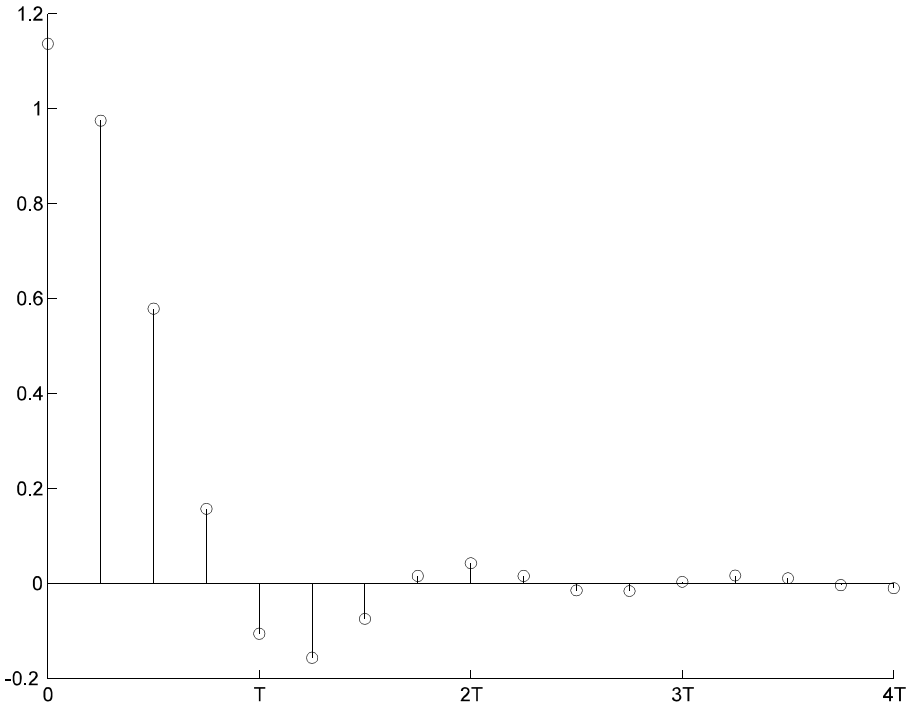

Figure 5.14
Squareroot cosine roll-off pulse (α = ½, c=1) sampled with 4 times the symbol rate. Only positive time is shown, the pulse is symmetrical around t = 0.

Note that this pulse itself does not have zero crossings at t = nT. The correct zero crossings are found when the pulse is used as a receiver filter for the same pulse, since then the convolution of the two, x(nT), is the output.

The transfer function above may be transformed to give the squareroot cosine roll-off pulse, [5.2]:

$$\mathrm{scro}(t) = \sqrt{\frac{c}{T}} \cdot \frac{4\alpha}{\pi} \cdot \frac{\cos\frac{(1+\alpha)\pi t}{T} + \frac{\pi(1-\alpha)}{4\alpha}\,\mathrm{sinc}\frac{(1-\alpha)t}{T}}{1-\left(\frac{4\alpha t}{T}\right)^2}$$

For a practical system the scro(t) has to be cut off at some time. The interval $-4T \le t \le 4T$ as in the figure may be sufficient to remove most of the ISI. Note also the method of Example 3.12 for creating a time-limited pulse from a required transfer function.

The scro(t) may not contain the appropriate scaling for the transmitter and receiver filter. We may calculate its "energy" using the Parseval relation:

$$\int_{-\infty}^{\infty} scro^2(t)\,dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} |Scro(\omega)|^2 d\omega = \frac{1}{2\pi}\int_{-(1+\alpha)\frac{\pi}{T}}^{(1+\alpha)\frac{\pi}{T}} X(\omega)d\omega = \frac{1}{2\pi}\int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} cT\,d\omega = c$$

where the integral of $X(\omega)$ is found using the vestigial symmetry property. Thus the "energy" for any Nyquist characteristic is c defined earlier in this section. As discussed around the introduction of c, its dimension is squareroot of energy, so the c calculated above is not a real energy, and some more scaling is required. For a transmitter filter, the dimension of the pulse should be voltage, and for the receiver, the proper dimension is inverse squareroot of time.

As described in Example 3.12, an implementation in digital hardware also has to take quantization of coefficients into account. The effect on the sidelobes at frequencies where the transfer function should be zero is mentioned in the example, but for an implementation as a transmitter/receiver filter the effect on the intersymbol interference may be of more concern. This effect may be found from studying the zero crossings of a quantized version of scro(t) convolved with itself. An experiment shows that the effect is very small even for a coarse quantization such as 5 - 6 bits.

There are other characteristics with low intersymbol interference. An example is the so-called **Gaussian characteristic**.

$$x(t) = c\,e^{-k\left(\frac{t}{T}\right)^2} \quad \text{with } X(\omega) = cT\sqrt{\frac{\pi}{k}}e^{-\frac{T^2}{4k}\omega^2}$$

The time function never has any zero crossings, but the values at mT can be made arbitrarily small by an appropriate choice of k. E.g. $x(T) = 0.01 \cdot c$ is obtained for $k = 4.6$. In general, more bandwidth is needed than the cosine roll-off characteristics. Derivatives of the Gaussian characteristic may also be used.

In this section we have described systems with no intersymbol interference, but it may be possible that this strategy does not produce the minimum error probability. The receiver filter $G_R(\omega)$ e.g. cancels a zero in the channel transfer function with a high amplification

(a pole) to make the total transfer function as the X(ω) described here, and it may increase the noise more than it reduces the ISI. An optimal receiver in presence of ISI uses the entire sequence $\{z_\ell\}$ for estimation of the symbol a_i. Optimal receivers are treated e.g. in [5.3, 5.13]. The Nyquist criterion is still important and many systems are based on this.

5.4 Efficiency of digital communication systems

In the previous section, we have considered bandwidth limited systems, and we showed that a bandwidth of B Hz allows the symbol rate $1/T = 2B/(1+\alpha)$ since some kind of roll-off characteristic has to be used. The user will of course be interested in **maximizing the amount of information that could be transferred at a given bandwidth**, or in other words the user wants to maximize the efficiency R/B which for an M-PAM system with roll-off α is

$$\frac{R}{B} = \frac{\log_2 M}{TB} \text{ (bit/s)/Hz} = \frac{2\log_2 M}{1+\alpha} \tag{5.21}$$

For M equally likely and equidistant symbols we get from (5.13) and (5.14):

$$P_e = \frac{2M-2}{M} Q\left(\sqrt{\frac{6E_{av}}{(M^2-1)N_o}}\right) = \frac{2M-2}{M} Q\left(\sqrt{\frac{6\log_2 M}{(M^2-1)} \cdot \frac{E_b}{N_o}}\right)$$

For a given probability of error, this equation demonstrates the relation between M and the signal-to-noise ratio E_b/N_o, and (5.21) gives the relation to R/B. The relations for $P_e = 10^{-5}$ are shown in Figure 5.15. If a lower probability of error is chosen, the M-PAM curves move to the right but the picture is still the same.

In the figure you may also compare the R/B performance to the best possible, the so-called **channel capacity** (*kanalkapacitet*), C/B, from the information theory founded by C.E. Shannon in 1948 [5.4] and later developed by others (cf. [5.3, 5.13]). To use the usual formulation of the channel capacity bound, we calculate the (received) power $P_{av} = E_{av}/T = E_b \cdot R$, and the power for the noise could also be calculated as $P_N = BN_o$ since $N_o/2$ is the power spectral density of the noise and BN_o is the noise after a filter with bandwidth B:

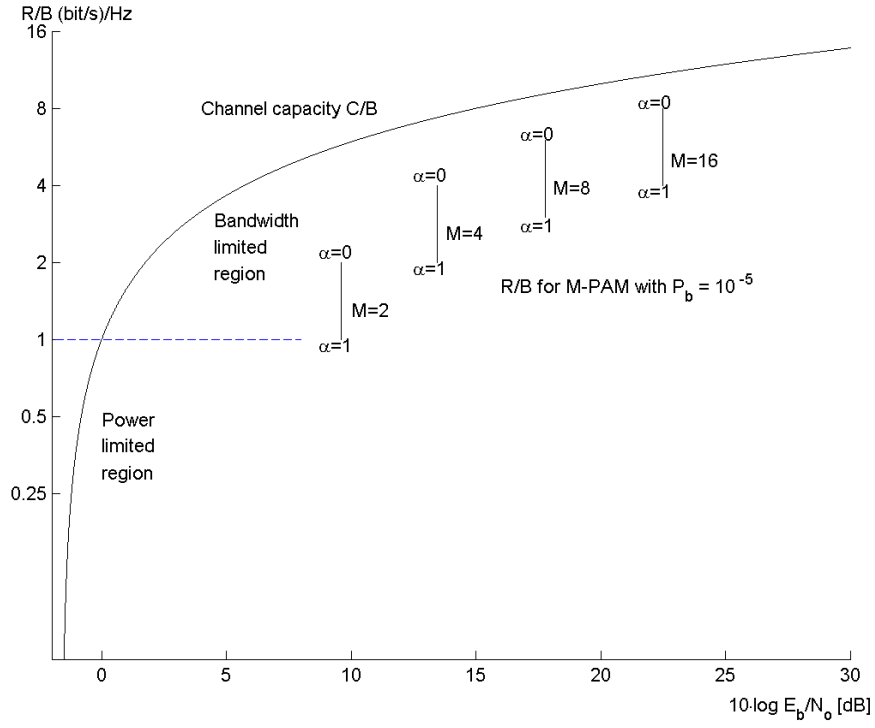

Figure 5.15
Efficiency R/B as a function of signal-to-noise ratio per bit E_b/N_o for $P_e = 10^{-5}$ for different M-PAM systems and different α compared to channel capacity C/B.

$$\frac{R}{B} \le \frac{C}{B} = \log_2\left(1 + \frac{P_{av}}{N_o B}\right) \text{ (bit/s)/Hz} = \log_2\left(1 + \frac{E_b}{N_o}\cdot\frac{C}{B}\right) \text{ (bit/s)/Hz} \qquad (5.22)$$

The capacity for a given signal-to-noise ratio, E_b/N_o, is found by solving the equation for C/B when we have inserted R = C in the expression for P_{av}. The information theory tells that a certain R/B is achievable at a given signal-to-noise ratio with an error probability of zero. The PAM-systems are seen to be about half as good for $P_e = 10^{-5}$ which is not so bad. The figure also shows that if the bandwidth is available it should be used fully. The alternative of increasing the bit rate by increasing M is relatively costly in signal power as we saw before.

The question is now: How do we approach the capacity? More advanced signal constellations and systems may help somewhat, but in general the picture is as in Figure 5.15. Two problems are found:

- The error probability is not zero and many systems would require a more reliable communication than e.g. the 10^{-5} shown. We shall return to this problem below.
- To achieve the capacity in (5.22), the signals need to have a Gaussian distributed amplitude - just like the noise - and we have up to now only used M equiprobable symbols.

In Figure 5.16 we have illustrated the second problem for M = 2 and M = 16.

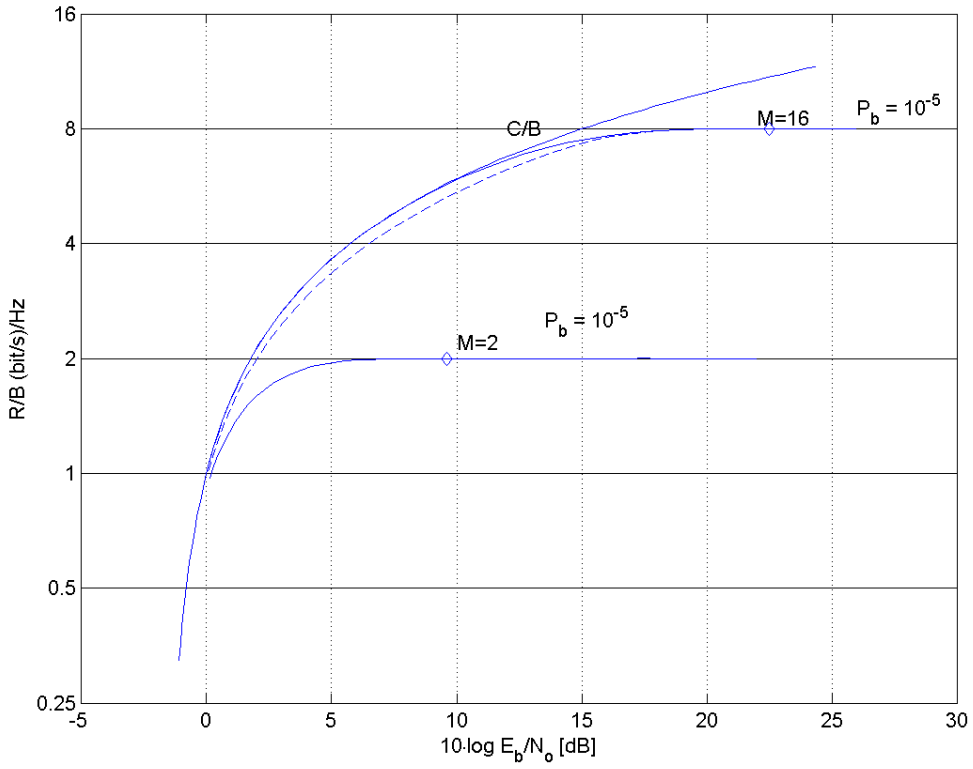

Figure 5.16
The best achievable R/B as a function of signal-to-noise ratio per bit E_b/N_o for two versions of two M-PAM systems (with $\alpha = 0$) compared to the ultimate channel capacity C/B. The two versions have equiprobable symbols (dashed line) and optimally distributed symbols (solid lines)

From this it may be seen that there is a loss of around 1.2 dB for the M = 16 case with equiprobable symbols compared to the case with optimal distribution in the midrange of signal-to-noise ratios. This loss is known as **shaping loss**, [5.5], or shaping gain for the optimally distributed system. The loss is less significant for lower M and it may be proven that it approaches 1.53 dB for $M \to \infty$. To obtain the optimal performance some technique is needed to shape the constellation of signals points towards the Gaussian distribution. This will reduce the bit rate below $\log_2 M/T$ and in practice a higher M has to be applied,

but there is still a gain. For low signal-to-noise ratios, it is seen that M = 2 performs as good as any other M, so binary communication is always used here. For high signal-to-noise ratios, the equiprobable distribution is optimal.

To solve the first problem, the extremely good idea by Shannon was to use **error-correcting coding** which we already met in Section 2.7, (see also [5.6] for details). The idea is to consider sequences of symbols instead of isolated symbols. There are M^n sequences of length n and if we restrict ourselves to use only M^k of these, the sequences may be spread out on the entire M^n space such that even if there are errors in a received sequence, a (maximum likelihood) **decoding** may be done to find the closest of the possible transmitted M^k sequences. With a proper choice of the **error-correcting code**, i.e. the set of M^k sequences, the error rate may be reduced, and it was proven by Shannon that even taking into account the extra power spent on transmitting n symbols instead of k symbols, there exist codes giving a better performance, the so-called **coding gain**. We take this extra power into consideration by modifying the definition (5.14) of E_b to

$$E_b = \frac{E_{av}}{\log_2 M} \cdot \frac{n}{k}$$

i.e. the energy per information bit in the coded system. This modification is inherent in the capacity relation given in (5.22). The error-rate may be proven to decrease exponentially with n if the resulting bit rate $k/n \cdot R = k/n \cdot \log_2 M/T$ is less than the channel capacity. The proof is an existence proof based on the average over all codes. Codes and decoding algorithms coming close to the capacity have been very hard to find. For the power limited region with $E_b/N_o < 1$, i.e. < 0 dB, giving $C/B < 1$, error-correcting codes have been a great success. Most error-correcting codes are binary or they have binary representations so M is almost always a power of 2 (see [5.6] for code constructions) and binary symbols are used in the communication. Even if the codes are binary, there is a considerable gain of using received values with more than two quantization levels ("soft decision" instead of "hard decision") if the decoding algorithm allows this. This has brought us close to the capacity for low signal-to-noise ratios.

Note that we are transmitting n symbols instead of k so the above method would increase the bandwidth, and since we are here mostly dealing with bandwidth limited signals this may not be immediately possible. However, going to a higher M with an appropriate signal

power, an error-correcting code may correct so many errors that we use less power (coding gain) measured per information symbol (k). There are also more direct methods to introduce coding in combination with the signal design without expanding the bandwidth. The most popular method was founded by G. Ungerboeck, [5.7, 5.13].

Example 5.4 -The effect of an error-correcting code

From Figure 5.8 and the corresponding equation we see that E_b/N_o = 9.59 dB is needed to achieve $P_b = 10^{-5}$ for a 2-PAM system. A simple binary code with n = 1005 and k = 670 may actually correct 36 errors with a well-known decoding algorithm, [5.6], and $P_b = 10^{-5}$ after the decoding is achieved at $P_b = 0.020$ before decoding corresponding to a signal-to-noise ratio of 4.97 dB, a coding gain of 4.62 dB. This gain is relevant if power is the problem since it takes into account the extra power spent on transmitting the redundancy for error correction, i.e. (1005-670)/670 = 50% more. R/B is reduced from 2 ($\alpha = 0$) to 1.33 so studying Figure 5.15 it is seen that we move a little closer to capacity. There are better codes and decoding algorithms, especially use of soft decision will improve the performance.

If the bandwidth of the signal is an issue we could use the same code to investigate how to have a coding gain without increasing the bandwidth. From Figure 5.8 we see that E_b/N_o = 13.43 dB is needed to achieve $P_b = 10^{-5}$ for a 4-PAM system which has R/B = 4. The same R/B is obtained with M = 8 and the code above: $R/B = 2(\log_2 8)\cdot(670/1005) = 4$. To achieve $P_b = 10^{-5}$ after the decoding we need again $P_b = 0.020$ before decoding corresponding to a signal-to-noise ratio of 12.37 dB, a coding gain of 1.06 dB. This is moderate since the code selected was not very well suited for the purpose, but a gain was achieved. An optimum system would be 3 - 4 dB better, [5.7], for a signal constellation with 8 points.

5.5 Controlled ISI: Partial Response

The idea of **partial response** (*partiel respons*) is to allow a controlled intersymbol interference in the system and instead of eliminating it, use the known ISI for the estimate of a symbol based on the preceding estimates. It was shown by A. Lender [5.8] in 1962 that it is indeed possible to transmit 2B symbols/s using a bandwidth of B Hz without interference even if the ideal characteristic with $\alpha = 0$ is not used in contrast to the elimination of ISI by the roll-off pulses in Section 5.3. Lender named his technique 'duobinary signaling' and since then it has been generalized to the so-called partial response where duobinary signaling is just one example (cf. Table 5.2). If we assume that we have introduced filtering for the transmission channel in such a way that the intersymbol interference is limited to the four preceding symbols and for a moment disregard the noise, we receive

$$y_k = f_0 a_k + f_1 a_{k-1} + f_2 a_{k-2} + f_3 a_{k-3} + f_4 a_{k-4}.$$

The a-sequence has to be initialized, e.g. as $a_{-4} = a_{-3} = a_{-2} = a_{-1} = 0$. At the reception we have

$$\hat{a}_k = \frac{\hat{y}_k - (f_1 \hat{a}_{k-1} + f_2 \hat{a}_{k-2} + f_3 \hat{a}_{k-3} + f_4 \hat{a}_{k-4})}{f_0}$$

where the hats are added to indicate estimated symbols which may have changed due to noise. Notice that we changed the terminology of the ISI from x_k in the previous section to f_k here. This is because we want simple coefficients (integers) in the following since the precoding to be introduced later would otherwise be difficult. In a real system some scaling is done, as the c in Section 5.3. As we shall see below, the characteristic is bandlimited to $|\omega| \leq \pi/T$ just as for (5.18), but in contrast to that, it is realizable.

Another technical advantage is that the total characteristic may be made closer to the physical characteristic of the channel making it more easy to provide the necessary filters (equalizers) to give the correct total characteristic. Thus partial response is a way of matching the transmitted signal to the channel transfer function. This may be done in other ways. We saw pulse shaping in Section 3.5 and we shall see line coding in Section 5.8. The treatment of partial response here is found natural in connection with the previous section on ISI.

As an example, let us treat the original **duobinary** (*duobinær*) signaling, also known as partial response Class 1. We have

$y_k = a_k + a_{k-1}$, and at the receiver it is easy to find $\hat{a}_k = \hat{y}_k - \hat{a}_{k-1}$.

If we assume that a symbol a_k could be transmitted through the low-pass characteristic for (5.18), $X_{Ny}(\omega) = T$ for $|\omega| \le \pi/T$ (with c = 1), the characteristic for the previous symbols could be found using the delay relation for Fourier transforms, and thus we get for the total characteristic

$$\begin{aligned} X(\omega) &= f_0 X_{Ny}(\omega) + f_1 X_{Ny}(\omega) e^{-j\omega T} \\ &= T\cdot(1 + e^{-j\omega T}) = T\cdot 2\cos\left(\frac{\omega T}{2}\right)\cdot e^{-j\omega T/2} \qquad \text{for } |\omega| \le \frac{\pi}{T} \end{aligned}$$

The amplitude part of this function is shown at Figure 5.18. The last factor in $X(\omega)$ gives a delay of T/2. From (5.18) we get the impulse response

$$x(t) = f_0 \frac{\sin(\pi t/T)}{\pi t/T} + f_1 \frac{\sin(\pi(t-T)/T)}{\pi(t-T)/T} = T^2 \frac{\sin(\pi t/T)}{\pi t(T-t)}$$

or expressed with t as the offset from the delay of T/2:

$$x(t+\frac{T}{2}) = \frac{4T^2}{\pi} \frac{\cos(\pi t/T)}{T^2-4t^2}$$

which clearly gives ISI as shown with the two dotted lines at Figure 5.18.

The total characterstic, $X(\omega)$, still has to be divided between sender and receiver as shown below:

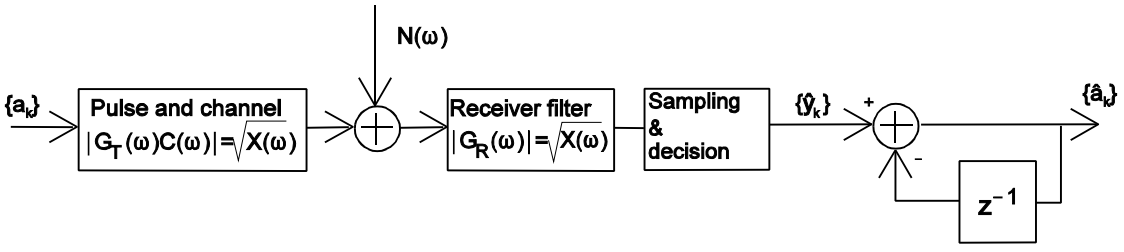

Figure 5.17
Duobinary partial response system (Class 1).

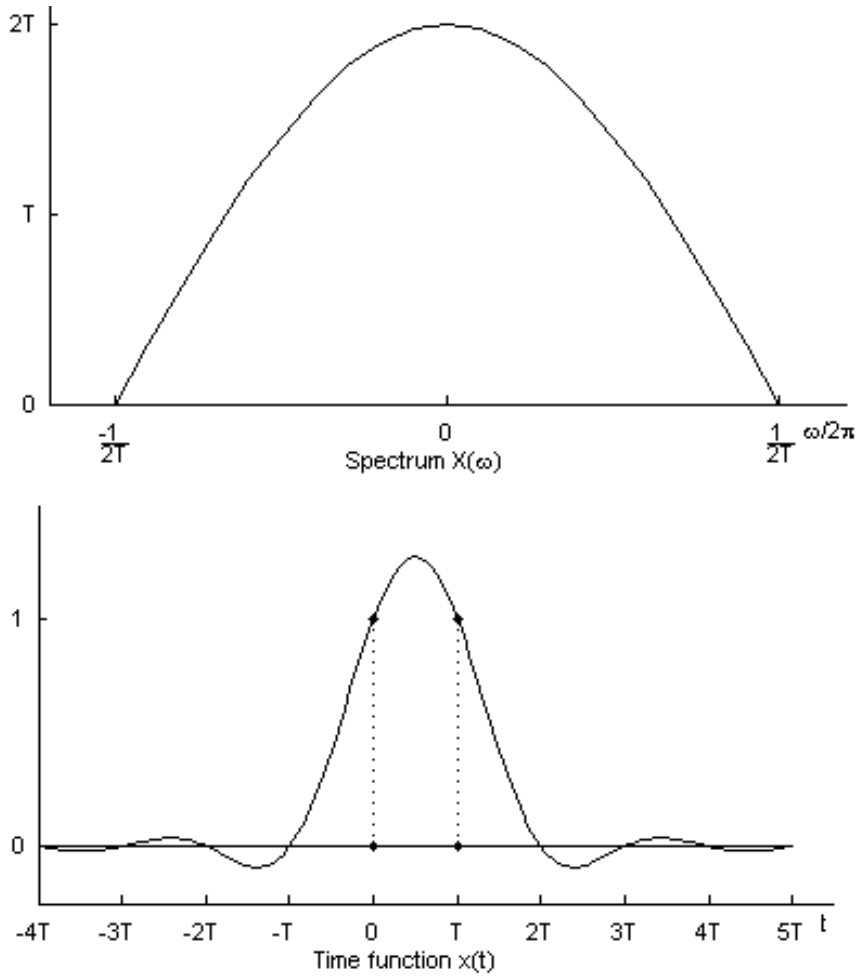

Figure 5.18
Amplitude spectrum X(ω) and time function x(t)
for partial response, Class 1 (duobinary).

Processing the received symbols for a Class 1 system as $\hat{a}_k = \hat{y}_k - \hat{a}_{k-1}$ allows us to use the known intersymbol interference to suppress its effect without increasing the bandwidth used, 1/(2T) Hz. But there is a cost, since $y_k = a_k + a_{k-1}$ means that more y-levels are found, e.g. −2, 0, and +2 with probabilities 1/4, 1/2, and 1/4 if $a_k \in \{-1, +1\}$ with equiprobable symbols. For a constant signal power, this means an inferior error probability as described in Section 5.2.4. A precise calculation shows that the signal-to-noise ratio has to be increased with 2.1 dB to obtain the same bit error probability (assuming use of precoding to be explained later).

Partial response has become a developed technique. Table 5.2 gives several examples known as classes. As an example, Class 4 is much used on telephony channels since X(ω)

is similar to the transfer function of such a channel where transformers limit the lower frequency range and the loss of the subscriber line the upper range. The filters needed to make the precise transfer function are then rather easy to build. The channel is said to be equalized to Class 4 by insertion of these filters. Equalization will be a subject of Section 5.6.

In Table 5.2 we use a general description of partial response with use of the z-transform and the so-called **system polynomial**

$$F(z) = f_0 + f_1 z^{-1} + f_2 z^{-2} + f_3 z^{-3} + f_4 z^{-4} \tag{5.23}$$

which is used to give

$$y_k = f_0 a_k + f_1 a_{k-1} + f_2 a_{k-2} + f_3 a_{k-3} + f_4 a_{k-4}.$$

System polynomial F(z) and name(s)	System transfer function X(ω) (excl. $e^{-j\omega \cdot delay}$) for $\lvert\omega\rvert \le \pi/T$	Impulse response x(t) (excl. delay)	# of levels
$1 + z^{-1}$ duobinary class 1	$2T\cos\frac{\omega T}{2}$	$\frac{4T^2}{\pi}\frac{\cos \pi t/T}{T^2 - 4t^2}$	2M − 1
$1 - z^{-1}$ dicode	$j2T\sin\frac{\omega T}{2}$	$\frac{8Tt}{\pi}\frac{\cos \pi t/T}{4t^2 - T^2}$	2M − 1
$1 - z^{-2}$ modified duobinary class 4	$j2T\sin\omega T$	$\frac{2T^2}{\pi}\frac{\sin \pi t/T}{t^2 - T^2}$	2M − 1
$1 + 2\,z^{-1} + z^{-2}$ class 2	$4T\cos^2\frac{\omega T}{2}$	$\frac{2T^3}{\pi t}\frac{\sin \pi t/T}{T^2 - t^2}$	4M − 3
$1 - 2\,z^{-2} + z^{-4}$ class 5	$-4T\sin^2\omega T$	$\frac{8T^3}{\pi t}\frac{\sin \pi t/T}{t^2 - 4T^2}$	4M − 3
$(1 - z^{-2})(1 + z^{-1})$ extended class 4	$j4T\cos\frac{\omega T}{2}\cdot\sin\omega T$	$-\frac{64T^3 t}{\pi}\frac{\cos \pi t/T}{(4t^2 - 9T^2)(4t^2 - T^2)}$	4M − 3
$(1 - z^{-2})(1 - z^{-1})$	$-4T\sin\frac{\omega T}{2}\cdot\sin\omega T$	$\frac{16T^2}{\pi}\frac{(4t^2 - 3T^2)\cos \pi t/T}{(4t^2 - 9T^2)(4t^2 - T^2)}$	4M − 3
$2 + z^{-1} - z^{-2}$ class 3	$T + T\cos\omega T + jT\sin\omega T$	$\frac{T^2}{\pi t}\frac{3t - T}{t^2 - T^2}\sin \pi t/T$	4M − 3
$2 - z^{-2} - z^{-4}$	$-T + T\cos 2\omega T + jT\sin 2\omega T$	$\frac{2T^2}{\pi t}\frac{2T - 3t}{t^2 - 4T^2}\sin \pi t/T$	4M − 3

Table 5.2
System polynomials, their transfer functions and impulse responses. From [5.9].

This description is also used in [5.9] where a general and thorough description is found. Note that the system characteristic is found as $X(\omega) = TF(e^{j\omega T})$ for $|\omega| \le \pi/T$ just as we did for the Class 1 above. Notice that the total characteristic $X(\omega)$ has to be split between transmitter and receiver as in (5.20) (and Figure 5.17), i.e. the squareroot is allocated to the receiver.

The z-transforms of $\{a_k\}$ and $\{y_k\}$ are:

$$A(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + ...$$
$$Y(z) = y_0 + y_1 z^{-1} + y_2 z^{-2} + ...$$

and we get

$$\mathbf{Y(z) = A(z) \cdot F(z)} \tag{5.24}$$

At the reception we have in z-transform notation

$$\hat{\mathbf{A}}(\mathbf{z}) = \frac{\hat{\mathbf{Y}}(\mathbf{z})}{\mathbf{F(z)}} \tag{5.25}$$

Example 5.5 - Partial response and noise - error propagation

In this example we consider a Class 2 partial response system, i.e.

$$F(z) = 1 + 2z^{-1} + z^{-2}.$$

Let us assume that we have binary information sequence, $b_k \in \{0,1\}$, e.g.

b_k 0 1 1 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0

This sequence is converted into bipolar symbols a_k in the usual way, $a_k = 2b_k - 1$. Including two initialization values a_{-1} and a_{-2} we have

a_k 0 0 | −1 1 1 1 −1 1 −1 1 1 −1 −1 −1 −1 1 1 1 −1 −1 −1

With F(z) we obtain:

y_k −1 −1 2 4 2 0 0 0 2 2 −2 −4 −4 −2 2 4 2 −2 −4

The alphabet of $\{y_k\}$ is seen to be $\{-4, -2, 0, 2, 4\}$ except for the first two values from the initialization of a_{-1} and a_{-2} with zeros. This alphabet size is consistent with Table 5.2, since $4 \cdot 2 - 3 = 5$. The received sequence $\{\hat{y}_k\} = \{y_k\}$ is easily decoded as $\hat{a}_k = \hat{y}_k - 2\hat{a}_{k-1} - \hat{a}_{k-2}$ (try to put it as a division by F(z)) to the original a_k and from the bipolar conversion we get the information

$$\hat{b}_k = \frac{\hat{a}_k + 1}{2}$$

which is also the transmitted b_k. But sometimes wrong $\hat{y}_k$ decisions are made by the receiver. Let an error be at the underlined symbol which is decided to be 2. We decode again $\hat{a}_k = \hat{y}_k - 2\hat{a}_{k-1} - \hat{a}_{k-2}$:

$\hat{a}_k$ 0 0| −1 1 1 1 −1 1 1 −3 7 −9 9 −13 13 −15 19 −19 ...

The error not only results in one erroneous symbol, but a whole sequence. We name this phenomenon **error propagation** (*fejludbredelse*). It may be detected, since we get outside the a-alphabet $\{-1, 1\}$, but we have no way to correct it. The error is of course a result of doing division after the error has occurred.

The error propagation from Example 5.5 may be omitted by doing the division with F(z) before the sequence is put into the system. This would create new symbol values, so instead we do the division modulo M, and assume that $b_k \in \{0, ..., M-1\}$ is the alphabet we shall transmit. The division by F(z) is known as **precoding** (*forkodning*):

$$P(z) = \frac{B(z)}{F(z)} \text{ modulo M.} \tag{5.26}$$

The p-sequence is then within the same alphabet $\{0, ..., M-1\}$ and we convert into a symmetric alphabet by

$$a_k = 2p_k - (M - 1), \qquad a_k \in \{-(M-1), -(M-3), ..., M-3, M-1\}$$

and the a_k are transmitted through the system as shown in Figure 5.19.

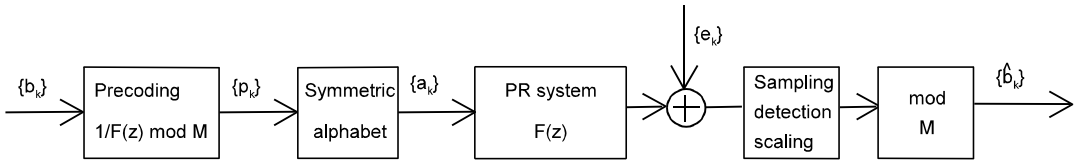

Figure 5.19
Partial response system with precoding.
Note that the noise is shown here only to remember the noise.
The real noise is somewhere inside the PR system.

The received sequence is $\{\hat{y}_k\} = \{y_k\} + \{e_k\}$ and it may be calculated as

$$\hat{y}_k = y_k + e_k = \sum_{m=0}^{4} f_m(2p_{k-m}-(M-1)) + e_k = 2\sum_{m=0}^{4} f_m p_{k-m} - (M-1)\sum_{m=0}^{4} f_m + e_k$$

The original sequence b_k may be found from the p_k - sequence, and by calculating modulo M we get:

$$\frac{\hat{y}_k + (M-1)\sum_{m=0}^{4} f_m}{2} \bmod M = \left(\sum_{m=0}^{4} f_m p_{k-m}\right) \bmod M + \frac{e_k}{2} \bmod M = b_k + \frac{e_k}{2} \bmod M$$

The b_k is a result of (5.26) describing how b_k was formed. Thus the errors only affect the symbols where the decisions $\hat{y}_k$ are wrong.

Example 5.6 - Precoding

We continue from Example 5.5 with $b_k \in \{0,1\}$, i.e. M = 2, and $F(z) = 1 + 2z^{-1} + z^{-2}$. Let the information sequence be

b_k 0 0 | 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 0 1 1 0

Dividing with F(z) modulo 2 is the same as dividing with $1 + z^{-2}$, because 2 modulo 2 = 0, i.e we find $b_k = p_k + p_{k-2}$ modulo 2 or $p_k = b_k - p_{k-2} = b_k \oplus p_{k-2}$, where $\oplus$ means addition modulo 2 (exor). We get the p-sequence:

p_k 0 0 | 0 1 1 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0

which was used as b_k in Example 5.5. The y-sequence becomes the same and letting the error occur at the same symbol we receive

$\hat{y}_k$ −1 −1 2 4 2 0 2 0 2 2 −2 −4 −4 −2 2 4 2 −2 −4

Now we find

$$\hat{b}_k = \frac{\hat{y}_k + 4}{2} \bmod 2$$

which (after the two initial symbols) gives

x x 1 0 1 0 1 0 1 1 1 0 0 1 1 0 1 1 0

Thus the error only gave a single error in the b-sequence - as expected.

In some cases, e.g. optical communication, the possibility of a symmetric alphabet does not exist, so the precoded p-sequence is transmitted directly, i.e. alphabet {0, ..., M−1}. The received sequence is $\{\hat{y}_k\} = \{y_k\} + \{e_k\}$ and it may be calculated as

$$\hat{y}_k = y_k + e_k = \sum_{m=0}^{4} f_m p_{k-m} + e_k$$

The original sequence b_k may be again be found by calculating modulo M:

$$\hat{y}_k \bmod M = \left(\sum_{m=0}^{4} f_m p_{k-m} \right) \bmod M + e_k \bmod M = b_k + e_k \bmod M$$

so the errors here also only affect the symbols where the decisions $\hat{y}_k$ are wrong.

5.6 Equalization

As described in Sections 5.1 - 5.3, the transmission channel affects in the received pulse, and the matched filter has to take the channel into account, too. This may not be convenient since the filters in the transmitter, $g_T(t)$, and in the receiver, $g_R(t)$, are chosen as fixed filters, which are not adapted to the particular transmission channel in use. To suppress the channel effects we add one more operation in the transmission system, an **equalizer** (*udligner*). In the frequency domain, the equalizer with transfer function $E(\omega)$ is ideally chosen such that the total transfer function

$$G_T(\omega)C(\omega)G_R(\omega)E(\omega)$$

is independent of $C(\omega)$, i.e. $E(\omega) = 1/C(\omega)$, for the frequency range of interest. Using the equalizer, the receiver filter may then be matched to the transmitter filter only, i.e. $G_R(\omega) = G_T^*(\omega)$ and chosen to avoid intersymbol interference (the Nyquist criterion). The sequence of the operations is normally as shown above. The expression shown is only to demonstrate the effect of an equalizer, but there may be several problems. A large drawback with a $1/C(\omega)$-equalizer is that it may amplify the noise so much that the performance is reduced. Not all equalizers are linear as here, since there may be great advantages of a non-linear equalizer. A very short introduction to these are given in Section 5.6.3. We shall discuss equalizers below after a small detour to practical communication engineering.

5.6.1 Eye diagram

There are many situations where a visual presentation of system performance would be of interest. Since baseband systems consist of repetition of transmissions of the same pulse with different amplitudes, a possible setup could be to connect the signal to an oscilloscope

with the horizontal sweep rate set to 1/T or a fraction of 1/T. Normally it is the signal after the matched filter that is studied here. The resulting oscilloscope display is called an **eye diagram** (*øjediagram*) or **eye pattern** because the memory of the oscilloscope screen shows many repetitions of the pulse with different amplitudes overlaying each other, and a period of this often looks like a human eye, see the example in Figure 5.20 for a signal with two amplitudes.

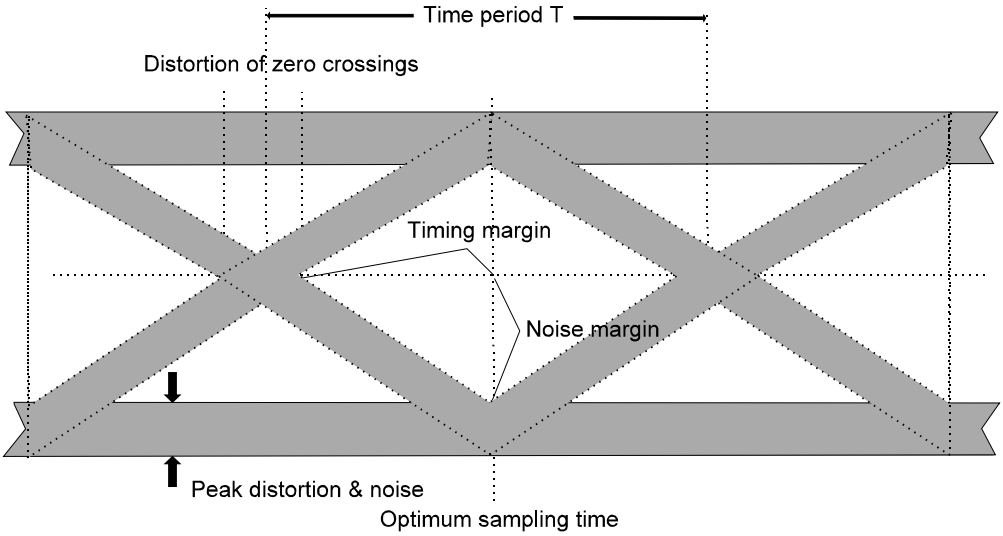

Figure 5.20
Eye diagram with different characteristics for a binary signal.

It is clearly seen where to sample the signal at the widest opening of the eye, and the effect of ISI may also be seen since the different instances of the pulse distort each other, which results in a closing of the eye when the traces are overlaid. The closing gives a smaller margin for the noise which is seen together with the ISI as a broadening of the traces of the pulses. Characteristics relevant to the synchronization (Section 5.7) are also shown in the figure.

5.6.2 Linear equalization

As described in Section 5.3, the signal after the filter $G_R(\omega)$ may exhibit intersymbol interference and the purpose of the equalizer is to remove/suppress this interference. The ultimate goal of a communication system is to minimize the error probability of which the intersymbol interference is only one contributor, and suppressing the interference may increase the remaining noise resulting in an inferior performance. Such a situation is often

found if the transfer function C(ω) contains spectral nulls (or very low transmission at certain frequencies) since then E(ω) may amplify the noise from the outside world. We shall not go further into this in the present section, but it is the main reason for using non-linear equalizers. As in Section 5.3, the sampled values after the receiver filter are

$$f_k = \sum_{n=-\infty}^{\infty} a_n x_{k-n} + n_k$$

where x_{k-n} is the total impulse response which shows intersymbol interference. The symbol that we want to detect at time kT, a_k, is influenced by the neighbor symbols through the weights $x_{k-n} = x((k-n)T)$, and then comes the added noise, n_k.

Since the impulse response may be assumed fixed and known, we may without loss of generality describe how we would equalize for the symbol a_0 using the signal values around that time, i.e. f_{-L}, f_{-L+1}, ..., f_0, ..., f_{L-1}, f_L. The simplest equalizer that may be envisaged is a FIR filter with output

$$\phi = \sum_{k=-L}^{L} h_k f_{-k} \tag{5.27}$$

designed in such a way that at its output (at the appropriate time) we would like to see a_0. Thus the action of the equalizer may be found by insertion of f_{-k} into (5.27):

$$\phi = \sum_{k=-L}^{L} h_k \left(\sum_{\ell=-\infty}^{\infty} a_\ell x_{-k-\ell} + n_{-k} \right) = \sum_{\ell=-\infty}^{\infty} a_\ell \sum_{k=-L}^{L} h_k x_{-\ell-k} + \tilde{n} = a_0 + r_0 + \tilde{n}$$

i.e. the intersymbol interference is forced to zero for some ℓ around $\ell = 0$. However, in most cases this will leave some remaining interference, collected in the term r_0. The noise is filtered to $\tilde{n}$.

Example 5.7 - A simple zero-forcing equalizer

Consider a simple channel where

$$f_k = a_k + c a_{k-1} + n_k, \quad \text{i.e. } x_0 = 1, \ x_1 = c, \text{ and } x_n = 0 \text{ for other } n$$

First we consider a small equalizer with L = 1:

$$\begin{aligned} \phi &= h_{-1} f_{-(-1)} + h_0 f_0 + h_1 f_{-1} \\ &= h_{-1}(a_1 + c a_0 + n_1) + h_0(a_0 + c a_{-1} + n_0) + h_1(a_{-1} + c a_{-2} + n_{-1}) \\ &= (c h_{-1} + h_0) a_0 + h_{-1} a_1 + (c h_0 + h_1) a_{-1} + c h_1 a_{-2} + \tilde{n} \end{aligned}$$

To give us a_0 we get $h_{-1} = 0$ and $h_0 = 1$ from the first parenthesis. The interference from a_1 is then also removed. Selecting $h_1 = -c$ forces the interference from a_{-1} to disappear and we are left with some residual interference from a_{-2}:

$$\phi = f_0 - c f_{-1} = a_0 - c^2 a_{-2} + \tilde{n}$$

A little more formally stated, the zero-forcing coefficients are found from solving

$$\begin{bmatrix} 1 & 0 & 0 \\ c & 1 & 0 \\ 0 & c & 1 \end{bmatrix} \begin{bmatrix} h_{-1} \\ h_0 \\ h_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Extending the equalizer to L=2, we get

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ c & 1 & 0 & 0 & 0 \\ 0 & c & 1 & 0 & 0 \\ 0 & 0 & c & 1 & 0 \\ 0 & 0 & 0 & c & 1 \end{bmatrix} \begin{bmatrix} h_{-2} \\ h_{-1} \\ h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The solution is $\mathbf{h} = (0, 0, 1, -c, c^2)$ leaving residual interference $c^3 a_{-3}$. In general for any L, the solution to the equation system is $h_k = (-c)^k$ for $k = 0, ..., L$, and the residual interference for this system becomes $-(-c)^{L+1} a_{-L-1}$. The transfer function for the filter becomes

$$H(\omega) = \sum_{k=0}^{L} h_k e^{-j\omega kT} = \sum_{k=0}^{L} (-c)^k e^{-j\omega kT} = \sum_{k=0}^{L} (-c e^{-j\omega T})^k$$

Letting $L \to \infty$, we get

$$H(\omega) = \sum_{k=0}^{\infty} h_k e^{-j\omega kT} = \sum_{k=0}^{\infty} (-c e^{-j\omega T})^k = \frac{1}{1 + c e^{-j\omega T}}$$

which is just the inverse of the channel transfer function found from $f_n = a_n + c a_{n-1}$, as we expected in the introduction to this section.

The zero-forcing filter above operates with the same sampling rate as the symbol rate, i.e. 1/T, which means that channel transfer function components at frequencies above 1/(2T)

Hz are only indirectly equalized trough their aliases folded down into the basic frequency range. Such an equalizer is known as a **symbol-spaced equalizer**. However, in some cases, and especially when adaptive methods are used for finding the coefficients, this may not be convenient, and the equalizer uses also input in between the symbol samples. Such an equalizer is known as a **fractionally spaced equalizer**. Usually the bandwidth of interest is 2/(2T) Hz, e.g. cosine roll-off with $\alpha = 1$, resulting in an oversampling factor of 2 for the equalizer. We introduce T_E as the time between equalizer samples, and normally T_E is a multiple of the sampling time for the oversampled receiver filter. We reformulate (5.27) to take into account all equalizer samples

$$\phi = \sum_{k=-L}^{L} h_k f(-kT_E) \tag{5.28}$$

where

$$f(t) = \sum_{n=-\infty}^{\infty} a_n x(t-nT) + n(t)$$

Doing the same calculation as above for the symbol-spaced version, we may now formulate the basic equation for an **zero-forcing equalizer**

$$\sum_{k=-L}^{L} h_k x(jT-kT_E) = \begin{cases} 1 & \text{for } j = 0 \\ 0 & \text{for } j = \pm 1, \pm 2, \ldots, \pm L \end{cases} \tag{5.29}$$

which gives 2L+1 equations for the 2L+1 unknown coefficients h_k. L has to be chosen large enough for the residual intersymbol interference to be of no importance.

Example 5.8 - A fractionally spaced zero-forcing equalizer

An example of a pulse is introduced later in Equation (7.28),

$$x(t) = \frac{1}{1+(2t/T)^2}$$

and a small equalizer (L=2) is found from the equation below. Remember that the rows in the matrix correspond to the symbol-spaced samples and that the columns are the fractionally spaced samples where we have chosen $T_E = T/2$:

$$\begin{bmatrix} \frac{1}{5} & \frac{1}{10} & \frac{1}{17} & \frac{1}{26} & \frac{1}{37} \\ 1 & \frac{1}{2} & \frac{1}{5} & \frac{1}{10} & \frac{1}{17} \\ \frac{1}{5} & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{5} \\ \frac{1}{17} & \frac{1}{10} & \frac{1}{5} & \frac{1}{2} & 1 \\ \frac{1}{37} & \frac{1}{26} & \frac{1}{17} & \frac{1}{10} & \frac{1}{5} \end{bmatrix} \begin{bmatrix} h_{-2} \\ h_{-1} \\ h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The solution is $\mathbf{h} = (-2.2, 4.9, -3, 4.9, -2.2)$, but some intersymbol interference is still found since for the illustration here we have chosen L too small. For this particular x(t) the small L makes the coefficients unrealistic high which would cause a heavy noise influence.

A drawback of the zero-forcing equalizer is that it does not take into account the noise found, and it is expected that a better result may be obtained by trying to reduce the mean-square error as much as possible for the samples where the noise has two contributions: the intersymbol interference and the outside noise.

If the equalizer is still given by (5.28) we would like to minimize

$$E[(\phi - a_0)^2]$$

and we have solved this problem before in Section 4.5, where the solution was given by the Wiener-Hopf equation, (4.14). The version (4.18) for a stationary signal cannot be used here, since in general the output from the matched filter, f(t), is cyclostationary as shown in Section 3.5.

For the derivation of the equation for this case, we assume that $\{a_k\}$ is a stationary process with independent variables and zero mean, i.e. $E[a_k a_\ell] = R_A(0)$ for $k = \ell$, and 0 for $k \neq \ell$. The noise n(t) also has zero mean and autocorrelation $R_N(k)$ and it is independent of $\{a_k\}$. Again, we shall without loss of generality derive the equation for estimation of a_0 on basis of the output $f(0 \cdot T - kT_E)$ (for $k = -L, ..., L$) from the receiver filter. With these assumptions, (4.14) becomes

$$\sum_{k=-L}^{L} h_k E[f(-jT_E)f(-kT_E)] = E[a_0 f(-jT_E)] \quad \text{for } j = 0,\pm1,\pm2,...,\pm L$$

The expected values at the left hand side may be evaluated as

$$\begin{aligned} &E[f(-jT_E)f(-kT_E)] \\ &\quad = E\left[\left(\sum_{n=-\infty}^{\infty} a_n x(-jT_E-nT)+n(-jT_E)\right)\cdot\left(\sum_{\ell=-\infty}^{\infty} a_\ell x(-kT_E-\ell T)+n(-kT_E)\right)\right] \\ &\quad = \sum_{n=-\infty}^{\infty}\sum_{\ell=-\infty}^{\infty} E[a_n a_\ell]x(-jT_E-nT)x(-kT_E-\ell T)+E[n(-jT_E)n(-kT_E)] \\ &\quad = R_A(0)\sum_{n=-\infty}^{\infty} x(-jT_E-nT)x(-kT_E-nT) + R_N(j-k) \end{aligned}$$

The right hand side may be evaluated as

$$\begin{aligned} E[a_0 f(-jT_E)] &= E\left[a_0\left(\sum_{n=-\infty}^{\infty} a_n x(-jT_E-nT)+n(-jT_E)\right)\right] \\ &= \sum_{n=-\infty}^{\infty} E[a_0 a_n]x(-jT_E-nT)+E[a_0 n(-jT_E)] \\ &= R_A(0)x(-jT_E) \end{aligned}$$

For the **minimum mean-square error equalizer** the equation then is

$$\begin{aligned} &\sum_{k=-L}^{L} h_k\left(R_A(0)\sum_{n=-\infty}^{\infty} x(-jT_E-nT)x(-kT_E-nT) + R_N(j-k)\right) \\ &\qquad = R_A(0)x(-jT_E) \quad \text{for } j = 0,\pm1,\pm2,...,\pm L \end{aligned} \tag{5.30}$$

The mean-square error is found from (4.15) which in the present context becomes

$$E[(\phi - a_0)^2] = R_A(0)\left(1 - \sum_{k=-L}^{L} h_k x(-kT_E)\right) \tag{5.31}$$

Example 5.9 - A simple minimum mean-square error equalizer

We continue with simple channel from Example 5.7:

$$f_k = a_k + c\,a_{k-1} + n_k, \quad \text{i.e. } x_0 = 1,\ x_1 = c, \text{ and } x_n = 0 \text{ for other } n$$

We assume that $\{a_k\}$ is a stationary process consisting of equally probable ±1 symbols that occur independently which gives $R_A(0) = 1$ and $R_A(n) = 0$ for $n \neq 0$. The noise n(t) also has zero mean, variance σ^2 and independent samples, and it is

independent of $\{a_k\}$. We consider an equalizer with L = 2, and (5.30) with $T_E = T$ gives

$$\begin{bmatrix} A & c & 0 & 0 & 0 \\ c & A & c & 0 & 0 \\ 0 & c & A & c & 0 \\ 0 & 0 & c & A & c \\ 0 & 0 & 0 & c & A \end{bmatrix} \begin{bmatrix} h_{-2} \\ h_{-1} \\ h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} 0 \\ c \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

where $A = 1 + c^2 + \sigma^2$, the autocorrelation of x calculated at time 0 plus the variance of the noise. The solution with symbolic constants is rather complex, but for c = 0.5 and $\sigma^2 = 0.1$ we get **h** = (−0.023, 0.063, 0.853, −0.366, 0.136). Note that a small contribution from the samples before our estimated symbol is also found in contrast to the zero-forcing equalizer in Example 5.7, where **h** = (0, 0, 1, −0.5, 0.25). This is true even without noise, but it may be shown by insertion that without noise the solution (5.29) for the zero-forcing equalizer fits into (5.30) for infinite L such that the transfer function for the filter becomes the inverse of the channel transfer function just as for the zero-forcing equalizer. The resulting error for the equalizer is calculated by (5.31) to 0.116.

Example 5.10 - A fractionally spaced minimum mean-square error equalizer

We continue with the pulse from Example 5.8, and we further assume that the data sequence $\{a_k\}$ consists of equally probable ±1 symbols that occur independently which gives $R_A(0) = 1$. We add white Gaussian noise with variance 0.1, and we still want to design a fractionally spaced equalizer with $T_E = T/2$.

The infinite sum in (5.30) has to be truncated in some way, and we use the same truncation as in Example 5.8

x = (1/37, 1/26, 1/17, 1/10, 1/5, 1/2, 1, 1/2, 1/5, 1/10, 1/17, 1/26, 1/37)

Another problem with the sum in (5.30) is that it is not a simple autocorrelation of **x** since only every other value is used ($T_E = T/2$). Three correlations (or convolutions) may be calculated to assist the calculations

$\mathbf{c}_{even\_even}$ = (1/37, 1/17, 1/5, 1, 1/5, 1/17, 1/37)*
(1/37, 1/17, 1/5, 1, 1/5, 1/17, 1/37)

$\mathbf{c}_{odd\_odd}$ = (1/26, 1/10, 1/2, 1/2, 1/10, 1/26)*(1/26, 1/10, 1/2, 1/2, 1/10, 1/26)

$\mathbf{c}_{odd\_even}$ = (1/26, 1/10, 1/2, 1/2, 1/10, 1/26)*(1/37, 1/17, 1/5, 1, 1/5, 1/17, 1/37)

which are to be used depending on the difference j − k, and k even or odd.

A small equalizer (L = 2) is then found from the equation below. For the first part of (5.30) we use the shorthand notation c_{ee}, c_{oo}, and c_{oe} for the convolutions above and a hopefully understandable indexing of these:

$$\begin{bmatrix} c_{ee}(0)+R_N(0) & c_{oe}(-1) & c_{ee}(-2) & c_{oe}(-3) & c_{ee}(-4) \\ c_{oe}(1) & c_{oo}(0)+R_N(0) & c_{oe}(-1) & c_{oo}(-2) & c_{oe}(-3) \\ c_{ee}(2) & c_{oe}(1) & c_{ee}(0)+R_N(0) & c_{oe}(-1) & c_{ee}(-2) \\ c_{oe}(3) & c_{oo}(2) & c_{oe}(1) & c_{oo}(0)+R_N(0) & c_{oe}(-1) \\ c_{ee}(4) & c_{oe}(3) & c_{ee}(2) & c_{oe}(1) & c_{ee}(0)+R_N(0) \end{bmatrix} \begin{bmatrix} h_{-2} \\ h_{-1} \\ h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} x(2T_E) \\ x(T_E) \\ x(0) \\ x(-T_E) \\ x(-2T_E) \end{bmatrix}$$

Inserting the values from $\mathbf{c}_{even\_even}$, $\mathbf{c}_{odd\_odd}$, and $\mathbf{c}_{odd\_even}$ above we obtain the solution $\mathbf{h}$ = (−0.2415, 0.2267, 0.7748, 0.2267,−0.2415). Some intersymbol interference is still found since for the illustration here we have chosen L too small. We may calculate the mean-square error from (5.31) as

$$E[(\phi - a_0)^2] = R_A(0)\left(1 - \sum_{k=-L}^{L} h_k x(-kT_E)\right)$$

$$= 1 - \mathbf{h}\cdot(0.2, 0.5, 1, 0.5, 0.2) = 0.095$$

The variance of the noise was 0.1 before the **h**-filter, i.e. 0.082 after the filter, so there is still some contribution from the remaining intersymbol interference.

In the example we knew everything about the channel, but it is of course not the case in a real environment. Thus we need adaptive methods that keep the solution adapted to the current situation. We have treated such a method in Example 4.6, the LMS algorithm. For the algorithm to work, it is necessary to compute an error in each step, and at first hand this may seem impossible, since we do not know the symbols transmitted. This may be solved by starting the equalization with a training period where a known (pseudo-random) sequence is transmitted, and the known sequence is used in the calculation of the error. After this initial period it is assumed that the equalization is close to the correct performan-

ce, such that very little intersymbol interference is found, and the decision device following the equalizer makes decisions with very few errors. The equalizer algorithm then changes to a decision-directed mode, where the error in the equalization is calculated against the symbols which are output from the decision. In such a process it is important that the transmitted symbols are fairly random, i.e. scrambling is usually required. In some applications transmission of a training sequence would be awkward, and there exist methods for equalization working directly on the data. This form of synchronization is known as self-recovering or **blind equalization**, [5.13].

5.6.3 Introduction to non-linear equalization

As mentioned in the previous section, situations exist where the linear equalizer harms the probability of error instead of improving it. Then other equalizer structures are needed. One is the so-called **decision-feedback equalizer** which may be seen in Figure 5.26. It consists of a feedforward linear equalizer which reshapes the pulse such that the maximum is found in the beginning and the intersymbol interference is mostly found after the present symbol assumed to be at the maximum. The effects on the following symbols may then be suppressed by using the decision for the symbol from which the name follows. Of course there is a possibility of wrong decisions that influence the performance, but it works reasonably well up to error rates of 1%.

The optimal solution is the so-called **maximum-likelihood sequence detector** which determines the sequence $\{\hat{a}_k\}$ the maximizes the probability of the received sequence of values $\{f_m\}$. One may also combine the equalizer with an error-correcting code and thereby obtain a better performance in a so-called turbo equalizer, [5.10].

In this course we shall not go into details of the non-linear equalizers, and the interested reader may consult [5.3, 5.11, 5.12, 5.13].

5.7 Symbol synchronization

In our treatment of receivers we have not discussed how sampling is performed at the correct moments but this may be crucial to the performance of the systems. Alignment of the timing in receiver and transmitter is called **synchronization** (*synkronisering*).

5.7.1 The synchronization problem

One may think of different ways to obtain a synchronization between transmitter and receiver. However some of these ways cannot be used: For practical reasons the synchronization cannot be obtained be tapping some universal time reference or by transmitting a timing signal through a separate transmission channel. Thus the only remaining possibility is in some way to bury the synchronization signal in the transmitted signal which is received as:

$$v(t) = \sum_{k=-\infty}^{\infty} a_k g(t - kT - \tau)$$

From this signal the receiver should be able to determine the timing (T or frequency 1/T) and the phase (τ). This task is named **symbol synchronization** (*symbolsynkronisering*) or, if only binary symbols are involved, **bit synchronization** (*bitsynkronisering*).

Many communication systems also require another synchronization process which in general may be determined by the information contained in the transmitted symbols. If these have some kind of structure, as e.g. one or more PCM-signals (Section 4.4), it is required to obtain **frame synchronization** (*rammesynkronisering*) in order to provide the correct meaning of the received signal. We have introduced the problem of selecting a synchronization word with good correlation properties in Example 2.11, and in Chapter 8 we shall provide examples of practical methods for frame synchronization - in ITU-T terminology named frame alignment - but we shall not go into detail in this course. In the next section a very simple combined bit and frame synchronization (5 - 9 bit frames) is treated.

5.7.2 Asynchronous synchronization

A signal is called asynchronous (*asynkron*) (or in ITU-T-terminology: anisochronous (*anisokron*)) if the important instants (e.g. the optimal sampling points) of the signal appear at time intervals which are not multiples of some fixed duration. For such signals synchronization is obviously difficult since very little about the timing can be inferred from the signal. By the term **asynchronous synchronization** (*asynkron synkronisering*) we shall mean a synchronization method that from some particular property of the signal makes an estimate of the correct phase of the signal, but then proceeds with sampling independently of the timing at the transmitter.

The well known start-stop principle, which has been used for teleprinters for many years and now is known to anyone as the synchronization used for the serial port in PCs, is the mostly used variation of asynchronous synchronization schemes. The transmission has an idle state which is logical 1 (traditionally named “mark”). The receiver detects a shift to 0 (“space”), the so-called startbit. When the startbit is detected, the receiver has to keep the synchronization in the remaining part of the transmitted frame which may contain a character of 5 to 8 bits and optionally an extra parity check bit (cf. Section 2.6). The modulation rate 1/T is assumed known to the receiver with a certain tolerance which requires a certain guard space between two characters, the so-called stopbits. At least one stopbit (1 as the idle state) has to be found but sometimes 1 1/2 or 2 are required for larger tolerance. The receiver locates the midpoint of the startbit by calculation from the (approximately) known length, and then the intention is to sample the next 6 to 11 bits at the midpoints and interpret them as the character, the parity bit and the stopbit(s). Besides symbol synchronization we also obtain a frame synchronization (with very small frames of 5 to 9 bits) The system is not very efficient since the **overhead** from start and stopbits is at least 22%. Thus it is only used for moderate symbol rates where a simple interface is important.

In certain communication systems the transmitted information is found in packages that appear in an asynchronous way. Examples are GSM, DECT, and the information read from a credit card pulled manually through the reader. In such systems each package contains a learning sequence and/or frame synchronisation word from which the transmission rate may be estimated together with the position of the symbols. Thus such systems form a kind of intermediate between the above described asynchronous system and the isochronous systems described below where the transmission is continuous.

5.7.3 Isochronous synchronization

This form of synchronization is named **isochronous** (*isokron*) since the sampling in the receiver is done at the same ("iso") points as in the transmitter. The synchronization is obtained from the received signal without overhead which makes the scheme much more efficient than the asynchronous scheme. It is obvious that the signal is required to be isochronous, i.e. the important instants (e.g. the optimal sampling points) of the signal appear at time intervals that are multiples of some fixed duration

The received signal

$$v(t) = \sum_{k=-\infty}^{\infty} a_k g(t - kT - \tau)$$

must contain components from which the frequency (1/T) and the phase (τ) may be determined. A number of methods exists for extracting these characteristics:

- The optimal method is direct estimation of the parameters T and τ. We have given a short introduction to (linear) estimation in Section 4.5.1 and in [5.12, 5.13, 5.14] the use of estimation for symbol synchronization is described. The value to be minimized is the probability of error. The estimation is often implemented in a purely digital way, e.g. in a signal processor, which makes it difficult to obtain very high symbol rates since it is normally necessary to sample faster than T in order to do the synchronization. The advanced methods are primarily used for poor signal-to-noise ratios as e.g. found in satellite and other radio transmission schemes. It may be shown, [5.14], that under certain assumptions about signal-to-noise ratio, this general method is approximated by some of the methods given below which are then sub-optimal methods.
- Minimum mean square error estimation of the parameters T and τ. This is not so different from the method above since it may be expected that reducing the error of the timing parameters also reduces the probability of symbol errors. However, it seems to be a more direct approach, especially in time continuous schemes. In [5.15] a very thorough treatment of synchronization problems is found.
- Early-late synchronization. A simple method based on the fact that the output from a matched filter should be maximum at the correct sampling point. It is described below. Similar methods are found based on zero crossings of the signal.
- Mueller and Müller synchronization. A similar method based on estimating g(t) from the received signal after the matched filter and then use that an ISI-free system should have g(t) with zero-crossings at $\pm$T. It is described below.
- Provision of periodic components by non-linear filtering of the signal. An example will be given below in Section 5.7.3.3.

The methods may be characterized in various ways. One way is the structure: feedforward or feedback. The first and the last methods would often be feedforward since they provide an estimate of the correct timing directly from the received signal whereas the other

methods typically are feedback or error-tracking systems using some detector to give (an estimate of) the difference between a proposed timing and the correct time. This difference is then processed and a better proposal is found. Thus the operation may be seen as a loop trying to minimize the difference. A similar algorithm was seen in Example 4.6 for adaptive prediction.

The algorithms listed here are only representative examples of algorithms for baseband systems. In modulated systems (Chapter 6), the algorithms may also be used after the received signal has been moved back to the baseband, but this movement may require another kind of synchronization, carrier synchronization, and there may be differences in the behavior if this synchronization is not perfect. This issue is discussed a little in the Section 5.7.3.2 below.

5.7.3.1 Early-late synchronization

A simple implementation of symbol synchronization is the so-called **early-late** timing recovery or early-late gate. The operation of this method is based on the fact, that at the output of a matched filter, (5.9), in a PAM system we see the autocorrelation function of the basic pulse plus the noise, possibly with some unknown shift so we do not know where to sample it. Thus the g(t) for the signal for which τ (and T) should be estimated is the total response − named x(t) in Section 5.3. In Figure 5.21 we show the output without noise, but $\tau = \pm T_S$, for the squareroot cosine roll-off pulse shown in Figure 5.14. Now assume that the maximum output is found at some sample n_0 which is the value to be used for the decision, but we do not know what n_0 is. The early-late approach is based on the observation that the two samples before (early), $g(n_0-1)$, and after (late), $g(n_0+1)$, have the same value if n_0 is correct. This is always the case after a matched filter since the autocorrelation function is symmetric. If we calculate the difference $s(n_0) = g(n_0+1) - g(n_0-1)$, a positive difference indicates that the optimal sampling point is to be moved in the positive direction, i.e. n_0+1, and the opposite situation is found for a negative value as it may seen from the figure. A small difference should not move the sampling point, since it is likely that a new point is worse. In a real system, the values observed at the correct sampling point are $a_k \cdot g(n_0)$ if the system is without ISI. For the values at $n_0 \pm 1$, the signal also contains contributions from the neighbor symbols multiplied by some value of the g pulse. If g(t) has most of its energy concentrated between $\pm T$ we could neglect these terms. Since the a_k may be positive or

negative, the difference cannot be used directly. Instead we use the difference between the absolute values of the samples. Noise is also added to the output, such that an averaging over several observations is needed to make a decision about moving the sampling point.

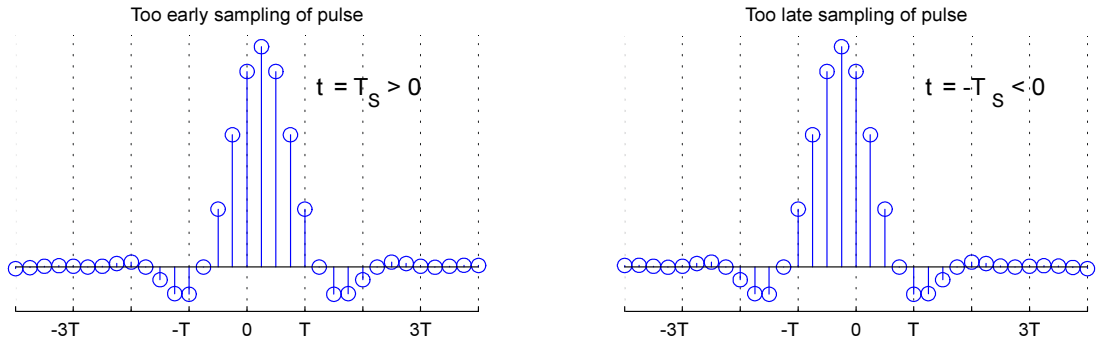

Figure 5.21
Output from matched filter using the pulse from Figure 5.14, but a timing offset ($\pm T_S$) in the receiver. Note that the pulse without timing offset is x(t) from Figure 5.13.

Each step in the early-late synchronization algorithm then becomes

Calculate

$$s_{n_0} = |v_{n_0+1}| - |v_{n_0-1}|$$

where $\{v_k\}$ is the output at the filter at time $k \cdot T_S = k \cdot T/m$.

Then calculate the average over S observations

$$s_{av} = \frac{1}{S} \sum_{k=0}^{S-1} s_{n_0 - k \cdot m}$$

If $s_{av} > \delta$ (a small value)

$n_0 = n_0 + m + 1$ (move 1 to the right)

else if $s_{av} < -\delta$

$n_0 = n_0 + m - 1$ (move 1 to the left)

else

$n_0 = n_0 + m$ (keep the sampling point).

The algorithm may lock to a minimum around T/2 as well, but this is an unstable situation. Similar algorithms exist based on the zero crossings where the correct sampling point is found as the midpoint between zero crossings (still after some averaging to suppress the noise). To make the synchronization sufficiently precise, the oversampling factor has to be larger than needed for the detection itself. This may be a problem for high data rates, and the samples at the optimal point and close to the optimal point have to be replaced by some

interpolation between the samples actually carried out, [5.15, 5.16], so in this way you also adjust for fractions of the sampling time. This does not change the principle of the algorithm. Since it cannot be assumed that T_S divides T − ideally this would have been true, but in practice T_S is often generated by a free-running clock for the sampler − the point to be interpolated for is moving between the samples so no fixed algorithm or filter may do it.

5.7.3.2 Mueller and Müller synchronization

The early-late method described above requires two samples just around the estimated sampling point, i.e. three samples per symbol in total which may be a problem for high-speed systems. Further, the estimation of the g(t) seems to be rather coarse being based just on the signal values which are a result of many neighbor pulses. For ISI-free systems you could remove this effect by sampling at ±T instead and the same effect as the one used for early-late is seen from the Figure 5.21 also at ±T. Another advantage could be that these samples have to be used anyway in the detection.

If we assume that the $\{a_k\}$ are independent random variables with zero mean, we may estimate $g(T-\tau)$ from the output of the matched filter (here denoted v(t)) by calculating at the assumed correct times $t = n_0 T_S$ (n_0 as for early-late):

$$E\left[\hat{a}_{n-1} v(n_0 T_S)\right] = \sum_{k=-\infty}^{\infty} E[\hat{a}_{n-1} a_k g(n_0 T_S - kT)]$$

$$= \sum_{k=-\infty}^{\infty} E[\hat{a}_{n-1} a_k g(nT - \tau - kT)] = E[a^2]\, g(T-\tau)$$

where $\hat{a}_{n-1}$ is the previously detected symbol. A similar calculation with $\hat{a}_{n+1}$ may give $g(-T-\tau)$, but $\hat{a}_{n+1}$ is not known at time nT, so instead we calculate

$$E\left[\hat{a}_n v((n_0-m)T_S)\right] = \sum_{k=-\infty}^{\infty} E[\hat{a}_n a_k g((n-1)T - \tau - kT)] = E[a^2]\, g(-T-\tau)$$

From Figure 5.21 it is seen that a positive difference $g(T-\tau) - g(-T-\tau)$ indicates that $\tau > 0$ and the assumed sampling points should be moved to a later time, $n_0 = n_0+m+1$, and the opposite is true for a negative difference. If the resulting pulse is not symmetric around zero − like for PR-systems − a slightly modified approach may be used.

The method is from [5.17], and the description here is from [5.14]. Note that this method is decision directed in contrast to the other methods described in this section since the decided symbols $\hat{a}_n$ are used in the algorithm.

It turns out that the Mueller and Müller method does not work properly for modulated systems where the carrier synchronization is not perfect. This problem is remedied in the algorithm described in [5.18] which uses two samples per symbol.

5.7.3.3 Provision of periodic components from the signal

As it may be seen from Sections 3.5 and 5.8, no periodic components ("lines") are found in the power spectrum for certain signals and we have achieved this with purpose. A linear processing of a signal cannot create such components which makes it necessary to use non-linear methods. The most popular methods are rectification, squaring or raising to even higher powers. As an example, double rectification (i.e. inverting negative pulses) for an AMI-signal (Example 3.7 or Section 5.8.2.1) recreates the original PAM-signal which may contain lines (e.g. with RZ pulses, cf. Example 3.10 and Equation (3.28)). It is rather difficult to calculate the spectra using the methods from Section 3.5 and we shall only provide the following simple example.

Example 5.11 - Spectrum for squared signal

We shall assume that $\{a_k\}$ is stationary with statistically independent symbols and zero mean. For the implementation it is also necessary to know T approximately thus making it possible to e.g. implement a bandpass filter with center frequency 1/T. The phase τ on the other hand is completely unknown. The squared signal is

$$v^2(t) = \sum_{k=-\infty}^{\infty} a_k^2 g^2(t-kT-\tau) + \sum_{n=-\infty}^{\infty} \sum_{m \neq n} a_n a_m g(t-nT-\tau) g(t-mT-\tau)$$

The first term may have periodic components allowing T and τ to be found since it is a PAM-signal with mean > 0 (Equation (3.28)). One way of determining T and τ could be to detect the zero-crossings (i.e. small values) of $v^2(t)$, but unfortunately the second term will disturb this determination. Noise may also do so, but we did not include noise in the expressions above in order not to make

them too complex. Thus it is necessary with some kind of averaging to suppress **phase noise** (*fasestøj*) or **jitter**. If we average over infinite time, the second term disappears since the mean of a is zero, so we have

$$E[v^2(t)] = E[a^2]\sum_{k=-\infty}^{\infty} g^2(t-kT-\tau)$$

which may be reformulated using

$$\sum_{k=-\infty}^{\infty} f(t-kT) = \frac{1}{T}\sum_{k=-\infty}^{\infty} F\left(\frac{2\pi k}{T}\right) e^{j2\pi kt/T}$$

and remembering that the Fourier transform of a product is a convolution

$$E[v^2(t)] = \frac{E[a^2]}{T}\sum_{k=-\infty}^{\infty} A\left(\frac{2\pi k}{T}\right) e^{j2\pi k(t-\tau)/T}$$

where the convolution is denoted as

$$A\left(\frac{2\pi k}{T}\right) = \int_{-\infty}^{\infty} G\left(\frac{2\pi k}{T}-\omega\right) G(\omega)\,d\omega$$

If g(t) is the pulse on the output of a matched filter, G(ω) becomes real (Equation (5.20)) which implies that A(−2πk/T) = A(2πk/T) and we obtain

$$E[v^2(t)] = \frac{E[a^2]}{T}\sum_{k=1}^{\infty} 2A\left(\frac{2\pi k}{T}\right)\cos\left(\frac{2\pi k(t-\tau)}{T}\right) + \frac{E[a^2]}{T}A(0)$$

Sometimes a special filter for synchronization purposes is found with the above property. From the expression it is seen that it is easy to filter out the 1st harmonic from the signal, cos(2π(t−τ)/T), which may be used to provide the correct sampling point since it determines both T and τ. For some signals there is no need to filter since the coefficients A(2πk/T) = 0 for $k \geq 2$ if G(ω) vanishes for $|\omega| \geq 2\pi/T$. This is the case for the cosine roll-off characteristics, (5.19), with $\alpha > 0$. If $\alpha = 0$ or G(ω) is more "narrow" than π/T (i.e. ISI is found) an unfortunate situation appears since all coefficients A(2πk/T) are 0. Thus squaring cannot be used for such signals and other methods have to applied.

The necessary filtering may be done with a **phase locked loop**, **PLL** (*faselåst kredsløb*) which is described in the literature, e.g. [5.11] - [5.14]. If designed properly, both the bandpass filtering (with center frequency equal to the expected 1/T) and the averaging are performed.

5.8 Line codes

In Chapter 7 we shall describe several transmission channels and their properties. The use of these channels gives a need for adjusting the transmitted signal to the properties of the channel. Even if the complete adjustment as described in Sections 5.3 and 5.5 (and with equalizer, Section 5.6) is not possible there will still be a considerable interest in processing the transmitted signal in such a way that certain properties are matched to the channel.

The main properties that you would like to have an effect on are

- Matching the spectrum to the channel in order to
  1. increase the received power as much as possible,
  2. reduce the power in the vicinity of 0 Hz (necessary since the transfer function for low frequencies is often very poor, e.g. due to the use of transformers in the transmission line),
  3. reduce crosstalk (which is more prevailing at high frequencies, see Section 7.2.3).
  4. avoid lines in the spectrum since such periodic components may disturb neighboring transmission systems, e.g. the symbol synchronization in these systems, cf. Section 5.7.
- Allow symbol synchronization in the received signal after simple processing which e.g. may reproduce the spectrum lines removed above, cf. Section 5.7.
- Temporal or spatial characteristics. e.g. avoiding isolated ones or long sequences of the same symbol.
- The probability distribution of the symbols, e.g. for shaping, cf. Section 5.4.
- Allow error-detection and maybe even error-correction.

Many of these properties may be influenced by changing the power spectrum in an appropriate way. We calculated the spectrum for baseband signals in Section 3.5 and the resulting equation was (3.27). From that equation it is seen that shaping the pulse g(t) differently may change the spectrum, but it is technically rather complicated. The more appealing possibility is to change the autocorrelation function of the symbol sequence $\{a_k\}$. This is done by a so-called **line code** (*liniekode*) which is a mapping of blocks of a-symbols (in the present context always binary symbols) to blocks of b-symbols:

$$b_{j+1} \ldots b_{j+n} = f(a_{k+1},\ldots, a_{k+m}; s_j).$$

The line code maps m binary symbols a_k to n symbols in the b-alphabet. Partial response polynomials, Table 5.2, may also be used as line codes, but the input sequence is convolved with the polynomial giving overlapping output symbols, not limited to blocks as above. Since the spectra were treated in Section 5.5, we restrict the treatment here to the block oriented approach.

Named after the alphabet size, simple line codes may be grouped as:

mBnB for binary b-symbols (0,1, or −1,+1),
mBnT for ternary b-symbols (−1, 0,+1), and
mBnQ for quarternary b-symbols (−3,−1,+1,+3).

The mapping above is also dependent on s_j which is a state of the encoder. The state may be used for giving dependency between the encoded blocks. Many examples are given in the following sections. In some cases, line codes are known as modulation codes in the literature, [5.13]. More advanced use of line coding in partial response systems or for shaping has larger alphabets, and such systems are out of the scope for the simple treatment here.

An important characteristic for a line code is the **code rate** (*kodens hastighed*)

$$R = \frac{m}{n \cdot \log_2(\text{size of b-alphabet})}$$

that describes the efficiency of the code and a high efficiency reduces the symbol rate of the b-symbols which may reduce the content of high frequencies. Codes with high code rate do not provide good error-detecting capabilities. Note that the code must be **transparent** (*transparent*), i.e. all a-sequences may be coded, i.e. $2^m \leq (\text{size of b-alphabet})^n$. Thus the rate is at most 1.

Analysis of a particular line code's effect on the above properties will of course demand calculations and simulations. However, an impression of the properties regarding power in

the vicinity of 0 Hz and the synchronization possibilities may be obtained by considering the following simple characteristics, [5.19, G.701 definitions 9008-9011]:

Disparity (*disparitet*) defined as the sum of b-symbols (+1, 0, − 1) within each coded block of n symbols:

$$\sum_{k=1}^{n} b_{j+k}.$$

A code is known as **balanced** (*balanceret*) if the disparity is 0 for all coded blocks. The definition of 'balanced' are by some authors more general, e.g. finite digital sum variation (see below).

Digital sum (*digital sum*) − also known as **running digital sum** − is the disparities summed over some period, e.g. from some reset state. It is possible to prove that in order to suppress 0 Hz, this sum should be kept finite. The state of the encoder, s_j, very often is a direct function of this sum and may be used to give the encoding of the next block in order to keep the sum close to 0. It may also be used by the receiver to detect transmission errors since a received block may conflict with the state or the digital sum.

Digital sum variation (*digital sumvariation*) is the maximal variation of the digital sum, i.e. max{dig. sum} − min{dig. sum} where max and min are determined from some sequence. The most interesting value is of course found from a worst case sequence. As mentioned above, a small digital sum variation means low content of low frequencies in the spectrum.

Runlength (*løbende længde)* is the number of identical symbols following each other. A small maximum runlength improves the synchronization possibilities.

When evaluating a particular line coding scheme the complexity of the implementation is also an issue.

In the following sections we shall describe a number of popular line codes. In all examples, **T means duration of a coded symbol** (i.e. symbols named b above). Reference [5.20] provides many more examples and gives references of origin etc. In Section 5.9 we describe

the considerations around the different parameters and properties when choosing line codes for the ISDN subscriber line. In the description of the line codes, the power spectrum is almost always given. Although it sometimes is too tedious to show the calculation, the methods presented in the Section 3.4 on Markov sources may almost always be used which is also implicit in the description of the encoder state given above.

5.8.1 mBnB-line codes

5.8.1.1 Manchester coding

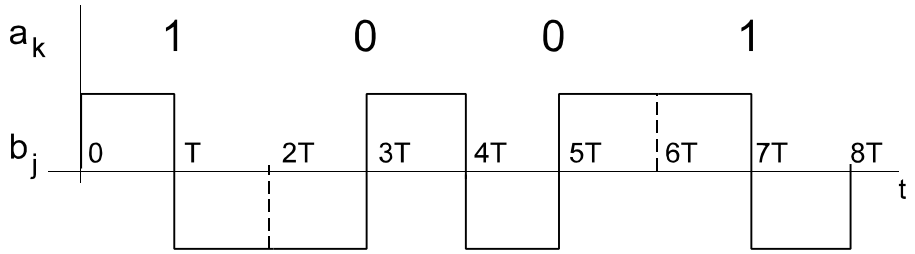

Figure 5.22
Example of Manchester coding with NRZ pulse g(t).

Manchester coding (*Manchester kodning*) is also known as **split phase**, **bi-phase** (*bifase*), **twinned binary**, or **1-of-2**. The popularity of this code is due to its good synchronization properties from the zero-crossing found in each block. The code is

a_{k+1}	b_{j+1}	b_{j+2}
0	−1	+1
1	+1	−1

An example of a coded sequence with NRZ pulses (each with length T) is shown in Figure 5.22. Note that the duration of each a-symbol is 2T.

The power spectrum may be calculated using the equation (3.27) after calculating $R_B(k)$ using the Example 3.2. However the simplest method is to join g(t) and −g(t−T) into one pulse, g_g(t) (with transform G_G(ω)), of duration 2T and then use Equation (3.28) for independent symbol sequences, $a_k \in \{-1,+1\}$. If the probability of a_k = ½, then E[A] = 0 and $\sigma_A^2 = 1$ and we get:

$$S_V(\omega) = \frac{\sigma_A^2}{2T}|G\text{–}G(\omega)|^2 + \left(\frac{E[A]}{2T}\right)^2 \sum_{\ell=-\infty}^{\infty} \left|G\text{–}G\left(\frac{2\pi\ell}{2T}\right)\right|^2 \delta(\omega - \frac{2\pi\ell}{2T})$$

$$= \frac{1}{2T}\left|G(\omega)(1 - e^{-j\omega T})\right|^2 + 0$$

$$= \frac{|G(\omega)|^2}{T}(1 - \cos\omega T).$$

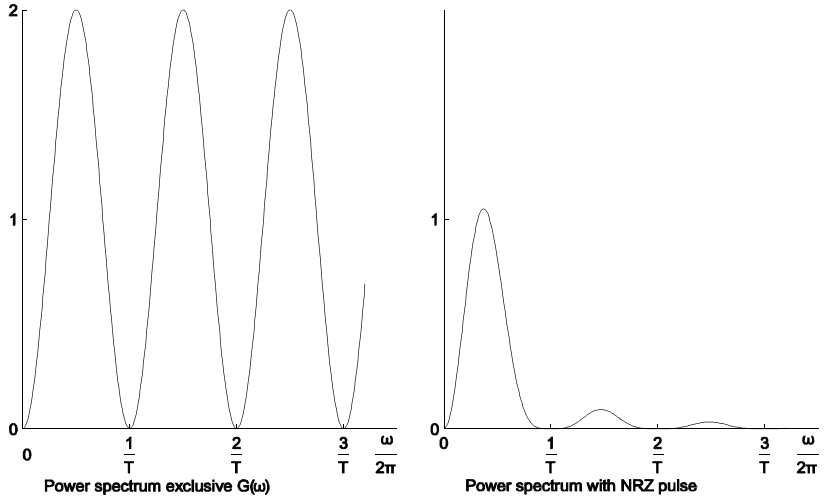

Figure 5.23
Power spectrum for Manchester coding.

The spectrum is shown in Figure 5.23. If the probability of $a_k = p \neq ½$, then E[A] = 2p − 1 and $\sigma_A^2 = 4p - 4p^2$ and using NRZ pulses we get lines in the spectrum at frequencies $(2\ell + 1)2\pi/(2T)$.

5.8.1.2 CMI coding

CMI-coding (**coded mark inversion**, mark is another name for logical 1) also known as 2-level AMI (see Section 5.8.2.1) is used in the ITU-T standardized 139264 kbit/s system with NRZ-pulse ([5.19, G.703], see also Chapter 8). The code is

a_{k+1}	b_{j+1}	b_{j+2}	
0	−1	+1	
1	+1	+1,	if 1 was coded −1 −1 last
1	−1	−1,	if 1 was coded +1 +1 last.

The encoder has one state variable telling the last coding of 1. The three disparities are 0, +2, and −2, respectively. The digital sum may assume values 0 and +2 or 0 and −2

dependent on the first encoded 1. Thus the digital sum variation is 2. The maximum runlength is 3. The spectrum may be calculated by realizing that the code may be seen as a sum of a periodic signal (period 2T), a Manchester code and an AMI-code. Another possibility is that of establishing a Markov source for the code, but here a periodic sequence also has to be subtracted first. We shall not attempt to calculate the spectrum here, but if a_{k+1} assumes 0 and 1 with equal probability we have:

$$S_V(\omega) = \frac{|G(\omega)|^2}{T}\left(\frac{3}{4} - \frac{\cos 2\omega T}{2} - \frac{\cos 3\omega T}{4}\right) + \frac{|G(\omega)|^2}{4T^2}\sum_{\ell=-\infty}^{\infty}\delta\left(\omega - (2\ell+1)\frac{2\pi}{2T}\right)$$

The low frequencies are well suppressed and the content of high frequencies is rather large. The zero crossing in the coding of 0 provides excellent synchronization capabilities. There is a possibility of error-detection since +1 −1 does not exist and the coding of 1 has to change. The code rate is only 1/2.

5.8.1.3 Balanced mBnB codes

As a generalization of Manchester coding other balanced codes with even n and disparity 0 for all blocks may be constructed. Below we calculate the spectrum of such an 6B8B code.

Example 5.12 - Power spectrum for 6B8B balanced code

In general there exist

$$\binom{2q}{q} = \frac{(2q)!}{q!\,q!}$$

balanced codewords of length 2q containing q times +1 and q times −1. For q = 4 this is 70 and since the code has to be transparent, 6 binary symbols ($2^6 = 64$) may be coded with balanced codewords of length 8, thus forming a 6B8B code.

Since the symbols are ±1, we have $R_B(0) = 1$. To calculate the autocorrelations for $k \neq 0$ we neglect synchronization of the codewords and assume that a random signal with probability 1/8 starts a new codeword at a certain b_n, and we may assume that $b_n = 1$ ($b_n = -1$ gives the same result due to the symmetry). If b_{n+k} is in the same codeword, the probabilities are $P(b_{n+k} = 1) = 3/7$ and $P(b_{n+k} = -1) =$

4/7, and thus the contribution to $E[b_n \cdot b_{n+k}]$ becomes $-1/7$. If b_n and b_{n+k} are in different codewords, they are independent and do not contribute to $R_B(k)$. The probability of b_{n+k} being in the same codeword is 7/8 for k = 1 down to 1/8 for k = 7 and 0 for k > 7, i.e. $(1 - |k|/8)$ for $|k| \le 7$. Thus

$$R_B(k) = \begin{cases} 1 \text{ for } k = 0 \\ -\frac{1}{7}\left(1 - \frac{|k|}{8}\right) \text{ for } 1 \le |k| \le 7 \\ 0 \text{ other } k \end{cases}$$

From (3.27) and after a few calculations we obtain

$$S_V(\omega) = \frac{|G(\omega)|^2}{T}\left(1 - \frac{7}{28}\cos\omega T - \frac{6}{28}\cos 2\omega T - \frac{5}{28}\cos 3\omega T - \frac{4}{28}\cos 4\omega T - \frac{3}{28}\cos 5\omega T - \frac{2}{28}\cos 6\omega T - \frac{1}{28}\cos 7\omega T\right)$$

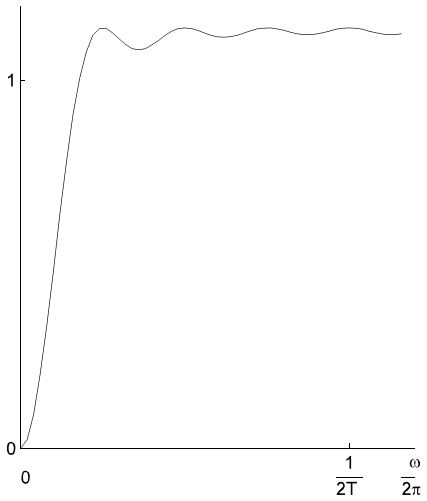

Figure 5.24
Power spectrum (excl. G(ω)) for 6B8B-code.

which may be reduced to

$$S_V(\omega) = \frac{|G(\omega)|^2}{T}\left(\frac{8}{7} - \frac{1}{56}\left(\frac{\sin 4\omega T}{\sin \omega T/2}\right)^2\right)$$

The spectrum is 0 at $\omega = 0$ and rather low at low frequencies. Note that the calculation is dependent on the choice of the 64 codewords out of the 70 possible balanced blocks in a random way. The spectrum may be affected by a specific choice.

5.8.1.4 Runlength limited codes

For many applications, e.g. storage on CD or DVD (see Section 7.5.3), binary codes with restrictions on the runlength are used. Both the minimal and the maximal runlength are restricted, e.g. at least 3 and at most 11. The encoding is normally a two-step process: A binary sequence of zeros and ones is created where the ones are isolated and separated by zeros with restrictions on the runlength. A one in such a sequence indicates that the next output symbol is an inversion of the previous one while a zero indicates no inversion. In this way the runlength of the output sequence is varying between 3 and 11 if the run length for the zeros varies between 2 and 10. It is easily seen that a Markov chain could generate the first step of such a code − a so-called **runlength limited** or **RLL code**. As an example we could have the transition and output matrices as

$$\mathbf{Q} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} * & 0 & * & * \\ 1 & * & 0 & * \\ 1 & * & * & 0 \\ 1 & * & * & * \end{bmatrix}$$

giving runs of zeros varying between 1 and 3. However, it would not be easy to map random binary sequences to this code since for some states only one transition is allowed and a block of input is likely to give varying length of outputs depending on the contents. Thus block oriented nBmB RLL codes are preferred. These are created by searching among the 2^m binary patterns and keeping only patterns fulfilling the runlength requirement also when concatenated with another codeword of m bits. Maybe some states for the encoder would be of help to control the concatenations of patterns such that patterns ending with many zeros could only be concatenated with patterns starting with a few zeros. One example of such a code is the 8B16B code used for DVDs, see Section 7.5.3.

5.8.1.5 Other mBnB codes

Codes of the mBnB-type are often used in optical communication. In some implementations of 565 Mbit/s systems (introduced in Chapter 8) a 5B6B kode with maximum disparity 2 and maximum runlength 6 is used, [5.20].

5.8.2 mBnT-line codes

5.8.2.1 AMI-code

The AMI-code (**Alternate Mark Inversion**, mark is a traditional name for logical 1) also known as the **bipolar** (*bipolære*) **code** is often used for reduction of the spectrum at low frequencies. Examples are ITU-T's standardized connection to 64 kbit/s [5.19, G.703] and the S-bus in the ISDN-system (cf. Section 1.5). The code is

a_{k+1}	b_{j+1}	
0	0	
1	+1,	if 1 was coded −1 last
1	−1,	if 1 was coded +1 last.

The state variable gives the last encoding of 1.

Assuming statistically independent a-symbols with probability 1/2 for 0 and 1 we calculated the power spectrum for the b-sequence in Example 3.7:

$$S_B(\omega) = \frac{T}{2}(1 - \cos\omega T)$$

and from (3.27) we obtain

$$S_V(\omega) = \frac{|G(\omega)|^2}{T^2} S_B(\omega) = \frac{|G(\omega)|^2}{2T}(1 - \cos\omega T)$$

The second factor in the spectrum expression is shown in Figure 5.25, i.e. the power spectrum exclusive $G(\omega)$. The spectrum is without lines which could be needed for synchronization. However such lines are easily provided using a double rectifier since it produces the original a_k-sequence and dependent on the pulse shape, lines may be obtained, as e.g. with RZ-pulses (cf. Example 3.10 and (3.28)) which are used in the above mentioned standards. From Figure 5.25 it is seen that low frequencies are suppressed. The spectrum is rather dependent on the probabilities of 0 and 1. If $P(a_k=1)$ is p, the spectrum may be found with help from Example 3.7 as:

$$S_V(\omega) = \frac{|G(\omega)|^2}{T} \cdot \frac{2(1-p)p}{1-2(1-2p)\cos\omega T + (1-2p)^2} \cdot (1-\cos\omega T)$$

In some situations a long sequence of identical a-symbols (e.g. some idle state) may appear and such sequences make the synchronization difficult. The runlength is in principle not limited and scrambling (Section 2.6) is advised. The digital sum may assume values 0 and +1 or 0 and −1 dependent on the encoding of the first, which means that the digital sum variation becomes 1.

The code offers a possibility of error detection due to the sign changes required for 1s. The code rate is 0.63.

5.8.2.2 HDB3-code

The above mentioned problems with long zero sequences for the AMI-code have lead to a number of modifications. Here we shall describe **HDB**n (**High Density Bipolar**) which is a modified AMI-code allowing at most n zeros in a row. Especially, the HDB3 is used in ITU-T PDH-standards (cf. Chapter 8) for 2048 kbit/s, 8448 kbit/s og 34368 kbit/s [5.19, G.703] in connection with RZ-pulses. Basically the encoding is as the AMI-code:

a_{k+1}	b_{j+1}	
0	0,	if less than four sequential 0s
1	+1,	if last $b \neq 0$ was −1
1	−1,	if last $b \neq 0$ was +1,

but a sequence of 4 zeros is coded as 0 0 0 1, 0 0 0 −1, 1 0 0 1, or −1 0 0 −1. The first two encodings are selected in such a way that they violate the usual AMI-rule of alternating ±1. In general the encoding is selected in such a way that the digital sum deviates the least possible amount from 0. All 4 encodings are **bipolar violations** (*bipolær forstyrrelse*) of the bipolar (AMI) rule. This explanation may not be quite clear but the table below should clear up things. Note that “Last b ≠ 0" refers to the coded symbols, i.e the sequence of symbols including the violations.

	Last 0 0 0 0 - encoding	
Last $b \neq 0$	0 0 0 1 / 1 0 0 1	0 0 0 −1 / −1 0 0 −1
1	−1 0 0 −1	0 0 0 1
−1	0 0 0 −1	1 0 0 1

Even though the HDB3 clearly is a Markov source, the calculation of the power spectrum is extremely tedious. A brute force attempt gives 56 states. A more elegant approach extends our treatment of Markov sources in Section 3.4 to variable length output, and the result is that only the 4 natural states from the table above are used [5.21]. The expression for the spectrum is still very complex, so we show only the result in Figure 5.25 without $G(\omega)$, i.e. only the spectrum for the line coded discrete sequence, $S_B(\omega)$, is shown. For comparison, the AMI spectrum is also shown.

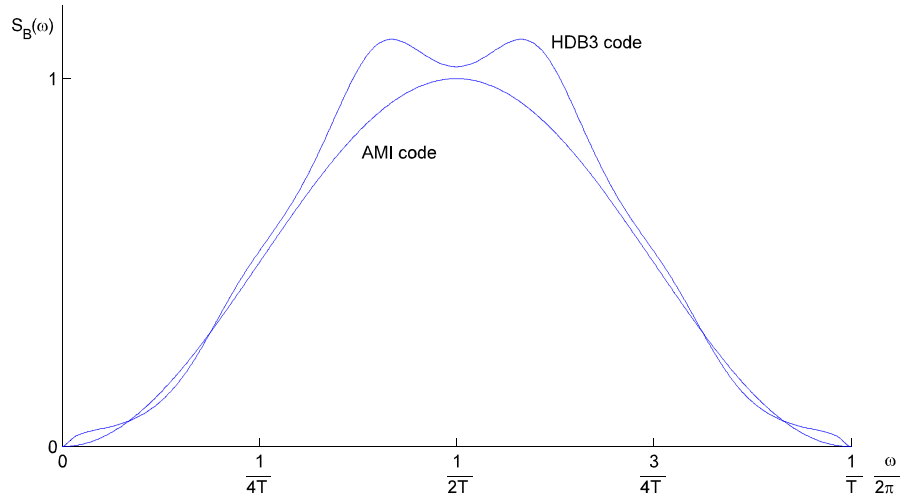

Figure 5.25
Power spectra (below 1/T Hz) for AMI and HDB3 coding
excluding the pulse g(t).

The spectrum is close to the AMI spectrum, but slightly more power is found at low frequencies. However, there is still a good suppression of these frequencies. The digital sum only assumes values −1, 0, and +1 (digital sum variation 2) indicating that suppression. Again no lines are found in the spectrum but a double rectifier will provide a sequence similar to the $\{a_k\}$ sequence (only more 1s are found at the violations). Dependent on the pulse shape, lines may then be obtained, as e.g. with RZ-pulses (cf. Example 3.10 and (3.28)) which are used in the above mentioned standards. As mentioned, only 3 zeros in a row are possible. This may be used for error-detection together with the digital sum.

5.8.2.3 MMS43-code

For ISDN connections (cf. Section 1.5) ordinary subscriber lines are used for transmission of the digital signals. A more detailed description is given in Section 5.9 where several requirements for the line code are discussed. In order to keep attenuation and crosstalk

within reasonable limits it is necessary to use more efficient line codes, i.e. the code rate should be better than the 0.63 for AMI/HDB3. Further the low frequencies should be avoided. Some implementations, e.g some Siemens systems in Denmark, are using the MMS43-code which is a 4B3T-code with rate 0.84. Often it is named **the 4B3T-code** since it is the only 4B3T-code which has found practical applications, [5.20, 5.3, 5.22]. The encoding is shown in Table 5.3, from [5.23].

MMS43	State s=0		State s=1		State s=2		State s=3	
4B	3T	New s	3T	New s	3T	New s	3T	New s
0 0 0 1	0 − +	0	0 − +	1	0 − +	2	0 − +	3
0 1 1 1	− 0 +	0	− 0 +	1	− 0 +	2	− 0 +	3
0 1 0 0	− + 0	0	− + 0	1	− + 0	2	− + 0	3
0 0 1 0	+ − 0	0	+ − 0	1	+ − 0	2	+ − 0	3
1 0 1 1	+ 0 −	0	+ 0 −	1	+ 0 −	2	+ 0 −	3
1 1 1 0	0 + −	0	0 + −	1	0 + −	2	0 + −	3
1 0 0 1	+ − +	1	+ − +	2	+ − +	3	− − −	0
0 0 1 1	0 0 +	1	0 0 +	2	0 0 +	3	− − 0	1
1 1 0 1	0 + 0	1	0 + 0	2	0 + 0	3	− 0 −	1
1 0 0 0	+ 0 0	1	+ 0 0	2	+ 0 0	3	0 − −	1
0 1 1 0	− + +	1	− + +	2	− − +	1	− − +	2
1 0 1 0	+ + −	1	+ + −	2	+ − −	1	+ − −	2
1 1 1 1	+ + 0	2	0 0 −	0	0 0 −	1	0 0 −	2
0 0 0 0	+ 0 +	2	0 − 0	0	0 − 0	1	0 − 0	2
0 1 0 1	0 + +	2	− 0 0	0	− 0 0	1	− 0 0	2
1 1 0 0	+ + +	3	− + −	0	− + −	1	− + −	2

Table 5.3
Encoding table for MMS43.

As it may be seen, it is rather complicated. It is easily seen that the disparities vary from −3 to 3, and that the maximum runlength is 5. Each state in the table corresponds to a certain value of the digital sum (e.g. −1, 0, 1, 2 dependent on the starting state). Thus the digital sum variation becomes 3. If the a-sequence has statistically independent symbols and each value has probability 1/2, the steady state probability is 1/4 for each state and the average of the b-sequence becomes 0. It is difficult to calculate the spectrum, but we have done so using methods similar to the methods in Example 3.9, [5.24]. The result is

$$S_V(\omega) = \frac{|G(\omega)|^2}{T} \cdot \frac{d(\omega T)}{234260 - 126880\cos 3\omega T - 19968\cos 6\omega T} \quad \text{where}$$

$$d(\omega T) = 154713 - 55172\cos\omega T - 3419\cos 2\omega T - 92194\cos 3\omega T + 9820\cos 4\omega T + 1068\cos 5\omega T - 15328\cos 6\omega T + 768\cos 7\omega T - 256\cos 8\omega T$$

The spectrum with NRZ-pulses is shown in Figure 5.27 together with a number of other interesting spectra. From that figure it is seen that the content of high frequencies is reduced mainly due to the efficiency of the code. Other mBnT codes have been investigated for this particular application but MMS43 is the only survivor.

5.8.3 2B1Q-code

An even better efficiency for codes for the digital subscriber line is obtained with the **2B1Q-code** with rate 1 and encoding table

a_{k+1}	a_{k+2}	b_{j+1}
1	0	+3
1	1	+1
0	1	−1
0	0	−3

Here no dependency is found between the coded symbols which means that an a-sequence with statistically independent symbols and each value with probability 1/2 results in a b-sequence which is also statistically independent and where each value has probability 1/4. Thus the name “code” is nearly too much since it is an ordinary PAM-signal with zero mean. Thus the spectrum is found from (3.28):

$$S_V(\omega) = \frac{|G(\omega)|^2}{T} R_B(0) = \frac{|G(\omega)|^2 \cdot 5}{T}$$

The spectrum is shown in Figure 5.27 for NRZ-pulses. As for the MMS43-code the content of high frequencies is reduced mainly due to the high efficiency of the code. It is seen that this code does not suppress low frequencies which provides some problems with the analog subscriber lines which have a poor transfer function for these frequencies.

Note that even though the encoding rule does not influence the spectrum it is still smart since an error in a received b-symbol to its nearest neighbor only results in one bit error in the a-sequence (Gray-coding, cf. Table 5.1)).

5.9 Digital subscriber line, DSL

In the last two decades or so, much development has been seen in transmission techniques used for converting previously analog telephony subscriber lines into **digital subscriber lines, DSL**, (*digitale abonnentlinier*). The different access technologies for these metallic subscriber lines are also known under the common name **xDSL**. Here we shall describe the DSL used in ISDN (sometimes known as IDSL) in some detail, and we shall very briefly mention other xDSL techniques.

As described in Section 1.5, the subscriber of the basic ISDN service sends and receives 2·64 + 16 kbit/s = 144 kbit/s via his subscriber line. Since frame synchronization and other overhead should also be included, the required bit rate is 160 kbit/s, and the transmission should be duplex. In this section we shall describe basic transmission on the subscriber line (ITU-T has named it the U reference point in the recommendations). Part of the basis for the description is from [5.12, 5.25, and 5.2].

In Chapter 7 we shall present a circuit element, a so-called hybrid (*gaffel*(*kobling*)), aimed at separating the two directions of transmission such that a transmitter and a receiver can be connected to the same transmission line. However, this circuit element is not ideal since a small fraction of the transmitted power is lead to your own receiver, and even if it is a small fraction, it may very be much stronger than the weak signal that is received from the remote end. Thus something extra is needed to separate the transmission and reception, and three principles have been investigated:

1. The transmission is simplex, but the direction is often changed, "the ping-pong-principle" also known as half duplex.
2. The transmitted signal is moved to another frequency by modulation (to be treated in Chapter 6) and filters block the unwanted frequencies.
3. The unwanted signals which are echoes of your own transmission are cancelled by an **echo canceller** (*ekkoudligner*) which is an algorithm that estimates the echo from the transmitted signal (in previous sections $a_k g(t - k \cdot T)$) and then subtracts the estimated echo before the receiver.

The first two methods require a rather large bandwidth which may cause problems with the attenuation of the high frequencies and with the crosstalk between neighboring lines which also increases with the frequency. The echo cancellation therefore is the only realistic possibility, although this is not easy as we shall see below. A complete transmitter/receiver is shown in Figure 5.26.

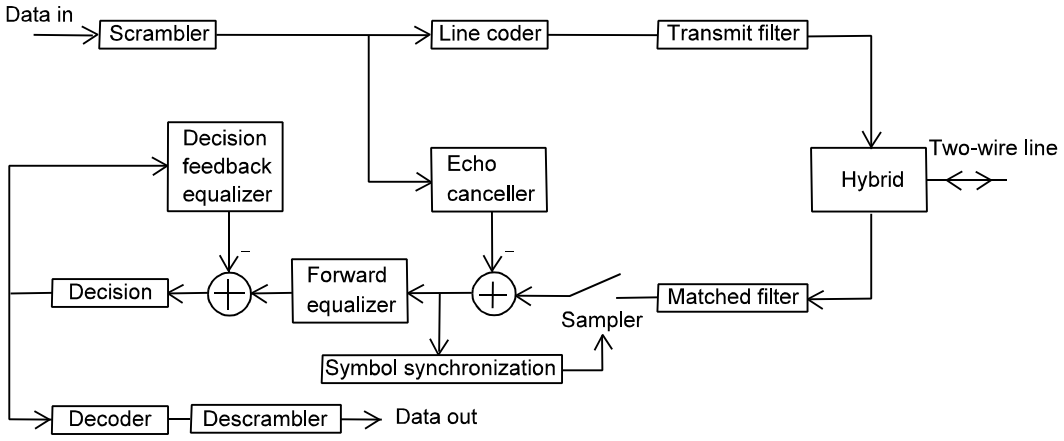

Figure 5.26
Transmitter/receiver for ISDN subscriber line.

The figure also shows a decision feedback equalizer (cf. Section 5.6.3) which has the purpose of compensating for the transfer function of the transmission line. Equalizers for subscriber lines are described in [5.2]. A scrambler is used for randomizing the transmitted data (cf. Section 2.6), and the symbol synchronization is also shown.

A basic decision in the construction is the selection of a line code which influence a number of characteristics, some of them listed below. Actually the line code is not standardized. Its choice has been left to the manufacturers. Some of the characteristics involved in evaluation of a line code are:

- Equalization of transmission line
- Cancellation of echo
- Sensitivity to the poor transfer function at low frequencies (due to the use of transformers)
- Sensitivity to crosstalk and reduction of generated crosstalk

- Signal-to-noise ratio for the required bit error rate. The attenuation of the signal is of great importance here. The noise may originate from the surroundings or from crosstalk from neighboring lines.
- Synchronization properties.
- The need for scrambling.

Most of the candidate line codes were described in Section 5.8: Manchester, AMI, MMS43 (4B3T), and 2B1Q, and from Section 5.5 we have the modified duobinary code (Class 4, Table 5.2) which may also serve as a 1B1T-line code. The power spectra for these candidates are shown in Figure 5.27 with use of NRZ-pulses. Each candidate shows advantages and disadvantages. Manchester is simple, no power at low frequencies, and excellent synchronization properties due to the zero crossing found in each symbol. However the spectrum used is very wide giving problems with crosstalk and attenuation. The modified duobinary code is better in relation to bandwidth. However even with precoding, it is more sensitive to noise than other 1B1T codes as e.g. the AMI. The AMI uses a slightly wider frequency range but this is compensated for by less sensitivity to noise. Like the partial response code, AMI is good at low frequencies and both need scrambling to safeguard symbol synchronization in the case of sequences of zeros. An advantage for AMI is the short filters used in the equalization and in the echo cancellation, e.g. the length of the echo is less than 10·T.

Simulations of MMS43 (4B3T, Table 5.3) have shown it to be better than the three systems analyzed above. The reduced bandwidth compensates the increased noise sensitivity which is due to the three-level symbols. However, a number of other difficulties is found. The content of low frequencies is significantly larger than for the three candidates above which increases the problems around transformers and reduces the attenuation of the unwanted signal in the hybrid thereby increasing the echo problem. The echo cancellation has to work with echos up to 45·T due to this and due to the memory of the code. Further, an efficient scrambling is required to remove possible correlation between the transmitted and the received sequence which may harm the echo cancellation. An extra problem is the need for frame synchronization to synchronize with the code blocks. MMS43 was very much used in the first commercial ISDN applications, e.g. Siemens equipment installed in Denmark.

In later systems, the 2B1Q-code were preferred due to the even better efficiency. However, the large content of low frequencies results in non-linearities due to transformers which requires non-linear echo cancellation [5.12]. It is necessary to use estimation methods for the symbol synchronization due to the NRZ pulse used. When these complex algorithm are implemented the performance of 2B1Q is slightly better than MMS43. Integrated circuits are available today for both the MMS43 and the 2B1Q code.

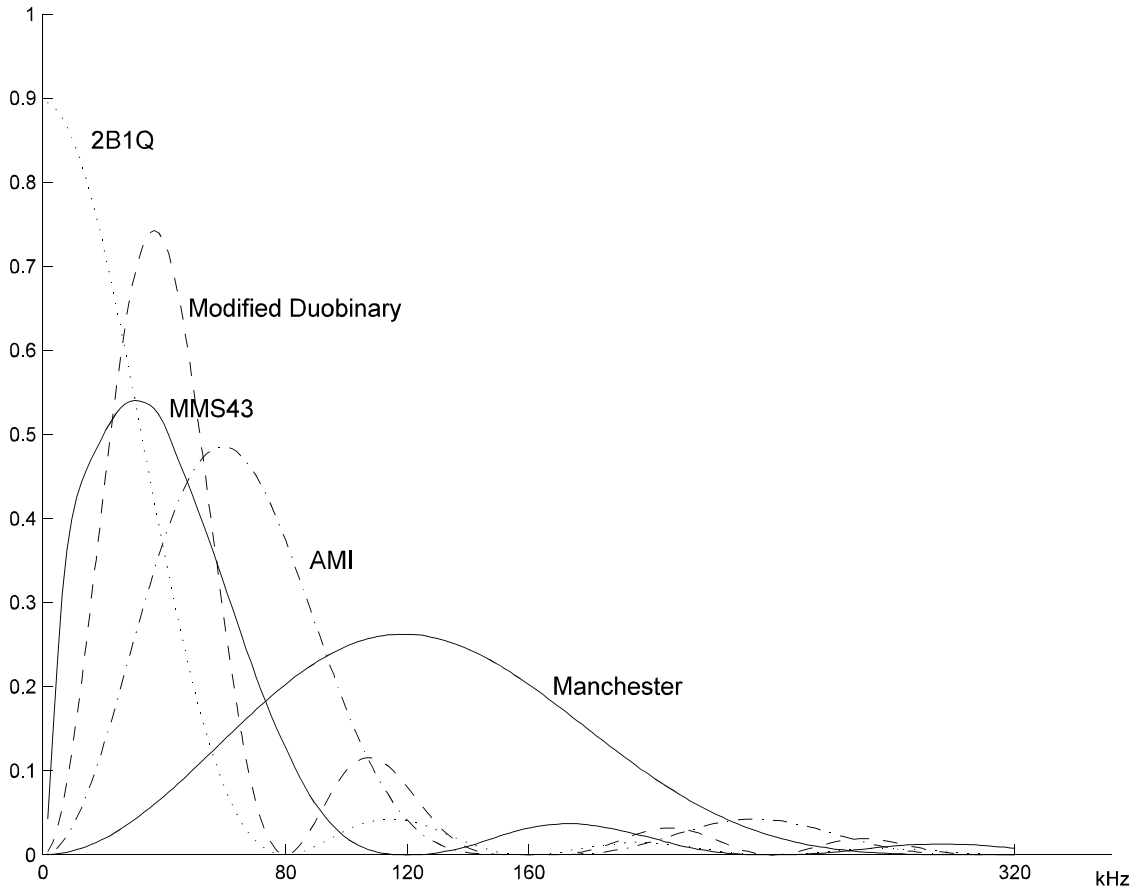

Figure 5.27
Power spectra for candidate line codes used with NRZ-pulses. The spectra are normalized in such a way that the total power in the region shown is constant.

Most metallic subscriber lines are able to transmit much more than 160 kbit/s. A number of techniques exists. The following table summarizes some contemporary systems. In the table some systems are beyond the descriptions in this chapter. Basically they are two-dimensional in contrast to the one-dimensional signals treated here: **CAP, Carrierless Amplitude and Phase Modulation** which, as the name tells, works with combined amplitude and phase modulation, **DMT, Discrete MultiTone** or even more complex

systems. DMT is discussed further in Chapter 6 about modulation (but not CAP which is basically use of pulses having only frequencies in a certain range - passband pulses).

System name	ITU-T	Bit rate	Type	# of pairs	Max. length	Line code or modulation
ISDN	G.961, I.430 etc.	160 kbit/s	Duplex	1	5.5 km	2B1Q or MMS43
HDSL	G.991.1	1.5 - 2 Mbit/s	Double Duplex	2	3.7 km	2B1Q or CAP
SDSL (HDSL2)	G.991.2	1.5 - 2 Mbit/s	Duplex	1	3.7 km	Coded modulation
ADSL	G.992.1	0.8 Mbit/s upstream 8.1 Mbit/s down + telephony	Duplex	1	3.7 km	DMT
ADSL Lite	G.992.2	0.5 Mbit/s upstream 1.5 Mbit/s down + telephony	Duplex	1	3.7 km	DMT
ADSL2	G992.3	3.5 Mbit/s upstream 12 Mbit/s down	Duplex	1	1.5 km	DMT
ADSL2+	G992.5	1 Mbit/s upstream 24 Mbit/s down	Duplex	1	1.2 km	DMT
VDSL2	G993.2	2.3 Mbit/s upstream 52 Mbit/s down	Duplex	1	0.3 km	DMT

Table 5.4
Some contemporary xDSL systems.

The main obstacle for a high speed DSL is crosstalk at the near end which is disturbing if transmitter and receiver use the same rather wide bandwidth. It is a bit easier to make **ADSL**, **Asymmetric Digital Subscriber Line** with up to 8.4 Mbit/s in one direction (down to the subscriber) and up to 768 kbit/s in the opposite direction. The maximum speed is only obtainable for rather short subscriber lines, i.e. 3 - 4 km. ADSL is now very popular in Denmark. The normal telephone (or ISDN) connection will still be operable together with the ADSL since it is separated from the ADSL signal by means of a splitting filter. A simpler standard, **ADSL-lite**, with a simpler coupling of the telephone is preferred by some manufacturers, but the capacity is smaller than the ordinary ADSL. The table also shows developments of ADSL into ADSL2, ADSL2+, and VDSL.

The reference [5.2] gives a thorough description of all xDSL technologies including **VDSL**, which may provide up to 50 Mbit/s or more.

5.10 Error requirements for links and digital networks

This course is mostly focused upon point-to-point connections which in a broad perspective could be seen as links in a digital network. Other courses discuss these networks, their architecture and components used, and in Chapter 8 we shall discuss how the links are shared among different users. In this section we shall focus upon the requirements for an end-to-end connection, and we shall deduct requirements for the individual links.

It is necessary to specify error performance objectives for digital transmission channels in order to assure that the information transmitted is received in a relatively undisturbed way. These objectives should hold for any kind of (international) connection between users. ITU-T has made recommendation for this in the G.82x-series. Modern transmission systems are rarely just transmitting a stream of bits, but the transmission is organized into blocks (e.g. SDH in Section 8.3), and this also holds for many services. The recommendations for block based systems are known as G.826, G.828 (for SDH) [5.26], and G.8201 for the recent Optical Transport Network (OTN). and it has replaced a previous recommendation G.821. which specifies error performance objectives for 64 kbit/s bitstreams as they are found e.g in international telephony. G.826 and G.828 define the following **error performance parameters**:

- **Errored block, EB** (*fejlbehæftet blok*) - a block with at least one bit in error.
- **Errored seconds, ES** (*fejlramte sekunder*) - seconds with at least one EB.
- **Severely errored seconds, SES** (*alvorligt fejlramte sekunder*) - seconds with at least 30% EBs or at least one severely disturbed period. The precise definition of the latter is skipped here, but it means of course periods with a very high number of errors.

The **error performance objectives** using these parameters are given in Table 5.5

Rate (Mbit/s)	Bits/block	Maximum values for (27500 km) connection		
		Rate of ES	Ratio of SES	Ratio of EB outside SES
1.5 - 5	2 000 - 8 000	0.04	0.002	$3 \cdot 10^{-4}$
5 - 15	2 000 - 8 000	0.05	0.002	$2 \cdot 10^{-4}$
15 - 55	4 000 - 20 000	0.075	0.002	$2 \cdot 10^{-4}$
55 - 160	6 000 - 20 000	0.16	0.002	$2 \cdot 10^{-4}$
155.520 (SDH)	19 440	0.04	0.002	10^{-4}
622.080 (SDH)	77 760	not relevant for high rates	0.002	10^{-4}
2 488.320 (SDH)	311 040	not relevant	0.002	10^{-4}
9 953.280 (SDH)	1 244 160	not relevant	0.002	10^{-3}
OTN	see G.8201			

Table 5.5
Error performance objectives from G.826 and G.828.
The SDH rates are raw rates, the actual rates with user data are slightly less.

G.826 and G.828 also give apportionment rules, i.e. rules of how to distribute the end-to-end performance objective among the different parts of the hypothetical reference path of 27500 km. This international reference path is assumed to consist of two terminating countries (national portion) at each end and four intermediate countries (international portion). The apportionment rules are shown in Table 5.6

The performance objectives of recommendation G.828 are generally stronger than the objectives in the previous recommendations which means that radio based systems (e.g microwave links or satellite links) require use of advanced methods such as error-correction, [5.6], to satisfy the requirements. Optical communication systems are closer to the requirements already at the basic communication channel (bit error rates less than 10^{-9}), but G.8201 also requires error-correction here.

Percent of total error performance objective			
National portion		International portion (4 intermediate countries are assumed)	
Block allowance	Distance allowance	Block allowance	Distance allowance
17.5% to both terminating countries	0.2% per 100 km	2% per intermediate country. 1% per terminating country	0.2% per 100 km
Note: Satellite hops each receive 35% but no distance allowance			

Table 5.6
Apportionment rules for error performance objectives from G.826 and G.828

The objectives described above give a very coarse description of the end-to-end transmission channel. Sometimes it is necessary to provide an even better performance for information of sensitive nature, e.g. money transactions or material where the data are compressed very hard. The quality is easily improved using error-correction, but error-correcting codes require a more precise description of the channel than the above performance objectives give, or otherwise the error-correction becomes very inefficient (i.e. a too low code rate is required).

5.11 References for Chapter 5

5.1 H. Nyquist: "Certain topics in telegraph transmission theory", AIEE, Vol. 47, 1928, pp. 617-644.

5.2 T. Starr, J.M. Cioffi, and P.J. Silverman: "Understanding Digital Subscriber Line Technology". ISBN 0-13-780545-4, Prentice Hall PTR 1999.

5.3 S. Benedetto, E. Biglieri, and V. Castellani: "Digital Transmission Theory". ISBN 0-0-13-214313-5, Prentice-Hall Inc. 1987.

5.4 C.E. Shannon: "A Mathematical Theory of Communication", Bell Systems Technical Journal, Vol. 27, 1948, pp. 379-423, 623-656.

5.5 G.D. Forney, Jr. and G. Ungerboeck: "Modulation and Coding for Linear Gaussian Channels", IEEE Trans. Info. Theory, vol. IT-44, October 1998, pp. 2384-2415.

5.6 J. Justesen and T. Høholdt: "A Course in Error-Correcting Codes". ISBN 3-03719-001-9, European Mathematical Society Publishing House 2004.
(Used in DTU course 01405 Error-Correcting Codes).

5.7 G. Ungerboeck: "Channel Coding with Multilevel/Phase Signals", IEEE Trans. Info. Theory, vol. IT-28, January 1982, pp. 55-67.

5.8 A. Lender: "The duobinary technique for high-speed data transmission", IEEE Trans. Communication Electronics, Vol. 82, May 1963, pp. 214-218.

5.9 P. Kabal and S. Pasupathy: "Partial-response signaling", IEEE Trans. Communications, Vol. COM-23, September 1975, pp. 921-934.

5.10 L. Hanzo, T.H. Liew, and B.L. Yeap: "Turbo Coding, Turbo Equalization and Space-Time Coding". ISBN 0-470-84726-3, John Wiley and Sons Ltd. 2002.

5.11 John G. Proakis and Masoud Salehi, “Communication Systems Engineering, Second Edition”, ISBN 0-13-095007-6, Prentice Hall Inc. 2002.

5.12 R.D. Gitlin, J.F. Hayes and S.B. Weinstein: "Data Communications Principles". ISBN 0-306-43777-5, Plenum Press 1992.

5.13 J. G. Proakis: "Digital Communications, Fourth Edition". ISBN 0-07-118183-0, McGraw-Hill 2001.

5.14 J. Kurzweil: “Digital Communications”. ISBN 0-471-15772-4, John Wiley & Sons 2000.

5.15 H. Meyr, M. Moeneclaey, and S.A. Fechtel: "Digital Communication Receivers. Synchronization, Channel Estimation and Signal Processing ". ISBN 0-471-50275-8, John Wiley & Sons 1998.

5.16 R.E. Crochiere and L.R. Rabiner: "Multirate Digital Signal Processing ". Prentice Hall, 1983.

5.17 K.H. Mueller and M. Müller: "Timing recovery in digital synchronous data receivers", IEEE Trans. Communications, Vol. COM-24, May 1976, pp. 516-531.

5.18 F.M. Gardner: "A BPSK/QPSK timing-error detector for sampled receivers", IEEE Trans. Communications, Vol. COM-34, May 1986, pp. 423-429.

5.19 ITU-T: "Recommendations from IXth Plenary Assembly, Melbourne 1988, Blue Book". ISBN 92-61-3321-2, -03341-5, -03351-2, and -3381-x, International Telecommunication Union, Geneva 1989.

5.20 E. Bødtker: "Oversigt over liniekoder til brug ved transmission af PCM signaler i telenettet". Teleteknisk Forskningslaboratorium Rapport Nr. 58. September 1979.

5.21 G. L. Cariolaro, G. L. Pierobon, and S. G. Pupolin: "Spectral analysis of variable-length coded digital signals", IEEE Trans. Information Theory, Vol. IT-28, May 1982, pp. 473-481.

5.22 F-W. Bödeker, W. Hartmann, E-U. Scheuing: “Leistungsausrüstung zur Übertragung digitaler Signale mit 34 Mbit/s auf Koaxialkabeln”. Nachrichtentechnische Zeitschrift, Band 35, August 1982, pp. 496-502, ISSN 0027-707X

5.23 Deutsche Bundespost: "Technische Richtlinie. ISDN Spezifikation der Schnittstelle U_{k0} Schicht 1, 1987.

5.24 J. Justesen: “Calculation of power spectra for block coded signals”. IEEE Trans. Communications, Vol. COM-49, March 2001, pp. 389 - 392.

5.25 A. Finnestad, T. Røste, R. Schumann-Olsen og K. P. Sundby: "ISDN totråds transmisjon", Telektronikk, Nr. 4, 1988, pp. 203-217.

5.26 ITU-T Study Group 13, "Recommendation G.828: Error Performance Parameters and Objectives for International, Constant Bit Rate Synchronous Digital Paths", Génève, Switzerland, March 2000.

INDEX

Note the English and Danish index words are mixed due to technical reasons.