Academic year
2024-2025

Faculty of Applied Engineering

Digital Signal Processing

Signals & Transforms – Text book

Walter Daems

Cursusdienst
UNIVERSITAS
Prinsesstraat 16
2000 Antwerpen
T +32 3 233 23 72
F +32 3 233 65 81
E info@cursusdienst.be
W www.universitas.be

Bachelor of Science in de Industriële Wetenschappen

1514FTIDSP 6-Digital Signal Processing

University of Antwerp

This document has been typeset using LaTeX and the `uantwerpendocs` package.
Calculations have been performed using Matlab/Octave and generic programming and script languages (C/C++/Perl/Python).
Graphics have been composed using PGF, TiKZ and InkScape.
All this material has been prepared on a GNU/Linux workstation.

Typesetting of this document was enabled by:

DSP-ST-2024-3.10-TB

Contents

Preface

The scenery

An easy classification of the art of Digital Signal Processing (DSP) is to label it as one of the subdomains of electronics.

However, DSP has its roots in the profound world of mathematics that has been developed over the past centuries. In fact, a lot of the state-of-the-art techniques originate from the days (and the minds) of the great old mathematicians like Euler, Fourier, Laplace, Lagrange and Gauss.

Indeed, understanding and applying digital signal processing techniques requires a good understanding of basic and more advanced mathematics. In this sense, digital signal processing can be rightfully claimed as being one of the domains of applied mathematics.

So, how can it be that these mathematical DSP techniques (e.g., the Fast Fourier Transform, attributed to the German mathematician Karl Friedrich Gauss (1777-1855)) only kicked in during the second half of the 20^{th} century? The answer is obvious: the development of digital electronics and more specifically the digital computer enabled the effective use of these techniques: in this way, electronics in general, can rightfully be called one of the ancestors of DSP. Many more researchers (Parks and McClellan on the work of Chebyshev, Meyer, Daubechies and Mallat on the work of Haar, ...) have contributed to build on that heritage.

The available literature on DSP is vast. Many good books exist on the subject. This work in particular was inspired by some of them you can find in the reference list at the end [Smi03, Smi08b, Smi08a, Lyo04, vdW92, OSB99, GR80, Sij04, vdEV87, MO94, Boz94, BtMvdBJvdV93, DLW06]. So, why write another introductory-level text on the very same subject? It allows treating the subjects at a level appropriate to undergraduates in Applied Engineering. There is almost no textbook available that has an appropriate mix of mathematics and engineering. Writing my own course material also allows elaborating difficult subjects based on teaching experience and updating evolving subjects swiftly.
It avoids also having to purchase multiple expensive books to cover the subject.
The reader should be aware that the mathematical treatment in this textbook is not very rigorous.

The script

This book is the second in a series of four books on electronic systems and signal processing:

1. Systems Theory — Signals and Transformations

2. Digital Signal Processing — Signals and Transformations
3. Digital Image Processing
4. Digital Signal Processing — Signal Processing Systems

This first two volumes focus on signals and signal transformations (in the analog and in the digital domain). The third volume focuses on image processing. The fourth focuses on building (digital) signal processing systems.

On the language used in this textbook: I tried to write this book from the perspective of a tutor guiding his tutees. Therefore, the text lacks the formality of many scientific text books. I hope you like this style. Where appropriate, I left out some mathematical derivations in order not to clutter the overall picture. I tried to add as much hints as needed to allow you working your way through the (sometimes difficult) material on your own if you like. The "he-him-his" formulation that has been used is not to emphasize the lack of women in engineering. This wording (instead of the more elaborate he/she, him/her and his/hers) has been chosen to keep this text simple.

A lot of effort has been put into this edition.

- This edition has been equipped with a solution book containing the solutions to the exercises. Making exercises is thé way to make sure you understand the theory. Exercises marked with (*) are a bit harder than the standard non-marked exercises. Those marked with (**) are for the enthousiastic reader (to fill any rare rainy days).
- This edition has been equipped with a formula collection that brings the important equations and definitions within reach in a convenient survey. If you think equations are missing from this collection, please inform me about this and I will consider adding them.

Though I try to avoid any errors, human erring is of all times. Do not hesitate to check with me if you find any errors. Even when you think there is an error and there's not, you will gain my appreciation for taking the exercises seriously.

Most of the material in this textbook is my original work. Some of it has been taken from other (free/open) sources. In case you notice a copyright infringement (or a reference that is not clear), please contact me. It is my firm desire to be 100% in line with the copyright legislation. If there happens to be an infringment, I appologize for it. I will do all that is reasonably possible to overcome that issue as soon as it is brought to my attention.

The crew

Finally, I would like to thank many people who contributed to this text.
First, a special thanks to my editor, Paul Levrie, for helping me by reviewing this text and supporting me with references, his profound experience, joyful humor and music. Some of the chapters (about the Laplace and the Z-transform) have been partly based on material of Paul and his former collegues Wilfried Vanneste and Gustaaf Deen.
Thanks to Maggy Goossens, Eric Paillet and Jan Steckel for bringing interesting background material to my attention.

Finally, my deepest gratitude goes to my beloved wife and children for enduring me devoting my time to writing this text, instead of spending my time with them.

Any contribution to this work is welcome. You won't get any money for it. I can only offer you to be on my list of favorite people. You can contact me by e-mail to `walter.daems@uantwerpen.be`.

I hope you enjoy discovering digital signal processing!

Walter Daems
Summer 2024
Kyndeløse, Sjælland (DK)

Symbol Table

Symbol	Meaning
Number sets	
$\mathbb{N}$	set of natural numbers (positive integer numbers)
$\mathbb{Z}$	set of integer numbers
$\mathbb{Q}$	set of rational numbers
$\mathbb{I}$	set of irrational numbers (real numbers that are not rational)
$\mathbb{R}$	set of real numbers
$\mathbb{C}$	set of complex real numbers
$\mathbb{X}^+$	set $\mathbb{X}$ restricted to positive numbers (not for $\mathbb{C}$)
$\mathbb{X}^-$	set $\mathbb{X}$ restricted to negative numbers (not for $\mathbb{C}$)
$\mathbb{X}_0$	set $\mathbb{X}$ with 0 excluded
Transform symbols - Forward	
$C_k = \text{FS}(x(t))$	C_k are the harmonic numbers of the Fourier Series of $x(t)$
$X(\omega) = \mathcal{F}(x(t))$	$X(\omega)$ is the Fourier Transform of $x(t)$
$X(\omega) = \text{DtFT}(x[n])$	$X(\omega)$ is the Discrete-time Fourier Transform of $x[n]$
$X[k] = \text{DFT}(x[n])$	$X[k]$ is the Discrete Fourier Transform of $x[n]$
$X(s) = \mathcal{L}(x(t))$	$X(s)$ is the Laplace Transform of $x(t)$
$X(z) = \mathcal{Z}(x[n])$	$X(z)$ is the Z-Transform of $x[n]$
Transform symbols - Inverse	
$x(t) = \text{FS}^{-1}(C_k)$	C_k are the harmonic numbers of the Fourier Series of $x(t)$
$x(t) = \mathcal{F}^{-1}(X(\omega))$	$X(\omega)$ is the Fourier Transform of $x(t)$
$x[n] = \text{DtFT}^{-1}(X(\omega))$	$X(\omega)$ is the Discrete-time Fourier Transform of $x[n]$
$x[n] = \text{DFT}^{-1}(X[k])$	$X[k]$ is the Discrete Fourier Transform of $x[n]$
$x(t) = \mathcal{L}^{-1}(X(s))$	$X(s)$ is the Laplace Transform of $x(t)$
$x[n] = \mathcal{Z}^{-1}(X(z))$	$X(z)$ is the Z-Transform of $x[n]$
Mapping symbols	
$x(t) \xrightarrow{\text{FS}} C_k$	C_k are the harmonic numbers of the Fourier Series of $x(t)$

continued on next page

continued from previous page

Symbol	Meaning
$x(t) \xrightarrow{\mathcal{F}} X(\omega)$	$X(\omega)$ is the Fourier Transform of $x(t)$
$x[n] \xrightarrow{\text{DtFT}} X(\omega)$	$X(\omega)$ is the Discrete-time Fourier Transform of $x[n]$
$x[n] \xrightarrow{\text{DFT}} X[k]$	$X[k]$ is the Discrete Fourier Transform of $x[n]$
$x(t) \xrightarrow{\mathcal{L}} X(s)$	$X(s)$ is the Laplace Transform of $x(t)$
$x[n] \xrightarrow{\mathcal{Z}} X(z)$	$X(z)$ is the Z-Transform of $x[n]$
DAC/ADC related acronyms	
DNL	Differential nonlinearity
INL	Integral nonlinearity
DAC	Digital to Analog Converter
ADC	Analog to Digital Converter
S/N	Signal to Noise ratio
THD	Total Harmonic Distorsion
RMS	Root-Mean Square value
SNAD	Signal to Noise and Distorsion ratio
SFDR	Spurious Free Dynamic Range

Chapter 1

Introduction

1.1 Electronics

The fundamentals of electromagnetism were discovered long before the upcome of electrical engineering. Well known scientists like Ampère, Coulomb, Faraday, Gauß, Hertz, Kirchhoff, Maxwell, Ohm, Schrödinger, Thomson and Volta, paved the way towards the first engineering applications. Simple but revolutionary inventions like electrical telegraphy (Morse), phonography (Edison), telephony (Bell) and wireless communication (Marconi) clearly indicated in the 19th century the upcome of one of the world's greatest revolutions: electronics.

During the twentieth century the inventions emerged at a continuous high pace. Enabled by the invention of the first vacuum tubes, diode (Fleming) and triode (De Forest), electronics emerged, electronic circuit analysis and design techniques evolved and found their use in the first applications to drastically change the world's face: radio (Tesla), television (Farnsworth), radar (Watson–Watt) and even computers (von Neumann & Turing).

It was the size of the first vacuum tube computers, that revealed a *first major urge* to the electronics engineering community: the need for *downscaling* size and power consumption, in order to allow increasing speed and performance.

The major breakthrough, replacing discrete vacuum tube circuits by integrated semiconductor circuits, was initiated by the invention of the bipolar junction transistor (BJT) in 1947 (Bardeen, Brattain and Shockley) and the field effect transistor (FET) in 1951 (Shockley). It should be noted that the first experiments to make a semiconductor diode took place already in 1906 (Pickard), but the resulting devices were not reliable. The planar integration of these semiconductors to form integrated circuits, the only missing piece of the puzzle, was only engineered at the end of the fifties (Noyce, Kilby, Hoerni and Moore).

A vast amount of IC technologies has been developed over the years, most of them using silicon as a substrate. The semiconductor industry has been growing since then at an unprecedented rate (an average increase of about 15% per year in semiconductor sales).

The continuous and predictive miniaturization in VLSI processing technology opened many new challenges: the increased transistor density (due to progress in lithography technology) and the decreased defect density (due to better processing conditions) enabled the design of ever more complex and ever more performant circuits.

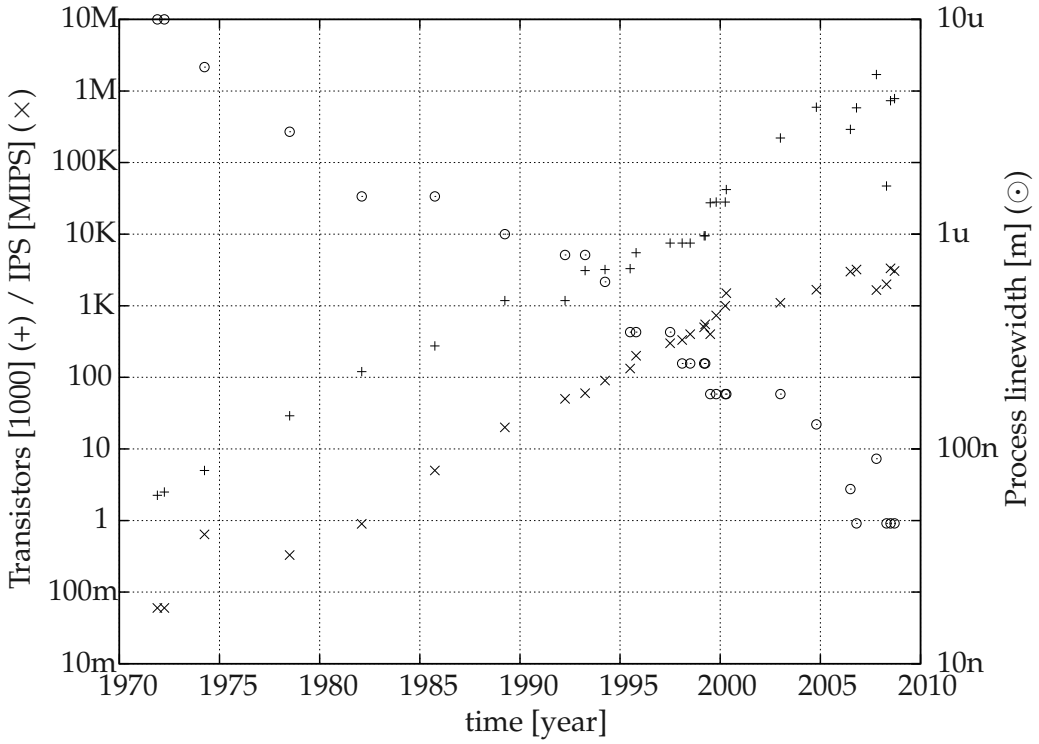


Figure 1.1: Transistor count, process line width and processor speed for the Intel microprocessor line as a function of time (*data source: various but mainly Intel Corp.*)

Figure 1.1 illustrates this. It shows the evolution of the transistor count, the processor speed and the process line width for the microprocessor line of Intel Corp. Notice how well these data confirm Moore's law (the circuit complexity increases linearly on a logarithmic scale as a function of time). The plot also illustrates the decreasing product cycle period. This is a *second major urge*, which has become the major challenge in nowadays circuit design: the *time–to–market*.

1.2 The digital (r)evolution

In the middle of this turbulent evolution on the electronics device and device processing level, a second turbulent evolution took place: the shift from analog to digital electronics.

Whereas the first electronic systems where purely analog, most recent systems now are almost fully digital. The table below lists a few typical examples:[1]

[1] In the table the transmission media, data/signal-formats and storage media have been happily intermixed. Keeping them rigorously apart can be done, but it overcomplicates the table. In addition, the more the signals appearing in the systems are digitized, the more storage and transmission medium are decoupled from the data-format.

Application	Analog system	Digital counterpart
telephony	POTS	ISDN, VoIP
radio	AM/FM	DAB
television	PAL/SECAM/NTSC	DVB
mobile telephony	(I)MTS, (N)AMPS	GSM, CDMA, UMTS, 3GPP
pictures	35mm	JPEG
video	Betamax, Video2000, VHS, Video8, Hi8	DVD, HD-DVD, Blu-ray Disc (MPEG 1,2,3,4)
camera	vidicon	CCD, CMOS image sensor
sound recording	phonograph,	CD, SACD
	reel-to-reel tape, compact-cassette	MP3, Ogg-Vorbis, WMA, RealAudio, AAC
data modems	Voiceband modems	XDSL, CableModem

So, what are the advantages that drive this digital (r)evolution? The two foundation concepts are:

- noise margin,
- signal regeneration.

The former causes a small disturbance of a signal to have no local effect (i.e., a local noise tolerance), the latter causes small disturbances to die away along a digital circuit's signal path (i.e., a global noise tolerance).

These basic concepts cause a range of benefits for digital electronics:

- insensitivity towards noise and environmental influences, making digital circuits robust and very reproducible;
- the ability to make perfectly matching structures;
- the ability to implement long-lasting memories not suffering from data deterioration.

1.3 Why digital signal processing?

Processing signals in the analog domain has been standard practice for a long time. What are the advantages of processing signals in the digital domain? The key fact here is that we are able to implement a perfect memory element. This opens the path to

- programmable circuits, ranging from microprocessor-like products (allowing different algorithms to run on the same hardware) to reconfigurable hardware (FPGAs, a.o.)[2];
- adaptive systems (that change behavior based on the data they process);
- multidimensional signal processing (e.g., processing images);
- processing signals using signal transformations (e.g., FFT convolution);
- making perfect integrators.

The latter allows

[2]Note that programmability allows reducing the time-to-market (the second major urge in electronics) for a dedicated signal processing related application. This often is referred to as "rapid prototyping".

- processing very low-frequency signals, and
- making linear-phase filters.

All these benefits were needed in the many domains that pushed DSP to becoming a breakthrough and mainstream technology:

- radar and sonar (for the war industry and later on the civil aviation industry)
- geophysical exploration (for the oil & gas industry)
- space exploration (to boldly go where...)
- medical imaging
- telecommunications
- industrial control applications
- multimedia

The end result is that while DSP used to be a specialist's graduate course, the introductory level has become a standard undergraduate course...and you are — whether you like it or not — taking it right now. Don't be mistaken, the state-of-the-art of DSP is still on the graduate level. It requires knowledge of more sophisticated mathematical concepts from the range of *functional analysis* (a.k.a. topological vector spaces) and *Statistical process theory*.

1.4 Why dedicated DSP hardware?

It is possible to run DSP algorithms on standard computing hardware (general purpose CPUs, microcontrollers, a.o.). However, processing speed is a crucial factor in our continuous need for higher-quality audio, higher-resolution video, ever more detailed exploration (of earth, human body and space) and ever increasing datarates in telecommunications.

DSP algorithms are very specific in nature (and very unlike typical computing algorithms in office or database applications). As we will see later on, very specific hardware can help us speed-up DSP algorithms (MACs, barrel shifters, special addressing units, etc.).

At the end of the 1970s, the first dedicated Digital Signal Processors were launched on the electronics market by Intel and AMI. At first not very successful, the big boom in DSPs only followed during the 1980s when Texas Instruments launched their TMS series. Around the same time Motorola Semiconductor (now Freescale Semiconductor) launched its very succesful m56K family.

Analog (!) Devices soon also entered the market with its ADSP, SHARC and later on Blackfin architectures.

In all architectures we see more and more dedicated hardware entering the picture, extending into the incorporation of full generic microprocessors on chip. This leeds to DSPs running full (real-time) operating systems (e.g., μCLinux/RTL, WindRiver VxWorks, ...).

At the same time, we see general-purpose (Intel, AMD, ...) and embedded microprocessors (ARM, Tensilica, ...) incorporating more and more DSP-specific features in hardware (MMX, SSE, ARM9E, TIE, ...).

Also traditional microcontroller vendors (Intel, Atmel, Infineon, a.o.) are equipping their microcontroller products with DSP capabilities.

Therefore, the trend seems to be that the gap in between different categories of computing hardware is diminishing. Whether the gap will reach zero remains to be seen.

1.5 Conclusion

Digital Signal Processing beats analog signal processing for very good reasons: programmability, performance that is (almost) infeasible in the analog domain (linear phase filtering, perfect integration, compression, multidimensional processing, ...).

However, concluding that analog electronics will disappear in the future, is one step too far: in the area of data conversion (from the analog to the digital domain and vice versa), high-speed communications and power electronics, analog will remain to be (inevitably) the preferred technology. High-speed analog and digital will go hand-in-hand in the mixed-signal systems of the future.

Chapter 2

Signals

In this chapter, you will learn:

- what signals are,
- how they can be represented in a graph,
- that they come in four flavors related to the concepts 'continuous and discrete quantities',
- that signal calculus is actually nothing more than function calculus,
- some important signal properties,
- howto decompose signals for some basic decomposition schemes.

After having read this chapter, a lot of questions will still be left unanswered:

- how do signal conversions from analog to digital and back, really take place?
- why are signal decompositions so important?

After having read/studied this chapter, you are expected to

- understand all the definitions and properties given here. This does not imply knowing the entire mathematical treatment by heart. There are many ways of understanding definitions and properties: being able to give a sound English description is one of them. Still, you'll find that in the conciseness' race, mathematics wins by far from English.[1]
- master the art of drawing a block diagram when given equations, and master the art of writing down the equations when given a block diagram.
- decompose signals using some basic decomposition schemes.

2.1 Definition of a signal

In the most practical sense, a signal is a relationship between two physical quantities. E.g., for the powerline entering your home (feeding your domestic appliances), this could be the AC voltage as a function of time, or the frequency (exhibiting some frequency distortion or noise around 50 Hz) as a function of time. It can also be quantities that seem less technical, e.g., the

[1] And, to be fair, English wins from Dutch, by quite a nose length.

price of your most favorite beer during your latest visit to the US, or the daily windspeed (in knots) during your surfing holiday.

More theoretically sound, we state that a signal is a mathematical function, mapping one quantity x to another y.

$$x \mapsto y$$

This strict mathematical view allows defining some terminology that allows discussing signals in a more rigorous way. The signal has an *independent* variable, associated to a particular *domain* and a *dependent* variable, associated to a certain *range*.

Graphs

Very often, signals are plotted in a graph (as functions are). We usually put the independent variable in an orthogonal graph along a horizontal axis, the so-called x-axis (or abcissa) and the dependent variable along a vertical axis, the so-called y-axis (or ordinate). Consider, e.g., the graphical representation of the observed water level at the Niagara Intake Water Station, New York, USA as a function of time (Figure 2.1). The careful reader will observe that the axes themselves are not present on the on the graph. Instead a system of bounding box plus tick marks is used to indicate the orientation, scale and values on the axes. Where mathematicians might shiver by the mere sight of this, it is quite common for scientists and engineers (and especially DSP engineers) to do so.

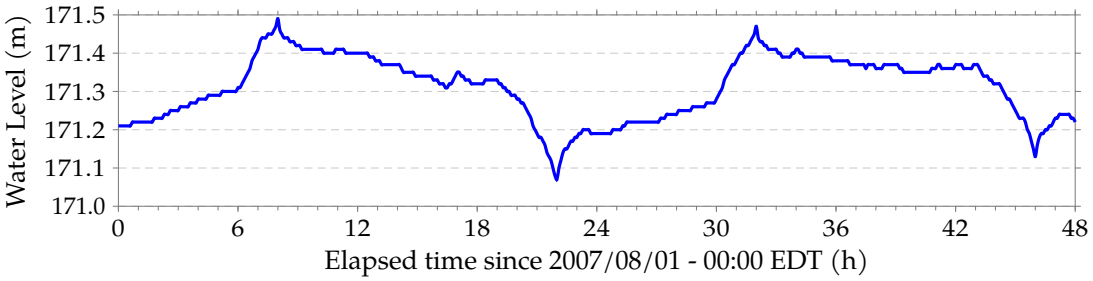


Figure 2.1: The observed waterlevel at the Niagara Intake Water Station, New York, USA, on August 1st and August 2nd, 2007 (*source: Water Level Stations Monitoring on Great Lakes Online* (*http://glakesonline.nos.noaa.gov/monitoring.html*))

Please, note the fact that in order for such a graph to become meaningful, a minimum amount of information is required. In addition to caption or title of the graph describing the relationship, the portrayed quantities, their units, and an indication of the scale, and, if relevant, one or more reference points for each axis, are minimal requirements for a graph to become a useful piece of information. Keep in mind that a graph without this minimal amount of information is *useless*. A grid may help, but very often tends to overload a graph.

Remarks Don't let the persistent appearance of time as the independent variable in the examples presented so far, make you believe that time is the only possible independent variable. Other simple parameters (e.g., frequency, sequence number) may take the role of independent variable, but so can also more complicated parameters (e.g., the coordinates of a pixel[2] in a picture, or the triplets of a voxel[3]).

[2]The word pixel is based on the contraction of picture and element.
[3]The word voxel is based on the contraction of volume and element

Also, the dependent variable can be multidimensional (e.g., luminance and chrominance in a traditional video signal), or the tuplets of temperature, static pressure and windspeed on a wheather map.
Every application has its own specific signals with their own independent and dependent quantities.

Definition
In view of all this, a signal can be mathematically written as:

$$\vec{y}(\vec{x}) : D^n \rightarrow R^m : \vec{x} \mapsto \vec{y} \tag{2.1}$$

Please, note that arranging tuplets of independent or dependent quantities into vectors, does not by default imply that the related vector space (metric, i.e. with a proper distance function defined on it, or non-metric) becomes meaningful. It might be just a convenient way of organizing the data.

The notation of (2.1) emphasises the fact that a signal has a domain and a range, which (especially in DSP) is an important observation. It leads us to the next section on continuous versus discrete signals.

2.2 Continuous vs. discrete signals

Though fundamental scientists may (rightfully) argue this, we may safely consider most of the quantities in the real world to be continuous. In short, we may state that the world surrounding us is analog — (and to keep the scientists happy, let's add a single refinement) — on a macroscopic scale.

The term analog dates from the time where scientists tried to "solve" (more precisely, simulate) differential equations by building an *analog* electronic system, providing it with the appropriate inputs and seeing how the system would evolve in time, or to what equilibrium it would converge. Though simulating differential equations by building analog electronic systems is outdated, the term *analog* in the mean time did get its own life, and has become the adult brother of the term *digital* (stemming from the Latin 'digitus', a finger; it suggests a countable (discrete) quantity). But what exactly does analog and digital mean in the world of DSP? Let's state a more exact definition.

As stated before, a signal is characterized by an independent quantity and a dependent quantity. Both of them can be continuous or discrete.

This leads us to four different categories of signals we may encounter. These have been illustrated in Table 2.1 on the following page.

Classification
A signal with continuous independent and dependent quantities is called an *analog* signal. A signal with discrete independent and dependent quantities is called a *digital* signal. Signals with discrete dependent quantities are called *quantized* signals. Signal with discrete independent quantities are called *discrete-time* (as opposed to continuous-time signals) or *sampled* signals.

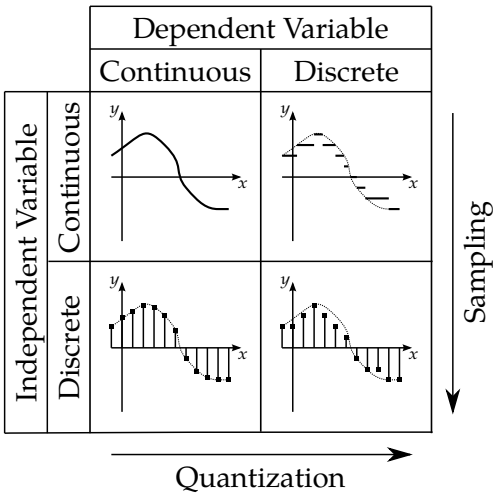


Table 2.1: Continuous vs. discrete representation of signals

Different nomenclatures exist in different books on DSP. Some authors call a digital signal a *sampled-data* signal, whereas others use this term for a *sampled* signal (with or without quantization). Therefore, let's stick to the terminology above: it is simple, common and straightforward.

Notation for discrete-time signals

Though we have no special notation for quantized signals, we do have one for sampled signals. Let's consider a continuous signal $x(t)$.
Sampling means: we only consider the values at discrete time points t_i:

$$t_i = i\Delta t$$

with i an integer number and Δt the sampling interval. This leads to a discrete-time signal for which we use a notation with square brackets to stress the fact that it is not a continuous-time signal:

$$x[i] \equiv x(t_i) = x(i\Delta t)$$

Remarks Table 2.1 and the equations above suggest that sampling is uniform[4] in the independent variable and quantization is uniform in the dependent variable. Though this is often the case, it is no prerequisite. Some variable bitrate audio codec systems are examples for the former, A-law and μ-law companding schemes (see section 5.5.1.1 on page 146) are obvious examples of the latter.

Graphs of discrete-time signals

Though we have no special graph notation for quantized signals, we do have one for sampled signals in general. As we only have information about the signal at discrete moments in time, we only plot the signal's values at these moments in time, using a firm dot. Usually (but not necessarily), we support the dot with a stem starting from the independent variable's axis, not because it might fall down otherwise, but to suggest the idea of an impulse (a concept that will become clear later).

[4]Here, uniform means the sampling points or the quantization levels are equidistant.

Figure 2.2 displays the waterlevel of the Niagara Intake Water Station (see Figure 2.1 on page 8) after it has been sampled with an interval of 1 hour.

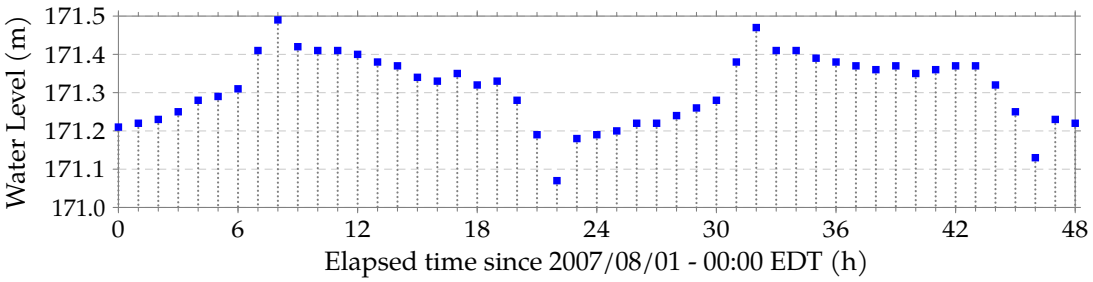


Figure 2.2: The observed sampled waterlevel at the Niagara Intake Water Station, New York, USA, on August 1st, 2007, using a sample period of 1 hour. (*source: Water Level Stations Monitoring on Great Lakes Online* (*http://glakesonline.nos.noaa.gov/monitoring.html*))

Remarks Don't be tempted to connect the dots of discrete-time signal graph by straight lines, a smooth curve, or a staircase line. The only thing we know about the sampled signal are its sample-point values. The signal is nothing but a list of numbers.

2.3 Converting signals: a first primer

Analog to digital

Discretizing the independent variable is called *sampling*. Discretizing the dependent variable is called *quantization*. The combination of both is called *analog to digital conversion*. The device that carries out such a transformation is called an *Analog to Digital Converter* (*ADC*). Notice that in the quantization process (that corresponds to rounding or truncation to a grid), information is lost, that can never be recovered! In fact, the quantization error can be considered to be a noise component superimposed on the signal.

The same issue arises with the intersample information lost during sampling. However, by taking proper precautions, the sampling information loss can be avoided. We will come back to this issue later.

Digital to analog

The process of transforming a digital signal into an analog one is called *reconstruction* or *digital to analog conversion*. The device that carries out such a reconstruction is called a *Digital to Analog Converter* (*DAC*).

Why convert?

The need for sampling is obvious. Our digital computing hardware just cannot deal with an infinite flow of information. Both processing speed and memory only come in limited amounts. We therefore need to chop the information (the signal) into a discretized (sampled) version that is manageable by our digital hardware. The same holds for quantization. Our computers store numbers using a finite precision. Increasing precision comes at a price:

slower processing speed. As you will experience later, precision vs. speed is an important trade-off in DSP.
If converting analog signals to digital signals implies losing information, then why bother? One would be tempted to say that analog signals are infinitely precise and that one can cram information infinitely close together in an analog signal. In theory, that's true. In practice however, analog signal processing systems suffer from noise and are bandwidth limited. There are other reasons:

- In DSP processors, noise and temperature tolerance, have been taken care of by the DSP processor design team. The DSP application's engineer therefore can focus on his application, without having to care about robustness w.r.t. operational parameters. When designing an analog signal processing system, the burden is yours![5]
- In the digital domain, mathematics rules, instead of device physics. This allows playing some tricks that are just impossible to imagine in the world of analog electronics.

Analog signal processing, however, is and will always be an important part of signal processing in general. For the very high frequency applications like transceiver frontends and for the very enabler of DSP, AD and DA conversion, analog will be there forever.

In addition, in order to fully understand DSP one cannot go without a good knowledge of the analog electronics appearing in DSP systems.

2.4 Signal operations

2.4.1 Operations

If signals are just (continuous or discrete) functions, then the whole myriad of possible operations on functions can be applied to any of these signals.

- unary operations
  - $\pm, |\cdot|$
  - $\exp(\cdot)$, $\ln(\cdot)$
  - $\sin(\cdot), \cos(\cdot), \tan(\cdot)$
  - integration (running sum), differentation (finite difference)
  - substitution of the independent variable
  - ...
- binary operations
  - multiplication, division
  - addition, subtraction
  - convolution, correlation

[5] To be fair: the noise problem is not totally gone, the DSP application engineer will have to deal with the limited numerical precision (round-off errors), which can be a pain in the neck too.

- ...

Of course any combination or any inverse of these functions is also a valid signal operation.

2.4.2 Signal processing block diagrams

As opposed to mathematicians, engineers are less abstract thinkers: we like to draw systems using a set of symbols more than writing down the composing equations.

This leads to the well known concept of *signal processing block diagrams*. In these block diagrams

- signals are represented by an arrow labeled with the signal's mathematical shorthand, and
- operations are represented by a (small) graphical symbol, a block (hence the name).

Usually, the inputs of a block diagram are on the left-hand side of the diagram, while the outputs are on the right-hand side of the diagram. You'll notice that in very complex systems, this rule cannot be maintained without compromising the *good looks* of your drawing. Just use your common sense when drawing a block diagram.

The symbols that are commonly used in these diagrams are depicted in Table 2.2 on the following page.

Exercises
As this is a course on DSP, the exercises are all discrete-time.

Exercise 2.4.2-1: Draw a block diagram to represent the following system of equations, relating the input of the system $x[n]$ to the output $y[n]$:

$$y[n] = 5x[n] + z[n]$$
$$z[n] = 3.14x[n] - 2y[n]$$

Exercise 2.4.2-2: Draw a block diagram to represent the system of exercise 2.4.2-1 relating the input $x[n]$ to the output $z[n]$.

Exercise 2.4.2-3: Draw a block diagram to represent the system of exercise 2.4.2-1 relating the input $y[n]$ to the output $x[n]$.

Exercise 2.4.2-4: Draw a block diagram to represent the following system of equations, relating the input of the system $x[n]$ to the output $y[n]$:

$$y[n] = x[n-3] + 5x[n-2] - 3x[n-1] + 2.3x[n]$$

Exercise 2.4.2-5: Draw a block diagram to represent the following system of equations, relating the input of the system $x[n]$ to the output $y[n]$:

$$y[n] = 4x[n-3] + 2.3x[n] - 0.5y[n-1]$$

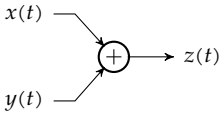
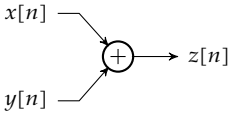
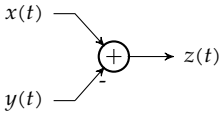
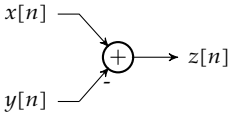
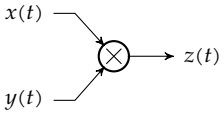
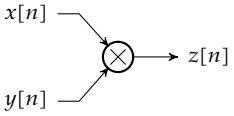
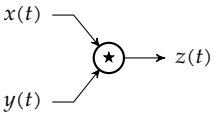
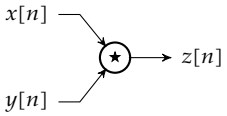
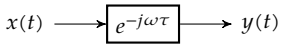
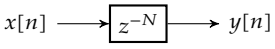
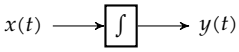
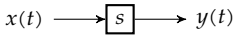
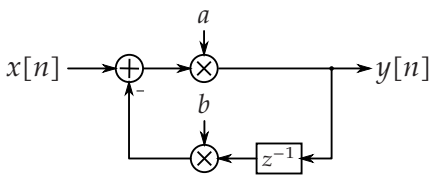
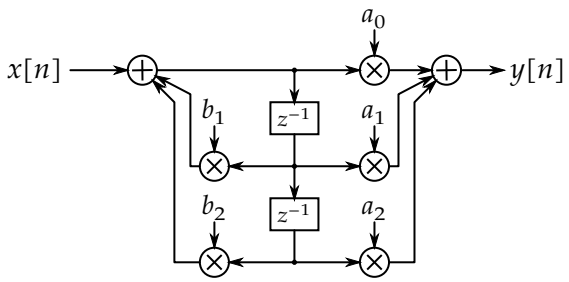
Addition	
	
$z(t) = x(t) + y(t)$	$z[n] = x[n] + y[n]$
Subtraction	
	
$z(t) = x(t) - y(t)$	$z[n] = x[n] - y[n]$
Multiplication	
	
$z(t) = x(t) \cdot y(t)$	$z[n] = x[n] \cdot y[n]$
Convolution	
	
$z(t) = x(t) \star y(t)$	$z[n] = x[n] \star y[n]$
Delay	
	
$y(t) = x(t - \tau)$	$y[n] = x[n - N]$
Unary Operation	
	
$y(t) = \mathtt{OP}\,(x(t))$	$y[n] = \mathtt{OP}\,(x[n])$
Integral / Sum	
	
$y(t) = \int_{-\infty}^{t} x(\tau)\mathrm{d}\tau$	$y[n] = \sum_{i=-\infty}^{n} x[i]$
Differential / Difference	
	
$y(t) = \frac{\mathrm{d}(x(t))}{\mathrm{d}t}$	$y[n] = x[n] - x[n-1]$

Table 2.2: Commonly used symbols in signal processing block diagrams

Exercise 2.4.2-6: Write down the equations corresponding to the block diagram below.

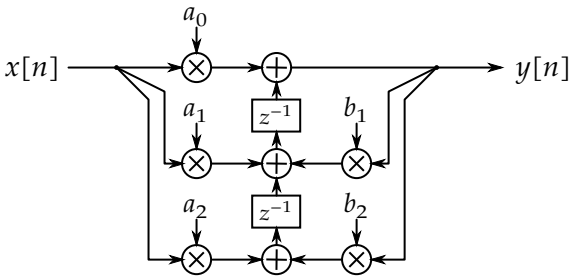


Exercise 2.4.2-7: Write down the equations corresponding to the block diagram below.



Hint: introduce new signal variables for intermediate nodes to divide and conquer the problem.

Exercise 2.4.2-8: Write down the equations corresponding to the block diagram below.



Hint: introduce new signal variables for intermediate nodes to divide and conquer the problem.

2.5 Elementary signal properties

2.5.1 Time limited vs. time unlimited signals

Continuous

A signal $x(t)$ is called *time limited* if it only has nonzero values for a limited interval in time. Often we also denote this signal as a *finite-support* signal. Mathematically:

$$\exists t_0, t_1 \in \operatorname{dom} x : t \notin [t_0, t_1] \Rightarrow x(t) = 0$$

If we choose t_0 and t_1 such that $t_1 - t_0$ is minimal, then we call the interval $[t_0, t_1]$, the support or the time span of the signal.

A signal that is not time limited, is time unlimited.

Discrete

A signal $x[n]$ is called time limited if it only has nonzero values for a limited index interval. Often we also denote this signal as a *finite-support* signal. Mathematically:

$$\exists n_0, n_1 \in \operatorname{dom} x : n \notin [n_0 : n_1] \Rightarrow x[n] = 0$$

If we choose n_0 and n_1 such that $n_1 - n_0$ is minimal, then we call the interval $[n_0, n_1]$, the support or the time span of the signal.

Again, a signal that is not time limited, is time unlimited.

2.5.2 Periodic vs. aperiodic signals

Continuous

A signal $x(t)$ is periodic if and only if:

$$\exists T \in \mathbb{R}_0^+, \forall t \in \operatorname{dom} x : x(t) = x(t+T)$$

The minimal value for T that exists is called the period of the signal. A signal that is not periodic is called aperiodic.

Discrete

A signal $x[n]$ is periodic with period N if and only if:

$$\exists N \in \mathbb{Z}_0^+, \forall n \in \operatorname{dom} x : x[n] = x[n+N]$$

The minimal value for N that exists is called the period of the signal. Again, a signal that is not periodic is called aperiodic.

2.5.3 Even/Odd signals

Though even and odd signals will rarely occur in real-life conditions, oddness and evenness are interesting theoretical properties a signal can exhibit. Later on, we will see how to exploit them to our benefit.

Continuous

A continuous signal $x(t)$ is called *even* if and only if

$$\forall t \in \operatorname{dom} x : x(t) = x(-t)$$

A continuous signal $x(t)$ is called *odd* if and only if

$$\forall t \in \operatorname{dom} x : x(t) = -x(-t)$$

Obvious examples of signals that exhibit these properties are sinusoids. Every sine of Figure 2.3 on page 24 is an odd signal. Every cosine of Figure 2.3 on page 24 is an even signal.

Discrete

A discrete signal $x[n]$ is called *even* if and only if

$$\forall n \in \operatorname{dom} x : x[n] = x[-n] \tag{2.2}$$

A discrete signal $x[n]$ is called *odd* if and only if

$$\forall n \in \operatorname{dom} x : x[n] = -x[-n] \tag{2.3}$$

Again, obvious examples of signals that exhibit these properties are discrete sinusoids. Every discrete sine of Figure 2.4 on page 25 is an odd signal. Every discrete cosine of Figure 2.4 on page 25 is an even signal.

Remarks

- Hidden in these definitions are three extra requirements:
  - the domain of the signal must be defined symmetrically around $t = 0$;
  - for odd signals, the range of the signal must be defined symmetrically around $y = 0$;
  - for odd signals, the sample at position zero must be zero.
- When dealing with discrete signals, we often deal with time-limited signals (i.e. all zero outside the "window" in which the signal appears). Very often, the nonzero sample sequence of these signals is numbered from index $n = 0$ to index $n = N - 1$. If we (artificially) make these signals periodic, by repeating the sequence over and over again, i.e.

  $$\forall k \in \mathbb{Z}, \forall n = 0 \ldots N - 1 : x[n + kN] = x[n]$$

  and the resulting signal is odd or even, then we also call the original time-limited signal odd or even. In this view, the respective equations (2.2) and (2.3) can be rewritten to be:

  $$\forall n \in [0 : N - 1] : x[n] = x[N - n]$$

  and

  $$\forall n \in [0 : N - 1] : x[n] = -x[N - n]$$

  Though of less practical use, one can also make the same reasoning for time-limited continuous signals.

2.5.4 Linear/Nonlinear signals

So, which signals are linear and which are nonlinear? Actually, this is a bit of a misnomer. While we can make such a distinction for pure mathematical functions (referring to the relationship between independent and dependent variable), we usually don't do so in the context of DSP.

A signal's just a signal.

2.6 Signal characteristics

Though many of the properties we listed above have some numerical aspect (e.g., the period for periodic signals), they are analytical (classification) properties that are no good for *measuring* an arbitrary signal. Let's define some metrics that allow us to tell something about a signal in a quantitative fashion.

We consider time-limited windows of the complete signals, as we are not able to measure infinitely long. This means that our measurement is only an estimate based on taking a *statistical sample* of the signal. We measure a sample, and we believe that it is a measure of the underlying process.

The difference "process (i.e., probability) vs. sample (i.e. statistics)" is an important difference to keep in mind.

2.6.1 Peak-to-peak value

The peak-to-peak value of a signal is easily defined.

Continuous

$$x_{PTP} = \max_{0 \le t \le T} x(t) - \min_{0 \le t \le T} x(t)$$

Discrete

$$x_{PTP} = \max_{0 \le n < N} x[n] - \min_{0 \le n < N} x[n]$$

2.6.2 Mean

This metric is also regularly referred to as the *DC-value* of a signal.

2.6.2.1 Statistics mean

If we know nothing about the process governing an unknown signal, then the only thing we can do is observe and measure the signal. Based on these measurements we can derive the statistical average or *mean*:

Continuous

$$x_{MEAN} = \frac{1}{T} \int_0^T x(t)\, \mathrm{d}t$$

Discrete

$$x_{MEAN} = \frac{1}{N} \sum_{n=0}^{N-1} x[n]$$

2.6.2.2 Process mean

If you know something about the process governing the signal (e.g., if it is a noise signal of which we know the governing physics), you may calculate the *process mean* instead. In this case, we need to make a distinction between continuously valued signals and quantized signals:[6]

[6] For the statistics purists: we don't go in to more theoretical discussions concerning stationarity, ergodicity and regularity of the signals. We consider our signals to be stationary and ergodic. If you don't know what this means:

Continuously valued signals

Let's denote the *probability density function* as $f_x(x)$.

$$\mu_x = \int_{-\infty}^{+\infty} x f_x(x)\,\mathrm{d}x$$

Quantized signals

Let's assume our quantized signal is composed of values x_i and let's denote the *probability mass function* as $f_x(x)$.

$$\mu_x = \sum_{i=-\infty}^{+\infty} x_i f_x(x_i)$$

2.6.3 Variance

This metric is regularly referred to as the *mean energy content* of the signal. One might also consider it's square root, the *standard deviation*. The standard deviation of a noise signal is often referred to as the magnitude of the noise signal.

2.6.3.1 Statistics variance

If we know nothing about the process governing an unknown signal, then the only thing we can do is observe and measure the signal. Based on these measurements we can derive the statistical average squared deviation or *variance*:

Continuous

$$x_{VAR}^2 = \frac{1}{T}\int_0^T (x(t) - x_{MEAN})^2\,\mathrm{d}t$$

Discrete

$$x_{VAR}^2 = \frac{1}{N-1}\sum_{n=0}^{N-1} (x[n] - x_{MEAN})^2$$

The factor $N-1$ is due to calculating a sample variance (using a sample mean value) instead of calculating the variance using a known process mean. For large N this difference becomes negligible.

Process variance

If you know something about the process governing the signal (e.g., if it is a noise signal), you may calculate the *process variance* instead. In this case, we need to make a distinction between continuously valued signals and quantized signals:[6]

Continuously valued signals

Let's denote the probability density function as $f_x(x)$.

$$\sigma_x^2 = \int_{-\infty}^{+\infty} (x - \mu_x)^2 f_x(x)\,\mathrm{d}x$$

Quantized signals

Let's assume our quantized signal is composed of values x_i and let's denote the probability mass function as $f_x(x)$.

$$\sigma_x^2 = \sum_{i=-\infty}^{+\infty} (x_i - \mu_x)^2 f_x(x_i)$$

2.6.4 RMS-value

Though the RMS value is regularly being mixed-up with the standard deviation, this value also should take into account the DC component.

[6] don't worry, it means one problem less in your life.

Continuous

$$x_{RMS} = \sqrt{\frac{1}{T}\int_0^T x^2(t)\,\mathrm{d}t} = \sqrt{x_{MEAN}^2 + x_{VAR}^2}$$

Discrete

$$x_{RMS} = \sqrt{\frac{1}{N-1}\sum_0^{N-1} x^2[n]} = \sqrt{x_{MEAN}^2 + x_{VAR}^2}$$

Exercises
Some exercises on signal characteristics

Exercise 2.6-1: Calculate the peak-to-peak value, the mean value, the variance and the RMS value of the following signal:

$$x(t) = 0.3\sin(20t) - 0.25\cos(10t)$$

Exercise 2.6-2: Calculate the peak-to-peak value, the mean value, the variance and the RMS value of the following signal:

$$x[n] = 0.3\sin[20n] - 0.25\cos[10n]$$

Use Octave or Matlab to execute the calculations.

Exercise 2.6-3: Calculate the peak-to-peak value, the statistics mean value, the statistics variance and the RMS value of the following signal:

$$x[n] = [-0.3, 0.4, 0.2, -0.3, 0.5, 0.7, 0.25, 0.1, -0.3, -0.4, -0.2, 0]$$

2.7 Example signals

To further illustrate the *signal* concept, let's walk through some examples that occur regularly when dealing with DSP theory. In the examples below, we assume t to be a real number, and n to be an integer number.

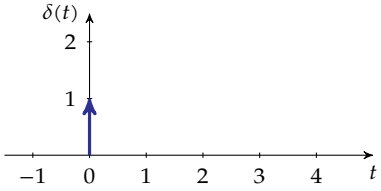
2.7.1 Unit impulse (Dirac function)

The unit impulse (or Dirac function) is probably the most important theoretical function we will ever encounter. Whereas the analog definition seems involved, the digital version is as simple as can be. We denote the unit impulse with the symbol δ.

Continuous

Let's define the Dirac function as a limit case:

$$\delta(t) = \lim_{d \to +\infty} \begin{cases} d & \text{if } |t| \leq \frac{1}{2d} \\ 0 & \text{if } |t| > \frac{1}{2d} \end{cases}$$



In the end, this boils down to a more practical form:

$$\delta(t) = \begin{cases} 0 & \text{if } t \neq 0 \\ +\infty & \text{if } t = 0 \end{cases}$$

with the boundary condition that

$$\int_{-\infty}^{+\infty} \delta(t)\,\mathrm{d}t = 1.$$

A graphical representation can be found on the right.

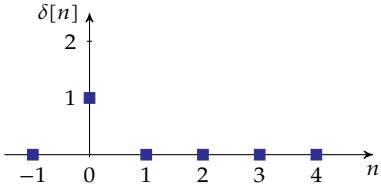
One important property of the Dirac impulse is to be noted: for any bounded function $f(t)$, and any τ in it's domain, one can state that:

$$f(\tau) = \int_{-\infty}^{+\infty} \delta(t-\tau) f(t)\,\mathrm{d}t \tag{2.4}$$

This the so-called *sifting property* of the Dirac impulse.

Discrete

$$\delta[n] = \begin{cases} 0 & \text{if } n \neq 0 \\ 1 & \text{if } n = 0 \end{cases}$$



Though it is not required for the definition of the discrete Dirac impulse, note that also in this case, the following property holds:

$$\sum_{n=-\infty}^{+\infty} \delta[n] = 1$$

As one might expect, the *sifting property* also holds for the discrete Dirac impulse.

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

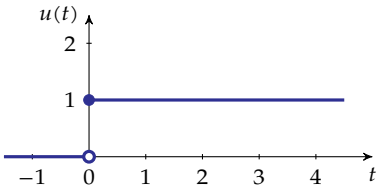
2.7.2 Unit step (Heaviside function)

We denote the unit step with the symbol u_0 or u for short. Note that the unit step function is the integral of the continuous Dirac function (for the continuous version) and the sum of the discrete Dirac function (for the discrete version).

Continuous

$$u(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t \geq 0 \end{cases}$$

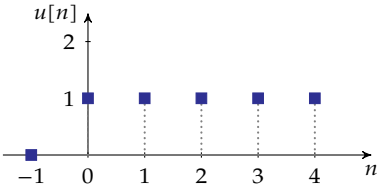
A graphical representation can be found on the right.



Discrete

$$u[n] = \begin{cases} 0 & \text{if } n < 0 \\ 1 & \text{if } n \geq 0 \end{cases}$$

A graphical representation can be found on the right.



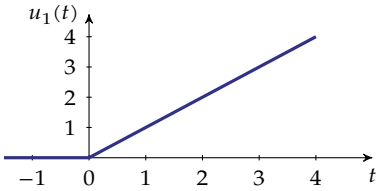
2.7.3 Unit ramps (generalized Heaviside functions)

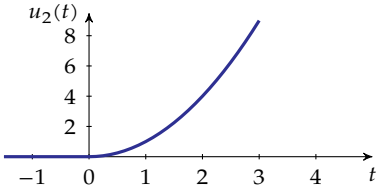
We denote the unit ramps with the symbol u_i. As can be easily concluded below, the unit step is actually the 0-th order unit ramp.[7] The term 'generalized Heaviside functions' is rather rare.

Continuous

$$u_i(t) = \begin{cases} 0 & \text{if } t < 0 \\ t^i & \text{if } t \geq 0 \end{cases} \tag{2.5}$$

Graphical representations for the cases $i = 1$ and $i = 2$ can be found on the right.

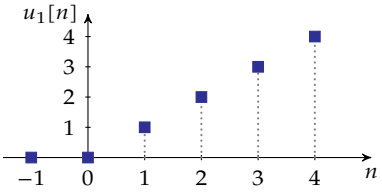




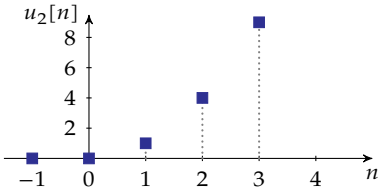
Discrete

$$u_i[n] = \begin{cases} 0 & \text{if } n < 0 \\ n^i & \text{if } n \geq 0 \end{cases} \tag{2.6}$$

Graphical representations for the cases $i = 1$ and $i = 2$ can be found on the right.



[7] In the strict sense '0 to the power of 0' is undefined and therefore the definitions of (2.5) and (2.6) cannot be used as such for time t or n equaling zero. However, let's assume in this context that we consider '0 to the power of 0' to be equal to one.



2.7.4 Sinusoids

The use of sinusoids in DSP is quite common, so let's take a first look at them.

Continuous

$$f(t) = \cos(\omega t)$$
$$g(t) = \sin(\omega t)$$

Assuming $\omega_0 = 2\pi/N$ with $N = 16$, graphical representations for the cases (a) $\omega = \omega_0$, (b) $\omega = 2\omega_0$, (c) $\omega = 4\omega_0$, (d) $\omega = 8\omega_0$, (e) $\omega = 12\omega_0$, (f) $\omega = 14\omega_0$, and (g) $\omega = 15\omega_0$ have been plotted in Figure 2.3 on the next page. Plotting so many cases seems rediculous, but in a minute, when we'll take a look at Figure 2.4 on page 25 (hold your horses for now), the relevance will become clear.

Discrete

$$f[k] = \cos[\omega k]$$
$$g[k] = \sin[\omega k]$$

Assuming $\omega_0 = 2\pi/N$ with $N = 16$, graphical representations for the cases (a) $\omega = \omega_0$, (b) $\omega = 2\omega_0$, (c) $\omega = 4\omega_0$, (d) $\omega = 8\omega_0$, (e) $\omega = 12\omega_0$, (f) $\omega = 14\omega_0$, and (g) $\omega = 15\omega_0$ have been plotted in Figure 2.4 on page 25. The originating continuous sinusoids can be observed in Figure 2.3 on the following page. Spend some time comparing the two figures. Then, for the discrete time case, pay attention to case (e), (f) and (g) and compare them to cases (a), (b) and (c). Except for the vertical axis' label, you will observe no difference! This is first occurence of the phenomenon of "frequency aliasing". We will not investigate this further for now, as analyzing and understanding this phenomenon is much easier if we can throw some more rigorous mathematics at it. We'll do this in chapter 3 on page 35.

2.8 Decomposition

In the real world, signals are nothing like the clean mathematical signals as we've seen them in section 2.7 on page 20.

Still, these theoretical, clean signals are easy to work with and allow to exploit a wealth of theoretical properties. So, would it be possible to decompose a real world signal into these more clean, basic signals? Yes, it is.

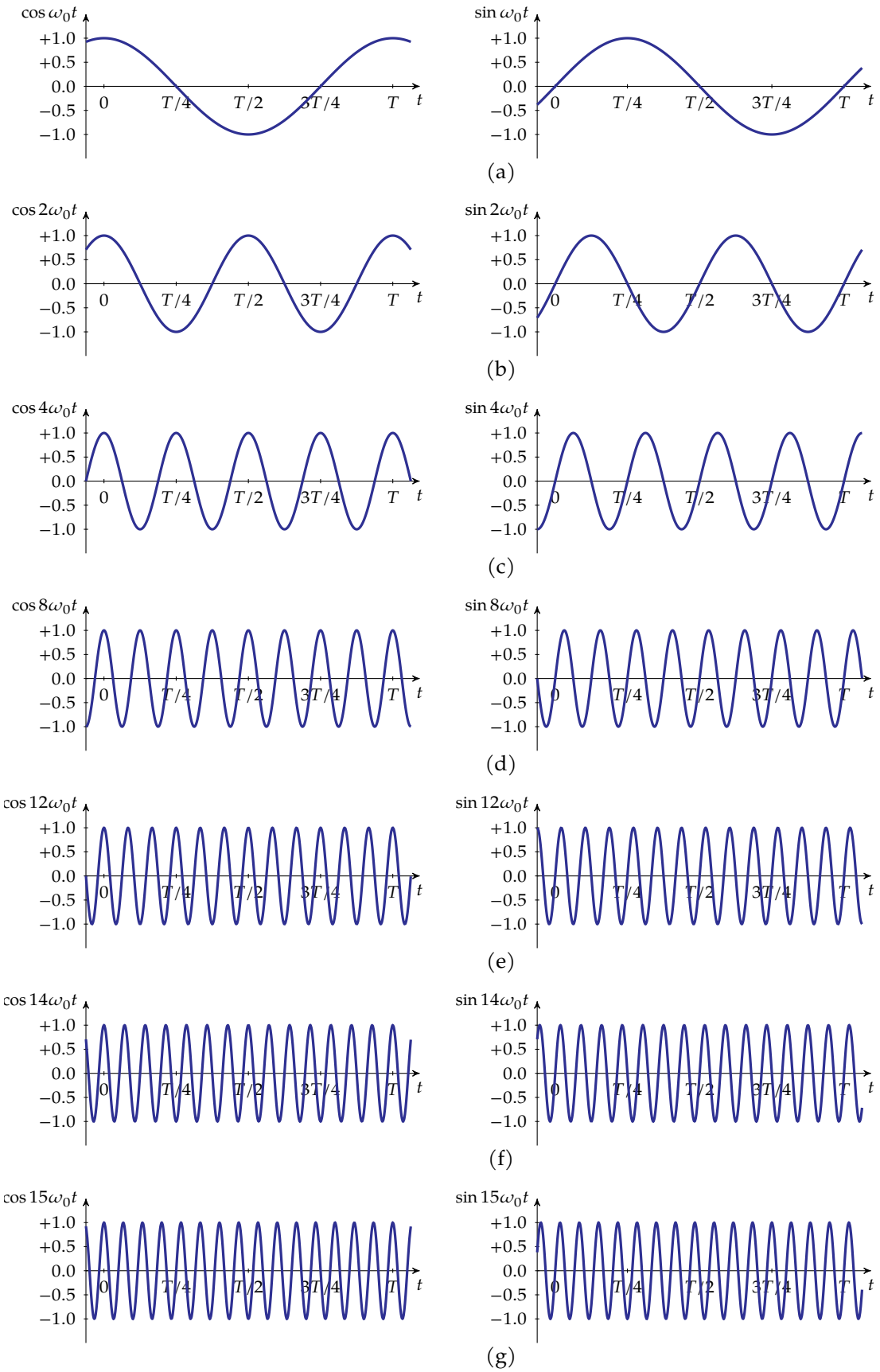


Figure 2.3: Continuous sinusoidal signals assuming $\omega_0 = 2\pi/T$ for the cases (a) $\omega = \omega_0$, (b) $\omega = 2\omega_0$, (c) $\omega = 4\omega_0$, (d) $\omega = 8\omega_0$, (e) $\omega = 12\omega_0$, (f) $\omega = 14\omega_0$, and (g) $\omega = 15\omega_0$.

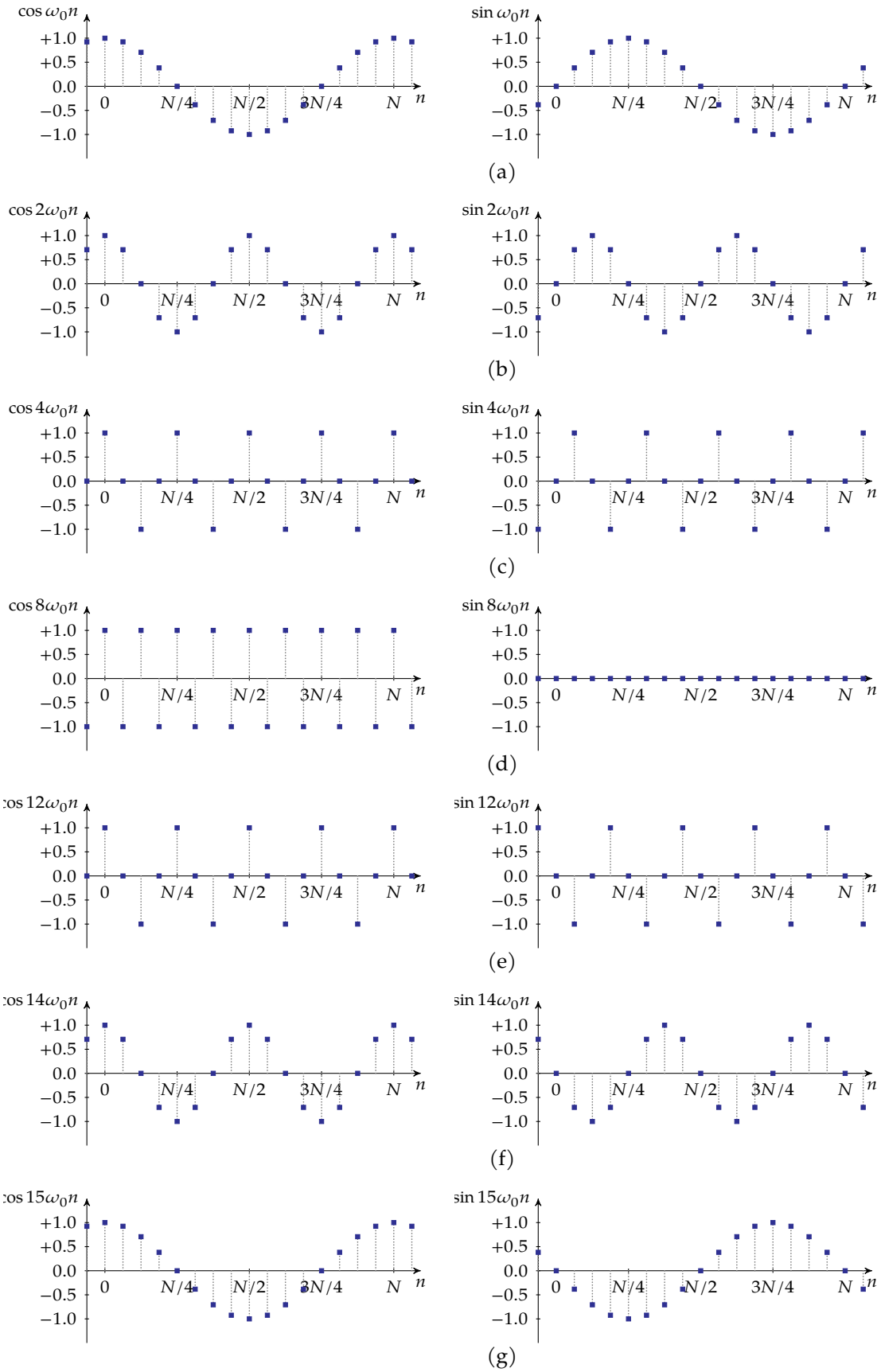


Figure 2.4: Discrete sinusoidal signals assuming $N = 16$, $\omega_0 = 2\pi/N$ for the cases (a) $\omega = \omega_0$, (b) $\omega = 2\omega_0$, (c) $\omega = 4\omega_0$, (d) $\omega = 8\omega_0$, (e) $\omega = 12\omega_0$, (f) $\omega = 14\omega_0$, and (g) $\omega = 15\omega_0$.

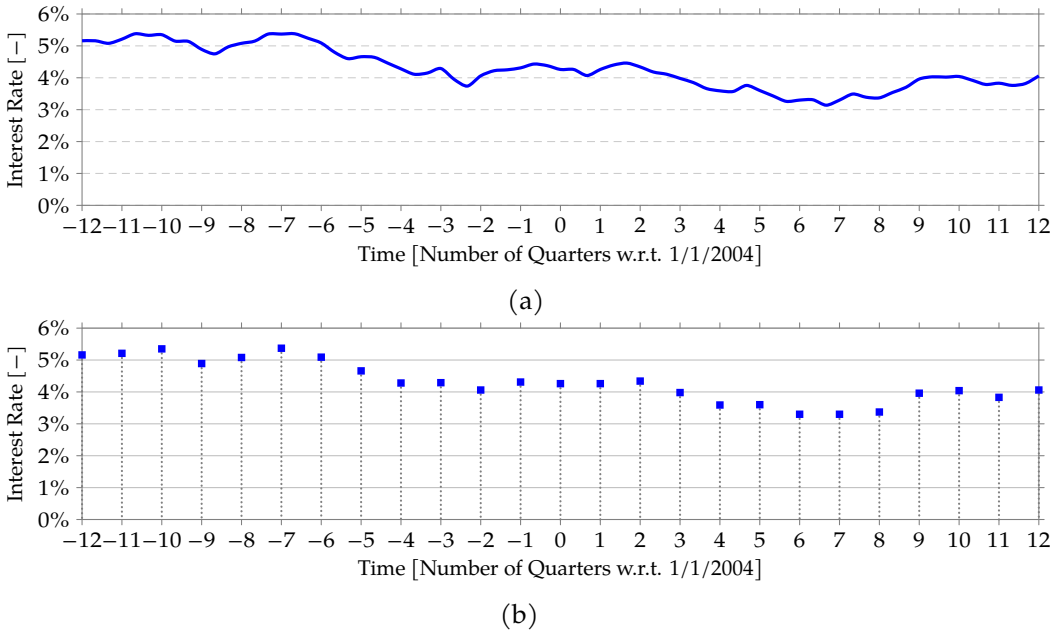


Figure 2.5: Long-term interest rate in Belgium around Q1/2004, (a) represented as a continuous-time signal, and (b) represented as a discrete-time signal (*source: European Central Bank* (*http://www.ecb.int*))

We will treat a number of possible decompositions (whose usefulness will only become clear later on). In order to keep the discussion compact and clear, we will use a single example to guide us. The data we're using is the long-term interest rate in Belgium around Q1/2004. We don't have a function recipe for the example signal at our disposition. Unsurprisingly, as you will rarely encounter a real world signal that comes with a function recipe attached to it.

The continuous time signal, of which we only have a limited graph available, is depicted in Figure 2.5(a), and its discrete-time twin's graph, obtained by sampling the signal on the first day of the quarter can be found in Figure 2.5(b).

2.8.1 Impulse decomposition (natural decomposition)

Continuous

The continuous impulse decomposition is rather abstract matter. It can be derived directly from (2.4). Check if you understand the mathematics below. If not, first, take a look at the discrete case, and start from there.

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

In this form, it looks rather trivial. The true meaning of this decomposition will become clear later on.

Discrete

On the contrary, the discrete impulse decomposition is as simple as can be. First take a look at Figure 2.6 on the next page.

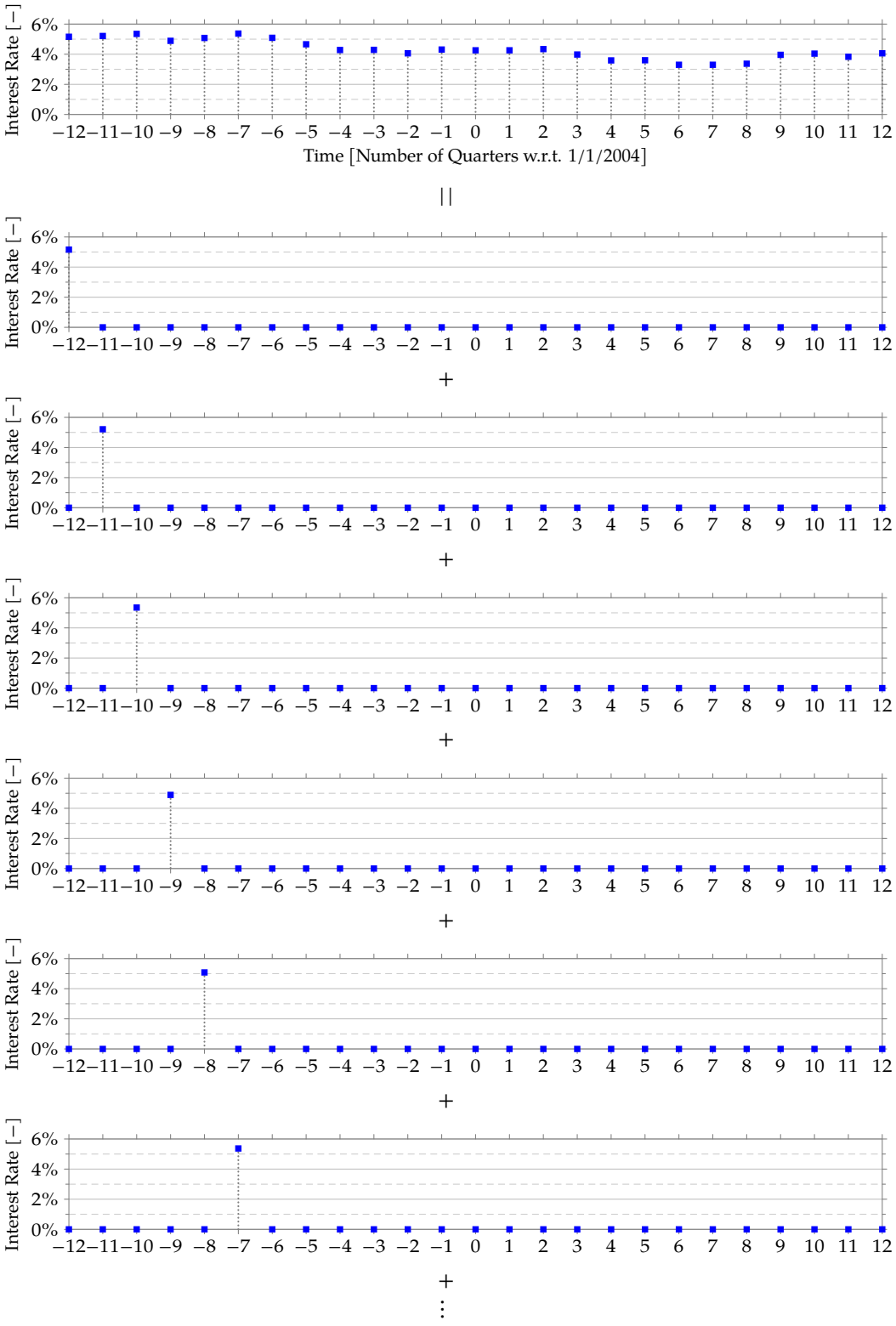


Figure 2.6: Illustration of the discrete impulse decomposition using the signal of Figure 2.5(b).

Mathematically, this boils down to:

$$f[n] = \sum_{i=-\infty}^{+\infty} \delta[n-i] f[i]$$

Because the impulse decomposition is the most natural decomposition of all decompositions, we also call it the *natural decomposition*.

2.8.2 Frame decomposition

Processing a signal by isolating single impulses is one end of the ballpark, treating the entire signal as a whole is the other end. Midway, we end up with frame decomposition.

What we actually do is cut the time-axis into limited time windows. This might be because any of the following reasons:

- the application's signal data only becomes available piece by piece
- the different pieces contain different information that needs to be separated anyway (e.g., in a Time Division Multiplex system)
- because some DSP techniques require processing limited time-windows (e.g., FFT convolution)
- because of memory constraints (or less obvious processing speed constraints): our system may not be able to store the entire signal as a whole.

Continuous

Instead of trying to distill some useless mathematics for this decomposition, let's take a look at Figure 2.7 on the facing page that illustrates this frame decomposition.

Discrete

Also in this case, let's take a look at the graphical example in Figure 2.8 on page 30.

2.8.3 Interlaced decomposition

Continuous

The interlaced decomposition does not exist for continuous-time signals.

Discrete

The discrete-time interlaced decomposition pulls the signal apart in two parts:

- the samples on even timepoints
- the samples on odd timepoints

Mathematically:

$$f[n] = f_{2n}[n] + f_{2n+1}[n]$$

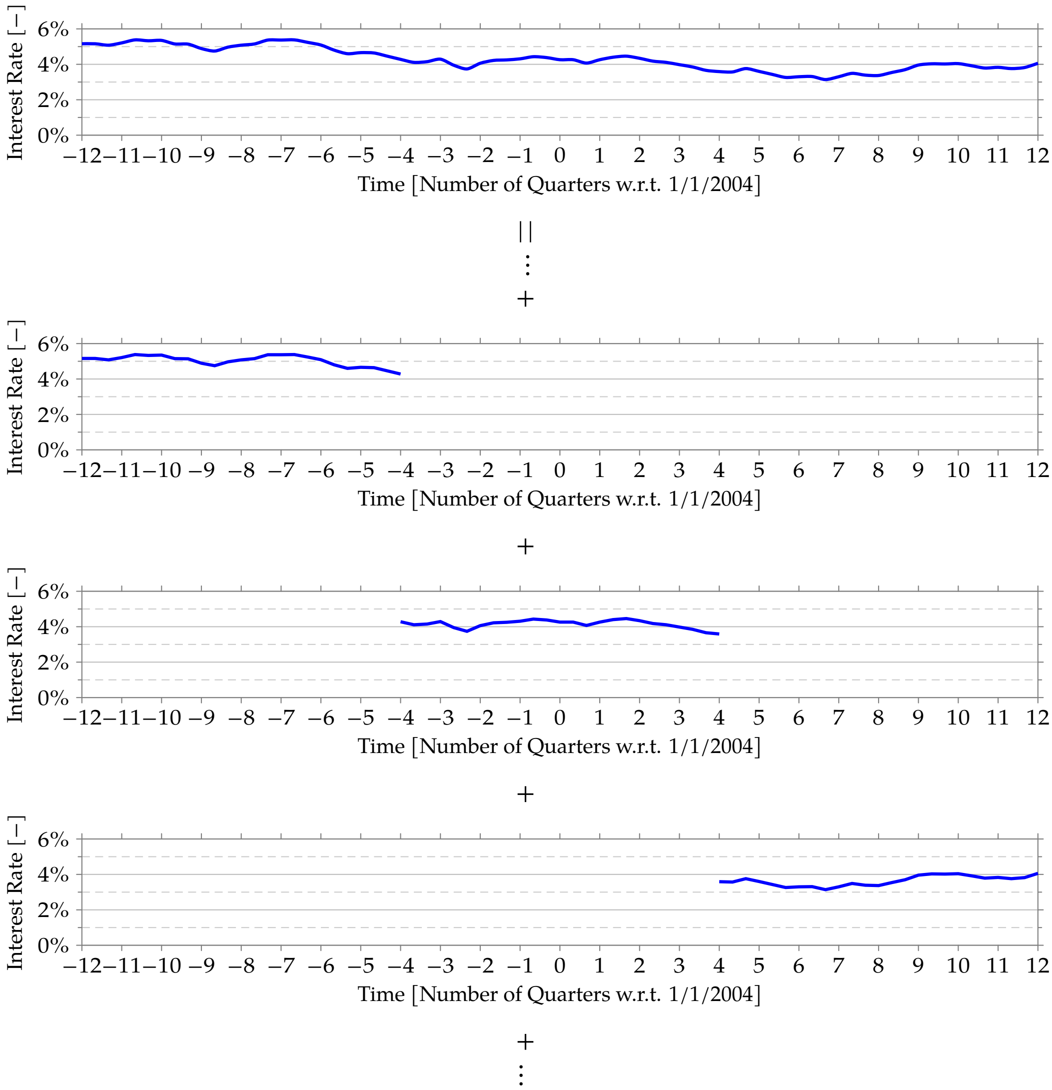


Figure 2.7: Illustration of the frame decomposition using the continuous-time signal of Figure 2.5(a), breaking it into biennial frames.

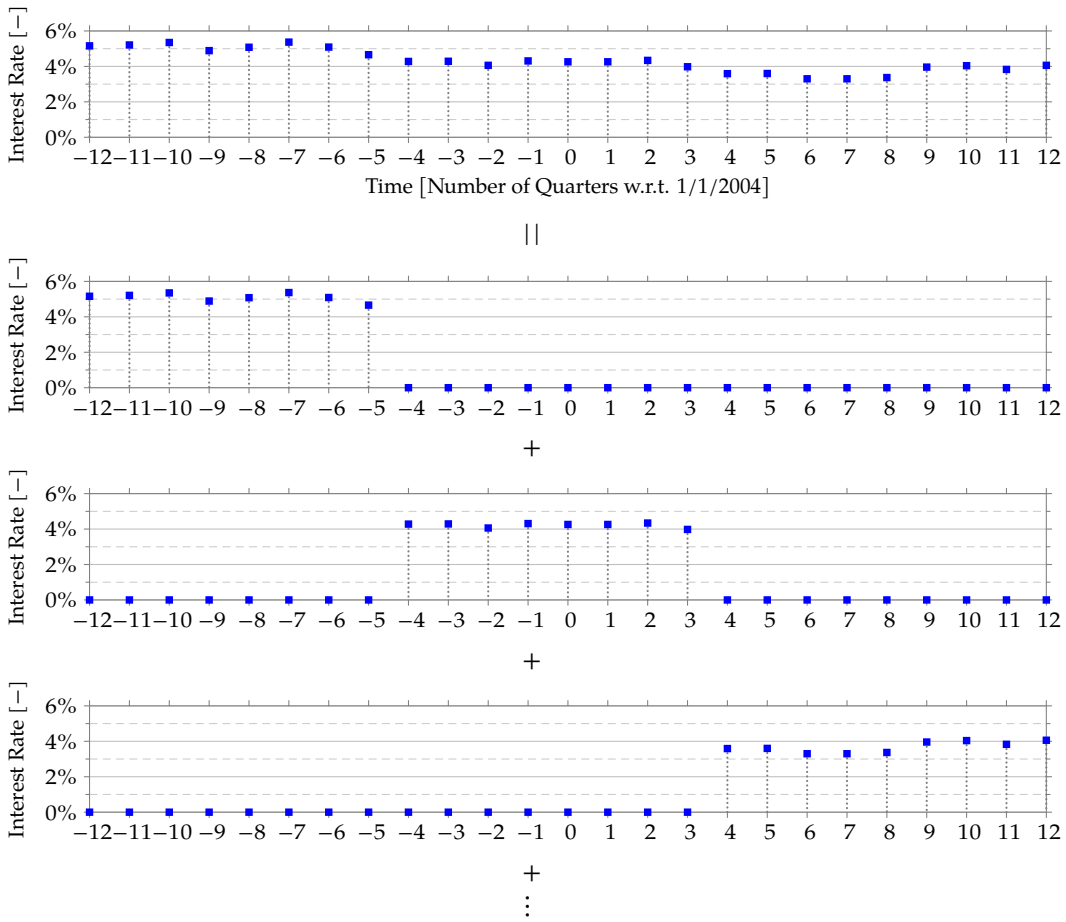


Figure 2.8: Illustration of the frame decomposition using the discrete-time signal of Figure 2.5(b), breaking it into biennial frames of 8 datapoints

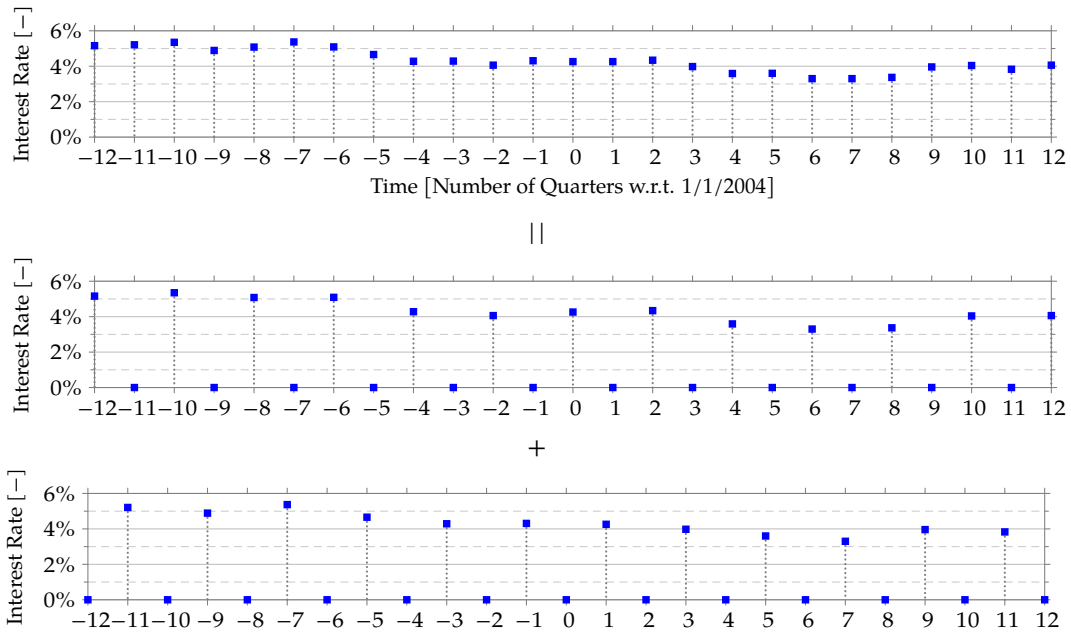


Figure 2.9: Illustration of the interlaced decomposition using the discrete-time signal of Figure 2.5(b).

with

$$f_{2n}[n] = \begin{cases} f[n] & \text{for n even} \\ 0 & \text{for n odd} \end{cases} \qquad f_{2n+1}[n] = \begin{cases} 0 & \text{for n even} \\ f[n] & \text{for n odd} \end{cases}$$

This has been illustrated in Figure 2.9.

2.8.4 Even/Odd decomposition

Continuous

Our goal is to find two signals, one of them even $(f_e(t))$, the other one odd $(f_o(t))$, that add up to the original signal $f(t)$. One can verify that the proposed solution below suits our needs.

$$f(t) = f_e(t) + f_o(t) \quad \text{with} \quad \begin{aligned} f_e(t) &= \frac{f(t) + f(-t)}{2} \\ f_o(t) &= \frac{f(t) - f(-t)}{2} \end{aligned}$$

This decomposition has been illustrated in Figure 2.10 on the next page.

Discrete

A similar reasoning leads to the discrete-time decomposition:

$$f[n] = f_e[n] + f_o[n] \quad \text{with} \quad \begin{aligned} f_e[n] &= \frac{f[n] + f[-n]}{2} \\ f_o[n] &= \frac{f[n] - f[-n]}{2} \end{aligned}$$

This decomposition has been illustrated in Figure 2.11 on the following page.

All these strange decompositions might bring you to the question: isn't there a decomposition

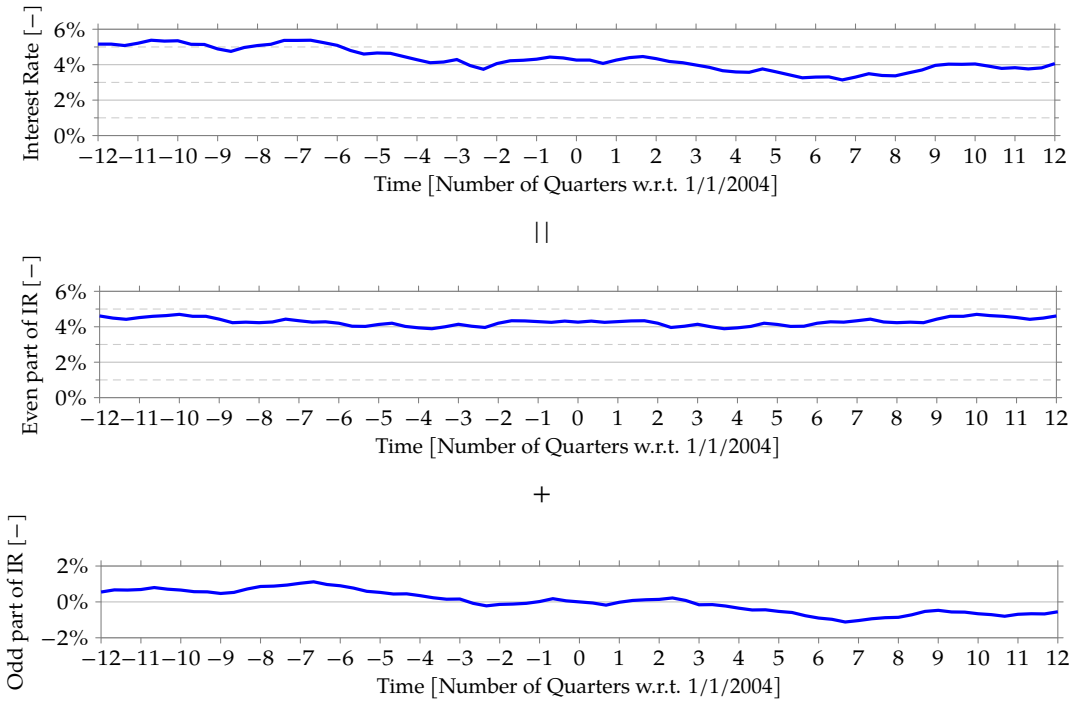


Figure 2.10: Illustration of the even/odd decomposition using the continuous-time signal of Figure 2.5(a).

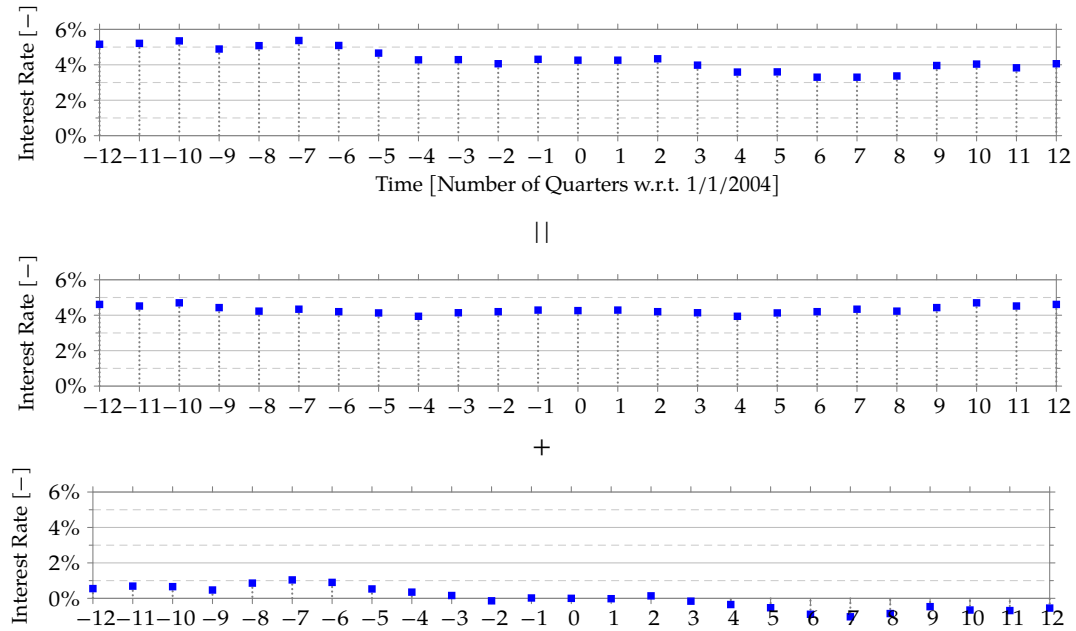


Figure 2.11: Illustration of the even/odd decomposition using the discrete-time signal of Figure 2.5(b).

that maps a signal to a set of basic signals, very much like a vector can be decomposed into its components along the vector space's orthogonal or orthonormal basis?

Yes, there is. You can read all about it in the next chapter!

Exercises

Some exercises on even/odd decomposition

Exercise 2.8.4-1: Determine the even/odd decomposition of the following signal:

$$x(t) = -0.2 \sin(20t)$$

Exercise 2.8.4-2: Determine the even/odd decomposition of the following signal:

$$x[n] = \pi \cos[5n]$$

Exercise 2.8.4-3: Determine the even/odd decomposition of the following signal:

$$x[n] = [-0.3, 0.4, 0.2, -0.3, 0.5, 0.7, \underbrace{0.25}_{n=0}, 0.1, -0.3, -0.4, -0.2, 0, -0.2]$$

Chapter 3

The Fourier transform

In this chapter, you will learn about:

- the kinds of Fourier transforms related to discrete time and frequency: the Discrete-time Fourier Transform and the Discrete Fourier Transform,
- how they are related,
- their properties,
- the implications of discretizing time and/or frequency,
- Shannon and his theorems.

After having read this chapter, some questions will still be left unanswered:

- how can we be smarter than Shannon?
- what exactly is circular convolution?

After having read/studied this chapter, you are expected to be able to

- calculate the DFT (or its inverse) when given a signal or a spectrum and use the DFT to calculate the DtFT (or its inverse),
- understand and explain the periodicity properties (i.e., the effects of time and frequency sampling),
- understand and correctly apply the sampling theorems (Shannon).

3.1 Introduction

You have studied the Fourier familily diagram before, in your course on Systems Theory. However, at that moment our discussion was limited to the continuous time and frequency domain. We will now delve into the bottom half of the family diagram (see Figure 3.1 on the next page).

The diagram connects continuous signals and their transformed versions with discrete signals and their transforms. Besides the mathematical beauty of this "evolutionary" diagram, the diagram and how transforms are related to eachother is important to understand the effects introduced by any of these transforms. In the diagram, the inner ring represents the signals in the time domain, while the outer ring represents the signals in the frequency domain (see illustration in Figure 3.2(a)).

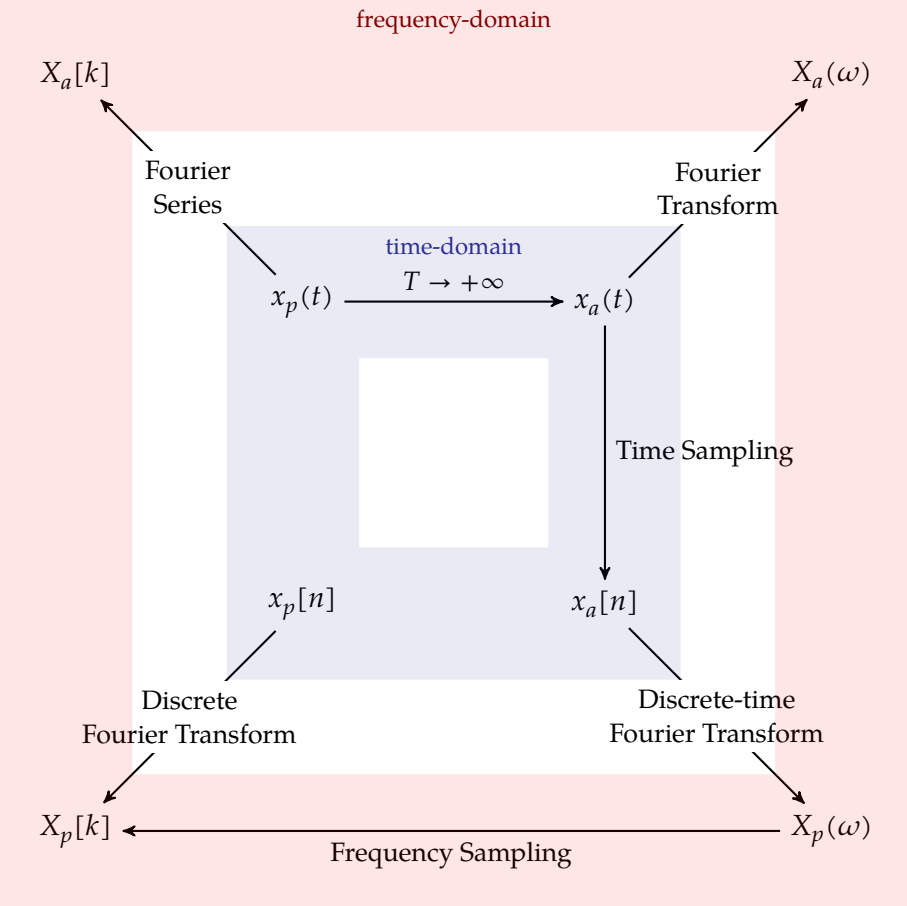


Figure 3.1: From Fourier series to Discrete Fourier Transform (DFT): the Fourier family diagram. In this course, we will focus on the bottom half.

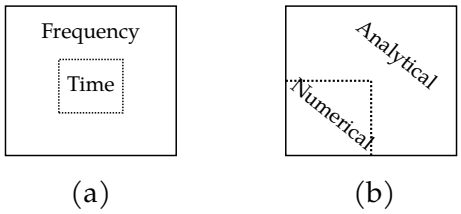


Figure 3.2: Distinct regions in the Fourier family diagram: (a) time versus frequency, (b) analytic versus numeric.

For now, the only thing to remember, is that we normally go through the transforms of the Fourier family in a specific order:

1. Fourier Series
2. Fourier Transform
3. Discrete-time Fourier Transform
4. Discrete Fourier Transform

You made the transition from Fourier Series to Fourier transform in your course on Systems Theory. We encourage the reader to review this transition.

There is one aspect that we did not yet highlight: the first three are analytic transforms, i.e. treating them with a computer (or a Digital Signal Processor) is not easy (see illustration in Figure 3.2(b)). The latter is a pure numerical method and our end goal in our quest for mathematical DSP tools. Prepare yourself for quite a journey!

Remarks Though as a DSP engineer, we will most likely encounter real-valued quantities and signals in the world that surrounds us, all the transforms treated in this chapter work without adaptation for complex-valued signals as well. We will see how to exploit this fact later.

3.2 The Fourier series and transform

For your convenience, we will repeat the definitions of the trigonometric and exponential Fourier series, as well as the Fourier transform.

Trigonometric Fourier Series

$$x(t) \sim \frac{A_0}{2} + \sum_{k=1}^{+\infty} A_k \cos(\omega_k t) + \sum_{k=1}^{+\infty} B_k \sin(\omega_k t) \qquad \text{assuming } \omega_k = k\frac{2\pi}{T}$$

and

$$A_k = \frac{\langle x(t), \cos\omega_k t\rangle}{\langle \cos\omega_k t, \cos\omega_k t\rangle} = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\cos(\omega_k t)\,\mathrm{d}t$$
$$B_k = \frac{\langle x(t), \sin\omega_k t\rangle}{\langle \sin\omega_k t, \sin\omega_k t\rangle} = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\sin(\omega_k t)\,\mathrm{d}t$$

Exponential Fourier Series

$$x(t) = \sum_{k=-\infty}^{+\infty} X_k\, \mathrm{e}^{\mathrm{j}\omega_k t} \qquad \text{assuming } \omega_k = k\frac{2\pi}{T}$$

with

$$X_k = \left\langle x,\, \mathrm{e}^{\mathrm{j}\omega t}\right\rangle = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\,\mathrm{e}^{-\mathrm{j}\omega_k t}\,\mathrm{d}t$$

Fourier Transform

$$\begin{cases} x(t) = \dfrac{1}{2\pi}\displaystyle\int_{-\infty}^{+\infty} X(\omega)\, e^{j\omega t}\, d\omega \\ X(\omega) = \displaystyle\int_{-\infty}^{+\infty} x(t)\, e^{-j\omega t}\, dt \end{cases}$$

3.3 The discrete-time Fourier transform (DtFT)

Remember from your course on Systems Theory, that we derived the Fourier transform from the Fourier series by disguising a aperiodic signal into a periodic cloak, to then plug it in the Fourier series (and lose the cloak in the process by increasing the period to infinity). We will take the same approach here. We'll try to dress up a discrete-time signal as a continuous-time signal and plug it into the Fourier transform.

3.3.1 Definition

So let's sample our continuous time signal with an interval T_s. In this way, we obtain a discrete-time signal:

$$x[n] = x(nT_s), \forall n \in \mathbb{Z} \tag{3.1}$$

To be able to start from the (continuous-time) Fourier transform, we need to model this discrete-time signal in a continuous way...

How about just zeroing our signal in between the sampling points?

$$x_{mod-bad}(t) = \begin{cases} x[n] & t = nT_s \\ 0 & t \neq nT_s \end{cases}$$

However, this is not a good solution, as any integral of a set of bounded discrete points (even when running from $-\infty$ to $+\infty$, as our Fourier integral does), yields zero. All information in our frequency spectrum is gone.

In fact, the spikes at the sample points are "not powerful enough". Let's give them a bit more juice and consider them to be Dirac impulses. We know that integrals of Dirac impulses are nonzero.
Still, we can do even better, let's try to make the area below our modeled function in between two sample points as close as can be to the same area in the original function (this corresponds to maintaining the same energy in both signals). This can be accomplished by multiplying the Dirac impulses by T_s. Hence:

$$x_{mod-ts}(t) = T_s \sum_{n=-\infty}^{+\infty} x[n]\delta(t - nT_s) \tag{3.2}$$

Forth... Now, let's apply the (continuous-time) Fourier transform to this function:

$$\begin{aligned} X_{mod-ts}(\omega) &= \int_{-\infty}^{+\infty} x_{mod-ts}(t)\, \mathrm{e}^{-\mathrm{j}\omega t}\, \mathrm{d}t \\ &= \int_{-\infty}^{+\infty} T_s \sum_{n=-\infty}^{+\infty} x[n]\delta(t-nT_s)\, \mathrm{e}^{-\mathrm{j}\omega t}\, \mathrm{d}t \\ &= T_s \sum_{n=-\infty}^{+\infty} x[n] \int_{-\infty}^{+\infty} \delta(t-nT_s)\, \mathrm{e}^{-\mathrm{j}\omega t}\, \mathrm{d}t \\ &= T_s \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}\omega n T_s} \end{aligned}$$

And then, an illogical choice was made, one might even speak about an error. Instead of taking the above equation as the (proper) definition for the forward discrete-time Fourier transform, it was decided to remove the factor T_s from it, defining as the DtFT:

$$X_p(\omega) = \frac{1}{T_s} X_{mod-ts}(\omega)$$

then this reduces to:

$$X_p(\omega) = \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}\omega n T_s}$$

And this is the (real) definition of the discrete-time Fourier transform. You may safely ignore the subscript p just for now. It will become clear later on why this subscript has been chosen.

...and back Applying the inverse transform yields:

$$x_{mod-ts}(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X_{mod-ts}(\omega)\, \mathrm{e}^{\mathrm{j}\omega t}\, \mathrm{d}\omega$$

As we know that our time-signal only has nonzero values at the sampling points, we can reduce this to:

$$\begin{aligned} x_{mod-ts}(nT_s) &= \frac{T_s}{2\pi} \int_{-\infty}^{+\infty} X_p(\omega)\, \mathrm{e}^{\mathrm{j}\omega n T_s}\, \mathrm{d}\omega \\ &\downarrow \quad \omega_s = \frac{2\pi}{T_s} \\ &= \frac{1}{\omega_s} \int_{-\infty}^{+\infty} X_p(\omega)\, \mathrm{e}^{\mathrm{j}\omega n T_s}\, \mathrm{d}\omega \end{aligned}$$

This inverse transform squeezes again the T_s-weighted impulse train out of the spectrum of the signal.

It can be shown that one can get rid of the impulses and the factor T_s in (3.2), by just integrating from $-\frac{\omega_s}{2}$ to $+\frac{\omega_s}{2}$.

$$x[n] = \frac{1}{\omega_s} \int_{-\frac{\omega_s}{2}}^{+\frac{\omega_s}{2}} X_p(\omega)\, \mathrm{e}^{\mathrm{j}\omega n T_s}\, \mathrm{d}\omega$$

In this way, we obtain the definition of the

Discrete-time Fourier Transform (DtFT)

$$X_p(\omega) = \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}\omega n T_s} \tag{3.3}$$

$$x[n] = \frac{1}{\omega_s} \int_{-\frac{\omega_s}{2}}^{+\frac{\omega_s}{2}} X_p(\omega)\, \mathrm{e}^{\mathrm{j}\omega n T_s}\, \mathrm{d}\omega$$

Remarks It is convenient to rewrite these definitions introducing a new variable $\tilde{\omega} = \frac{2\pi\omega}{\omega_s}$. Doing so leads to

$$X_p(\tilde{\omega}) = \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}n\tilde{\omega}}$$

$$x[n] = \frac{1}{\omega_s} \int_{-\pi}^{+\pi} X_p(\tilde{\omega})\, \mathrm{e}^{\mathrm{j}n\tilde{\omega}}\, \mathrm{d}\tilde{\omega}$$

Exercises

You won't find any exercises on the DtFT right here, because there's a very convenient way to calculate the DtFT starting from the DFT applying zero padding. We'll see how in due time.

3.3.2 Time sampling and reconstruction

Sampling

So how does the sampling affect the frequency spectrum? Well, let's substitute (3.1) in (3.2) and see what we obtain:

$$\begin{aligned} x_{mod-ts}(t) =& T_s \sum_{n=-\infty}^{+\infty} x(nT_s)\, \delta(t - nT_s) \\ =& x(t)\, T_s \sum_{n=-\infty}^{+\infty} \delta(t - nT_s) = x(t) \cdot \left(T_s\, \mathrm{III}_{T_s}(t)\right) \end{aligned}$$

This means that we can model the effect of sampling by multiplication with a Dirac impulse train with weight T_s and spacing T_s.
Now, there's two things to remember (and if you don't, review sections **??** on page ?? and **??** on page ??):

- time multiplication corresponds to frequency convolution (divided by 2π):

$$x(t) \cdot y(t) \xrightarrow{\mathcal{F}} \frac{1}{2\pi} X(\omega) \star Y(\omega)$$

- the Fourier transform of a impulse train is an impulse train itself:

$$T_s\, \mathrm{III}_{T_s}(t) \xrightarrow{\mathcal{F}} 2\pi\, \mathrm{III}_{\omega_s}(\omega)$$

  with $\omega_s = \frac{2\pi}{T_s}$.

Using this information leads to:

$$X_{mod-ts}(\omega) = X(\omega) \star \mathrm{III}_{\omega_s}(\omega)$$

This seems a complicated result. However, luckily, convolving with a Dirac impulse at frequency $k\omega_s$ is nothing more than shifting the center of the frequency spectrum to $k\omega_s$. Convolving with a Dirac impulse train, means repeating this procedure for all of the Dirac impulses and adding the resulting spectra.

Take a look at Figure 3.3 on the following page in which the entire procedure outlined above has been displayed graphically. The figure starts from a bandwith limited ($\omega < \omega_B$) signal $x(t) \xrightarrow{\mathcal{F}} X(\omega)$.[1]

This is quite an interesting conclusion: by sampling in time, we actually make the frequency spectrum periodic with a period

$$\omega_s = \frac{2\pi}{T_s}$$

i.e. the *sampling frequency*. This is also the reason why we added a subscript p (for 'periodic) to the spectrum in (3.3).

We denote the frequency range $[-\omega_s/2, \omega_s/2]$ as the *primary frequency period*.

Reconstruction

Recovering the original signal is obvious: we can recover the original spectrum by applying a perfect low-pass filter with a cut-off frequency equaling $\omega_s/2$:

$$R(\omega) = \begin{cases} 1 & \text{if } |\omega| \leq \omega_s/2 \\ 0 & \text{if } |\omega| > \omega_s/2 \end{cases} \tag{3.4}$$

The procedure selects the primary frequency period out of the full spectrum. It has been illustrated in Figure 3.4 on page 43. Multiplying the spectrum, with the low-pass filter of (3.4), corresponds to convolving the time-domain pulse train with:

$$r(t) = \frac{1}{T_s} \operatorname{sinc}\left(\frac{\omega_s t}{2}\right)$$

This time-domain signal is the so-called *interpolation function*, because it restores the information in-between the sampling points. The fact that the values of the sampling points are not affected, can be readily seen by observing that the sinc function becomes zero at time instances $kT_s, \forall k \in \mathbb{Z}$.

3.3.3 Shannon's Theorem

Shannon's theorem[2] is a milestone in information theory (of which DSP is a part).

Let's reconsider Figure 3.3 on the following page. What's striking about this figure is that we're still able to recognize the original spectrum (though repeated over and over) after

[1] In the figure $X(\omega)$ does not actually correspond to the Fourier transform of $x(t)$. The drawing is kept intentionally simple in order to make it more illustrative.

[2] The essence of Shannon's theorem (or the dual of it) has been discovered (and claimed) by many people (a.o. Edmund Witthaker, Harry Nyquist, Vladimir Kotelnikov, Claude Shannon). However Shannon's name has been attached to this theorem ever since.

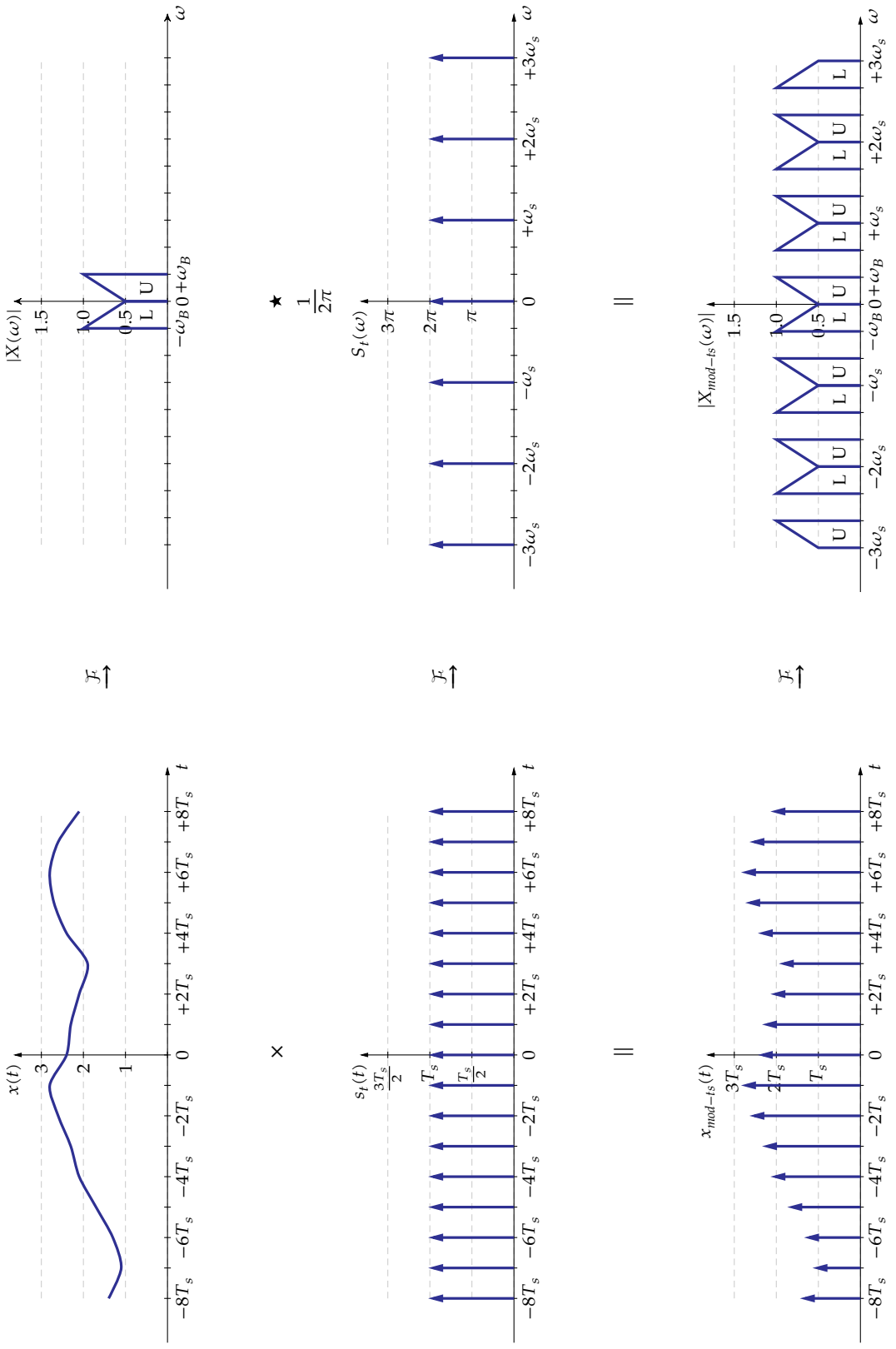


Figure 3.3: Graphical illustration of the effect of time sampling.

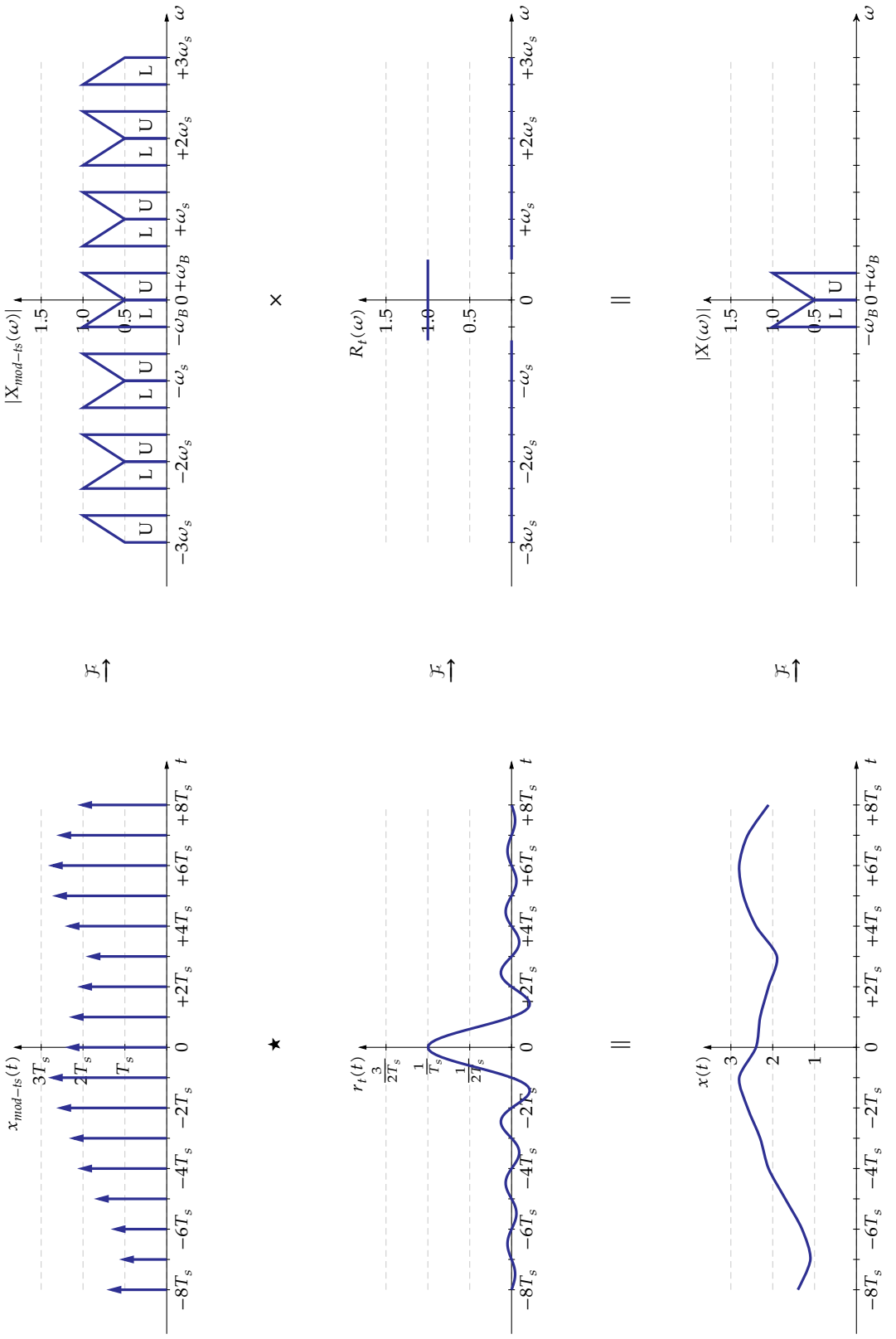


Figure 3.4: Graphical illustration of the effect of time reconstruction.

sampling the time domain. We can reconstruct the original time-domain signal by applying an appropriate low-pass filter.
Actually, we have been very lucky! Consider what would have happened if our sampling frequency ω_s would only have been half as big. This situation has been depicted in Figure 3.5.

The problem we're facing is the overlap between the high frequencies of the upper sideband and the high frequencies of the lower sideband. Our original frequency spectrum is gone.

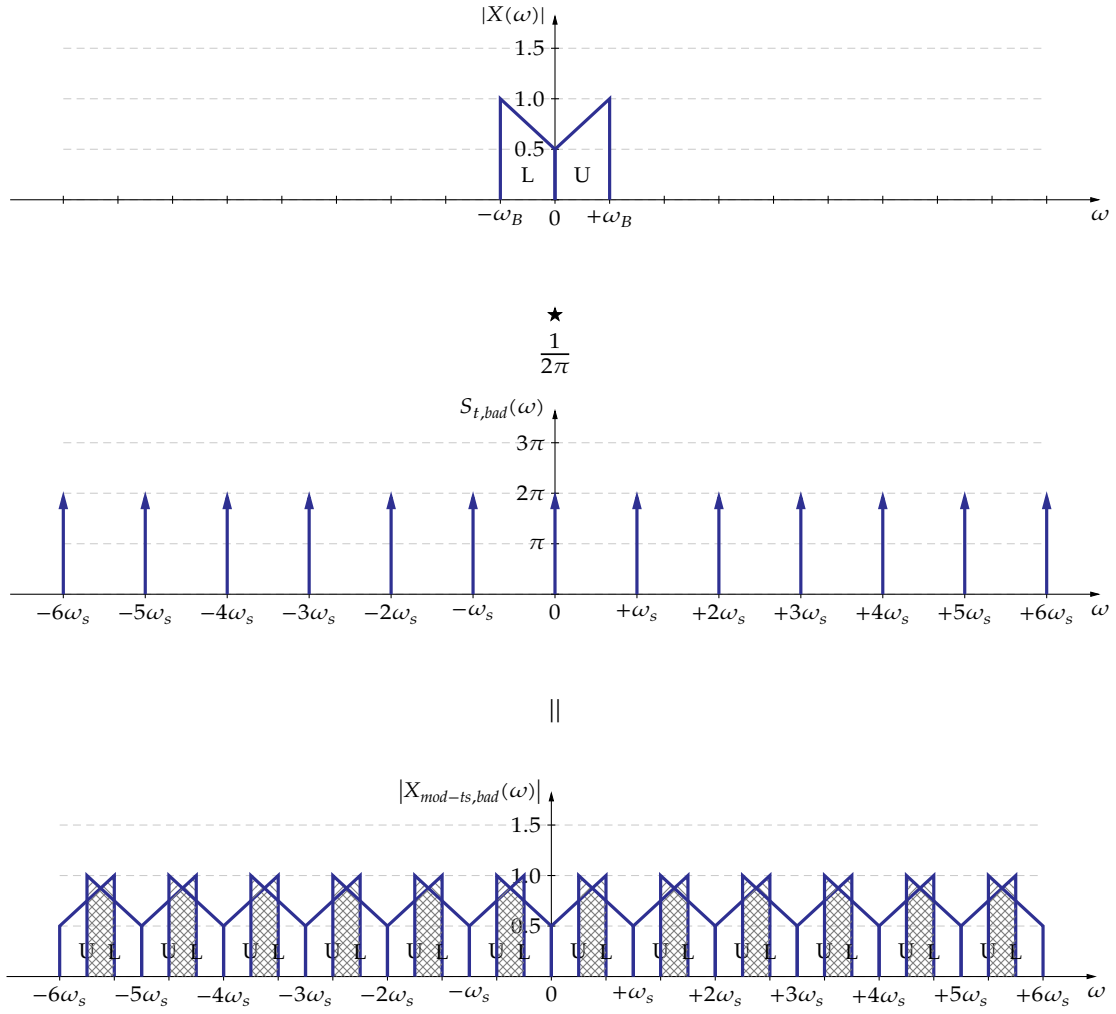


Figure 3.5: Graphical illustration of the effect of sampling with too low a frequency. The overlap of the individual frequency bands has been indicated using hatching.

This leads to the well known sampling theorem (according to Shannon):

Shannon's Time-sampling Theorem To be able to sample and reconstruct a signal with a maximum frequency content of $\pm\omega_B$ one must ensure that the sampling frequency ω_s fulfills:

$$\omega_s \geq 2\omega_B$$

This theorem puts a lower boundary to the required sampling frequency, given the bandwidth of a signal.

Remarks

- In literature you will often find this theorem formulated in terms of the *Nyquist frequency* ω_N:

$$\omega_N = \frac{\omega_s}{2}$$

  The theorem then can be written as:

$$\omega_N \geq \omega_B$$

- We formulated Shannon's theorem using angular frequencies. Of course, we can also state the very same theorem using regular frequencies:

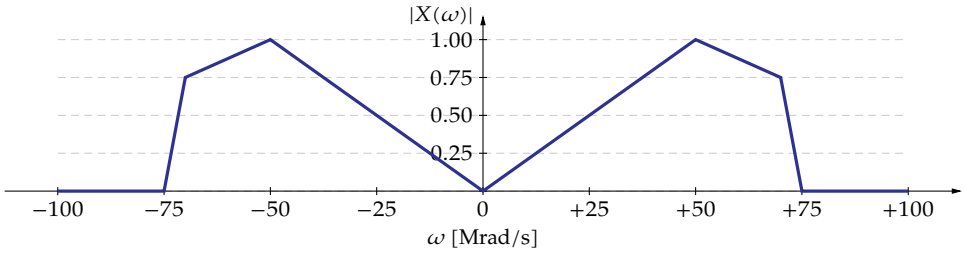
$$f_N = \frac{f_s}{2} \geq f_B$$

- We will see later that this theorem is not as rock solid as one might think. There are circumstances in which we can be smarter than Shannon (see chapter 5 on page 123).

Exercises

Some exercises on Shannon's time sampling theorem

Exercise 3.3.3-1: What is the minimal sampling frequency (in Hz) you can use to sample the signal $x(t) \xrightarrow{\mathcal{F}} X(\omega)$ such that it can be reconstructed without information loss.



Exercise 3.3.3-2: A compact disc contains music sampled at a rate of 44.1 kHz. What's the maximal frequency in the music signal that we can encode on a CD without causing information loss?

3.3.4 Aliasing

Dr. Livingstone, I presume?

The problem that Shannon's theorem tries to avoid is the polution of a signal's original spectrum by *aliases*. An alias is a frequency 'in disguise'.

The principle of aliases is very easily explained considering a simple sinus with angular frequency ω_α:

$$x_\alpha(t) = \sin(\omega_\alpha t)$$

Consider sampling the sinus with a frequency ω_s, corresponding to a sampling timestep of

$$T_s = \frac{2\pi}{\omega_s}$$

This yields a discrete-time signal:

$$\begin{aligned} x_\alpha[n] &= \sin(\omega_\alpha n T_s) \\ &= \sin\left(n 2\pi \frac{\omega_\alpha}{\omega_s}\right) \end{aligned}$$

Now, let's consider a bunch of "related" sinusoidal signals

$$x_{\beta,k}(t) = \sin\left(\omega_{\beta,k} t\right)$$

on the following frequency grid:

$$\omega_{\beta,k} = \omega_\alpha + k\omega_s, \qquad k \in \mathbb{Z}$$

Take a look at the sampled version:

$$\begin{aligned} x_{\beta,k}[n] &= \sin\left(\omega_{\beta,k} n T_s\right) \\ &= \sin\left((\omega_\alpha + k\omega_s) n \frac{2\pi}{\omega_s}\right) \\ &= \sin\left(n 2\pi \frac{\omega_\alpha + k\omega_s}{\omega_s}\right) \\ &= \sin\left(n 2\pi \frac{\omega_\alpha}{\omega_s} + n 2\pi k\right) \\ &= \sin\left(n 2\pi \frac{\omega_\alpha}{\omega_s}\right) \\ &\overset{!!}{=} x_\alpha[n] \end{aligned}$$

Conclusion: we cannot make any distinction between sinusoids that differ in frequency by a multiple of ω_s. By extension (using Euler's formula), we also cannot distinguish complex exponentials that only differ in argument by a multiple of $j\omega_s$. So, in general, we cannot distinguish frequencies that differ by a multiple of ω_s.

Examples Aliasing effects are not so uncommon as one might think. They happen where a sampling process is active, even if it is not an obvious one.

- A European television set (PAL or SECAM) presents motion pictures as a sequence of 25 frames/sec. This causes us to see a wheel (of the the car of our favorite hero) with 5 spokes spinning at 308 rpm to be seen as if it was rotating at 40rpm. Moreover, if our hero slows down such that the wheel rotates at 296 rpm, our brain would be fooled to believe that the wheel was turning backwards at 20rpm. Can you do the math that leads to these results?
- The very same effect appears in a mechanical workshop with fluorescent tube lights "flashing" at 50 Hz. Consider what we see if there's a lathe with a marker on its spindle that revolves at 3000rpm?
  This can be a life threatening situation!
  To overcome this, a typical workshop has fluorescent tube lights connected in triples to the three phases of the mains. Alternatively, they use a different lighting system, e.g. a traditional Edison light bulb.
- Most television broadcasting companies prohibit their news anchors to wear houndsthooth jackets, as these generate the famous Moiré effect. Can you imagine where the sampling process is active in this case?

Exercises
Some exercises on aliasing

Exercise 3.3.4-1: The signal $x(t)$, with

$$x(t) = \cos\left(2\pi \cdot 2 \times 10^8 \text{ Hz} \cdot t\right)$$

is sampled at 150 MHz. Draw the resulting magnitude spectrum in the Fourier domain after sampling.

Exercise 3.3.4-2: Your GSM samples your voice signal at a rate of 8 kHz. Your little nephew just got for his birthday a whistle from his dad, and while you are on the phone with your best friend he is blowing his whistle at an annoyingly high and sharp 8.25 kHz. Your best friend asks you what the low buzzing noise is. Can you explain and calculate how low (in frequency) the buzzing noise is he hears?

Note: when you try this — at home — with your cell phone, you'll notice that the effect described above is not happening in reality. Do you have any idea how your cell phone prevents it? If you don't have any clue, don't worry: Chapter 5 on sampling, quantization and reconstruction will tell you all about it.

3.4 The discrete Fourier transform (DFT)

In the previous section we transformed the Fourier transform into a discrete-time Fourier transform. This allows representing the time-signal with a discrete (yet still infinite) number of values. This is a major step into the direction of DSP. However, we're not there yet. The frequency spectrum (though periodical by consequence) is still a continuous spectrum.

3.4.1 Definition

An obvious idea is to sample the frequency spectrum using a frequency grid with pitch ω_f:

$$X_p[k] = X_p(k\omega_f), \forall k \in \mathbb{Z} \tag{3.5}$$

The quantity $1/\omega_f$ is also very often called the *frequency resolution*.

In view of the very same reasons mentioned in section 3.3.1 on page 38, we model this by multiplying our frequency spectrum with a grid of Dirac frequency impulses of the appropriate weight:

$$X_{p,mod-fs}(\omega) = \omega_f \sum_{k=-\infty}^{+\infty} X_p[k]\delta(\omega - k\omega_f) \tag{3.6}$$

You might get a déjà-vu feeling when reading the next sections. We're going through exactly the same math as in section 3.1 on page 38, but this time with time replaced by frequency and vice versa.

Because we know that all frequency information is contained in the frequency range from $-\omega_s/2$ to $\omega_s/2$, it makes sense to sample that range using a finite number of points. Let's take N points, with:

$$N = \frac{\omega_s}{\omega_f}$$

Back... Now, let's apply the discrete-time inverse Fourier transform to this function:

$$\begin{aligned}
x_{mod-fs}[n] =& \frac{1}{\omega_s} \int_{-\omega_s/2}^{+\omega_s/2} X_{p,mod-fs}(\omega)\, \mathrm{e}^{\mathrm{j}\omega n T_s}\, \mathrm{d}\omega \\
& \downarrow \quad X_{p,mod-fs}(\omega) \text{ and } \mathrm{e}^{\mathrm{j}\omega n T_s} \text{ are periodic with period } \omega_s \\
=& \frac{1}{\omega_s} \int_{0}^{+\omega_s} X_{p,mod-fs}(\omega)\, \mathrm{e}^{\mathrm{j}\omega n T_s}\, \mathrm{d}\omega \\
=& \frac{1}{\omega_s} \int_{0}^{+\omega_s} \omega_f \sum_{k=-\infty}^{+\infty} X_p[k]\delta(\omega - k\omega_f)\, \mathrm{e}^{\mathrm{j}\omega n T_s}\, \mathrm{d}\omega \\
=& \frac{\omega_f}{\omega_s} \sum_{k=0}^{N-1} X_p[k] \int_{0}^{\omega_s} \delta(\omega - k\omega_f)\, \mathrm{e}^{\mathrm{j}\omega n T_s}\, \mathrm{d}\omega \\
=& \frac{\omega_f}{\omega_s} \sum_{k=0}^{N-1} X_p[k]\, \mathrm{e}^{\mathrm{j}kn\omega_f T_s} \\
& \downarrow \quad N = \frac{\omega_s}{\omega_f} \\
=& \frac{1}{N} \sum_{k=0}^{N-1} X_p[k]\, \mathrm{e}^{\mathrm{j}\frac{2\pi kn}{N}}
\end{aligned}$$

...and forth Applying the forward discrete-time Fourier transform yields:

$$X_p(\omega) = \sum_{n=-\infty}^{+\infty} x_{mod-fs}[n]\, \mathrm{e}^{-\mathrm{j}\omega n T_s}$$

As we know that our frequency-domain signal only has nonzero values at the frequency sampling points, we can reduce this to:

$$\begin{aligned}
X_p(k\omega_f) &= \sum_{n=-\infty}^{+\infty} x_{mod-fs}[n]\, \mathrm{e}^{-\mathrm{j}k\omega_f n T_s} \\
&= \sum_{n=-\infty}^{+\infty} x_{mod-fs}[n]\, \mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{N}}
\end{aligned}$$

This inverse transform squeezes again the ω_f-weighted frequency impulse train out of the time-domain representation of the signal.

It can be shown that one can get rid of the impulses and the factor ω_f in (3.6) by just summing from $k = 0$ to $k = N - 1$ with $N = \omega_s/\omega_f$. This leads to:

$$X_p[k] = \sum_{n=0}^{N-1} x_{mod-fs}[n]\, \mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{N}}$$

In this way, we obtain the definition of the[3]

[3]Just for now, just ignore the subscript p for the time-domain signal in the equations below. The reason for labeling it in this way, will become clear later on.

Discrete Fourier Transform

$$x_p[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_p[k]\, \mathrm{e}^{\mathrm{j}\frac{2\pi kn}{N}} \tag{3.7}$$

$$X_p[k] = \sum_{n=0}^{N-1} x_p[n]\, \mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{N}} \tag{3.8}$$

Remarks

- Note that by sampling the spectrum, we lost information about frequencies in between the sampling grid. For obvious reasons, this effect is called *picket fencing*.
- When considering a signal on a fixed time window for different sampling frequencies (i.e. varying N), the DFT scales with N, which seems counter intuitive. Therefore, sometimes the factor $1/N$ is shifted from the synthesis equation to the analysis equation.
- There's more: often, the equations of the DFT are made symmetrical by distributing the factor $1/N$ over the two equations, leading to:

  $$x_p^{\bullet}[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_p^{\bullet}[k]\, \mathrm{e}^{\mathrm{j}\frac{2\pi kn}{N}}$$

  $$X_p^{\bullet}[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_p^{\bullet}[n]\, \mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{N}}$$

  In this way, the DFT and its inverse can be calculated using the very same algorithm. We only have to switch the role of n and k and reverse one of the resulting outputs.
- Applying the same frequency sampling technique to the (continuous) Fourier Transform yields the Fourier Series. You may try this on a rainy day...
- Often the following shorthand is used:

  $$W_N = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}}$$

  This allows writing equations (3.8) and (3.7) in the following compact form:

  $$x_p[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_p[k] W_N^{-kn}$$

  $$X_p[k] = \sum_{n=0}^{N-1} x_p[n] W_N^{kn}$$

Exercises

We will postpone making exercises on the DFT until Chapter 4 on page 67.

3.4.2 Frequency sampling and reconstruction

Sampling

So how does the frequency sampling affect the time-domain signal? Well, let's substitute (3.5) in (3.6) and see what we obtain:

$$\begin{aligned} X_{p,mod-fs}(\omega) &= \omega_f \sum_{k=-\infty}^{+\infty} X_p(k\omega_f)\, \delta(\omega - k\omega_f) \\ &= X_p(\omega)\, \omega_f \sum_{k=-\infty}^{+\infty} \delta(\omega - k\omega_f) = X_p(\omega) \cdot \left(\omega_f \amalg_{\omega_f}(\omega)\right) \end{aligned}$$

This means that we can model the effect of sampling by multiplication with a Dirac impulse train with weight ω_f.
Now, there's two things to remember (and if you don't, review sections **??** on page ?? and **??** on page ??):

- frequency multiplication corresponds to time convolution:

$$x(t) \star y(t) \xrightarrow{\mathcal{F}} X(\omega) \cdot Y(\omega)$$

- the inverse Fourier transform of a impulse train is an impulse train itself:

$$\text{III}_{T_f}(t) \xrightarrow{\mathcal{F}} \omega_f \text{III}_{\omega_f}(\omega)$$

Using this information leads to:

$$\begin{aligned} x_{mod-ts-fs}(t) &= \mathcal{F}^{-1}\left\{X_p(\omega)\right\} \star \mathcal{F}^{-1}\left\{\omega_f \text{III}_{\omega_f}(\omega)\right\} \\ &= \underbrace{x(t) \cdot \left(T_s \, \text{III}_{T_s}(t)\right)}_{x_{mod-ts}} \star \left(\text{III}_{T_f}(t)\right) \end{aligned}$$

with $T_f = NT_s$.

This seems a complicated result. However, luckily, convolving with a Dirac impulse at a timepoint iT_f is nothing more than shifting the center of the time-domain signal to iT_f. Convolving with a Dirac impulse train, means repeating this procedure for all of the Dirac impulses and adding the resulting time-domain signals.

Take a look at Figure 3.6 on the facing page in which the entire procedure outlined above has been displayed graphically. The figure starts from a time-limited ($0 \leq n < N_L$) signal $x[n] \xrightarrow{\mathcal{F}} X_p(\omega)$ for which the frequency is sampled with $\omega_f = \omega_s/12$.[4]

This is quite an interesting conclusion: by sampling the frequency, we actually make the time-domain signal periodic with a period

$$T_f = \frac{2\pi}{\omega_f} = \frac{2\pi N}{\omega_s} = NT_s$$

i.e. the *repetition period*. This is also the reason why we added a subscript p (for 'periodic) to the time-domain signal in (3.7).

Reconstruction

Recovering the original spectrum is obvious: we can recover the original (nonperiodic) time-domain signal by applying a rectangular windowing function with length T:

$$r(t) = \begin{cases} 1 & \text{if } |t| \leq \frac{T_f}{2} \\ 0 & \text{if } |t| > \frac{T_f}{2} \end{cases} \tag{3.9}$$

[4] In the figure $X_p(\omega)$ does not actually correspond to the Fourier transform of $x[n]$. The drawing is kept intentionally simple in order to make it more illustrative.

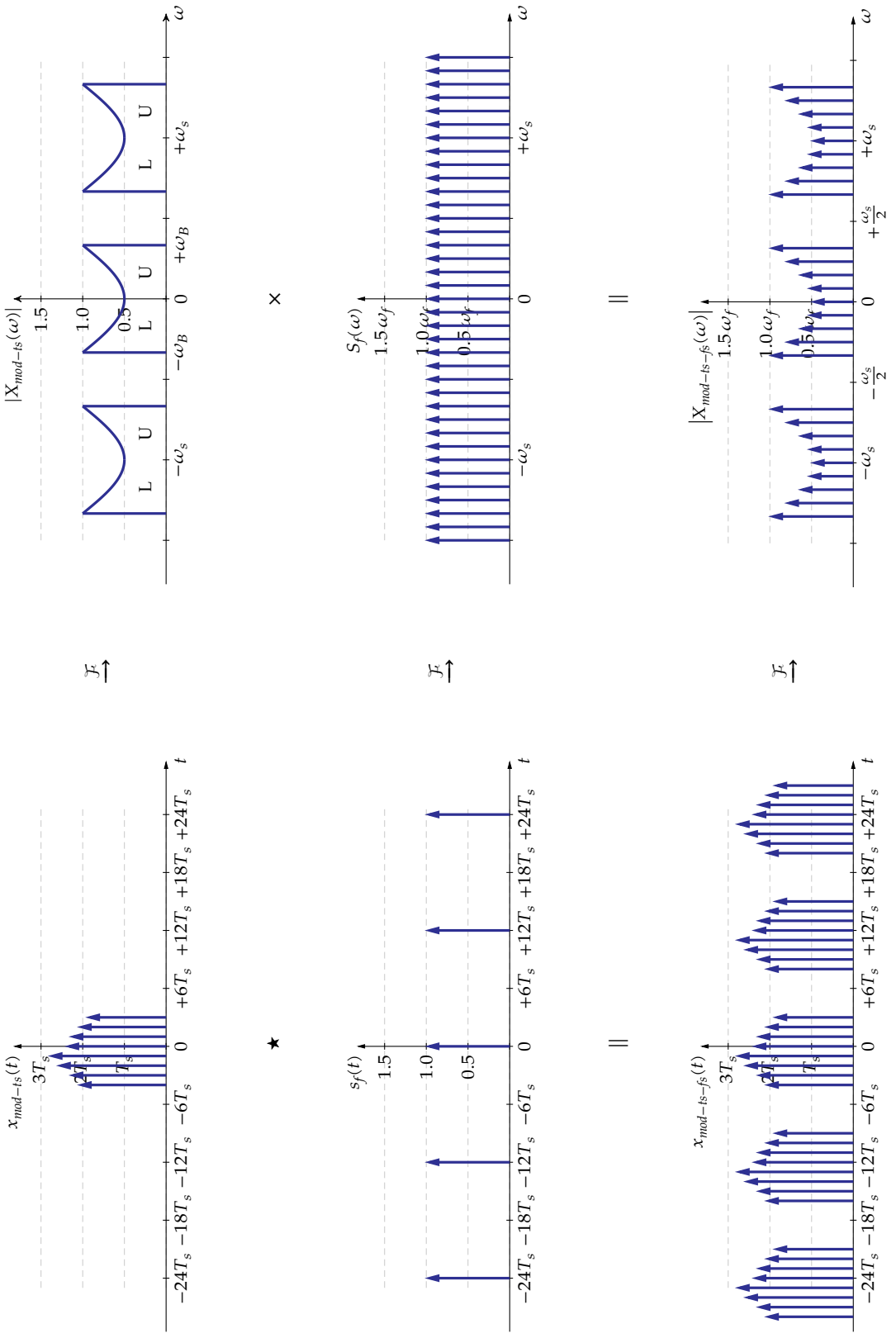


Figure 3.6: Graphical illustration of the effect of frequency sampling.

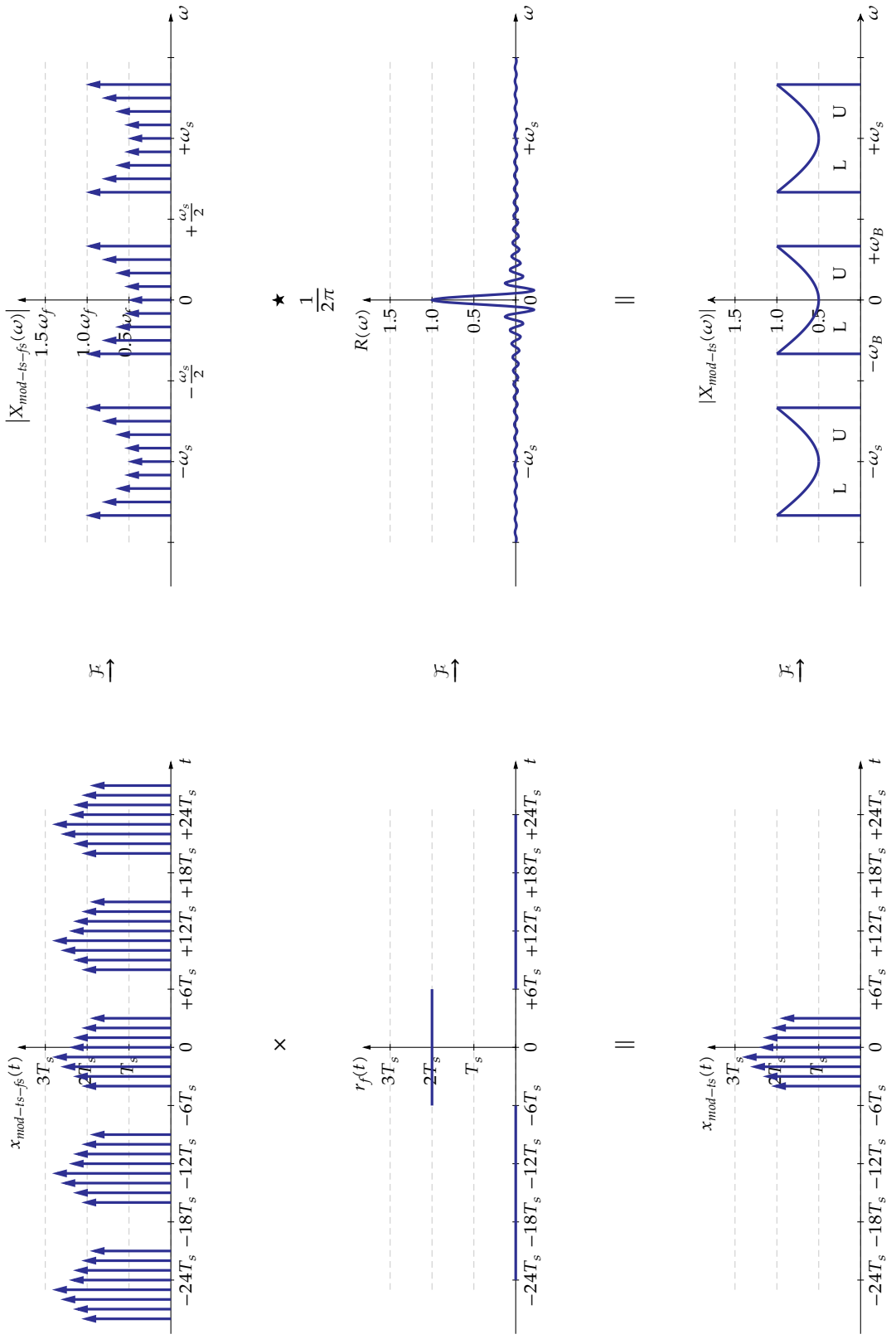


Figure 3.7: Graphical illustration of the effect of frequency reconstruction.

The procedure has been illustrated in Figure 3.7 on the preceding page. Multiplying the time-domain with the rectangular window of (3.9), corresponds to convolving the frequency spectrum train with:

$$R(\omega) = T_f \text{sinc}\left(\frac{\omega T_f}{2}\right)$$

This spectrum is the so-called *frequency interpolation function*. because it restores the information in-between the sampling points. The fact that the values of the sampling points are not affected, can be readily seen by observing that the sinc function becomes zero at time instances $k\omega_f, \forall k \in \mathbb{Z}$.

3.4.3 Shannon's Theorem revisited

Actually, the theorem below is — as far as I know — not explicitly claimed by anyone. So, let's just call it Shannon's frequency sampling theorem.

Let's reconsider Figure 3.6 on page 51. What's striking about this figure is that we're still able to recognize the original time-domain signal (though repeated over and over) after sampling the frequency domain. We can reconstruct the original spectrum by applying an appropriate window function.
Actually, we have been very lucky! Consider what would have happened if our repetition period T would only have been half as big (i.e. we would have taken a frequency pitch ω_f that was twice as big). This situation has been depicted in Figure 3.8 on the next page.

The problem we're facing is the overlap between "early" timepoints and "late" timepoints. Our original time-domain signal is ruined.

This leads to the — not so well known — frequency sampling theorem:

Shannon's Frequency Sampling Theorem To be able to sample and reconstruct a discrete-time signal of length N_L one must ensure that:

$$N = \frac{\omega_s}{\omega_f} \geq N_L$$

or

$$\omega_f \leq \frac{\omega_s}{N_L}$$

This theorem puts a lower boundary to the required frequency resolution given the length of a signal.

Remarks

- The optimal situation occurs when

  $$N_L = N$$

  i.e. one can transform a signal of length N_L (using the DFT) in a spectrum of as little as N components. In this case, one often denotes N as the *length* of the DFT.

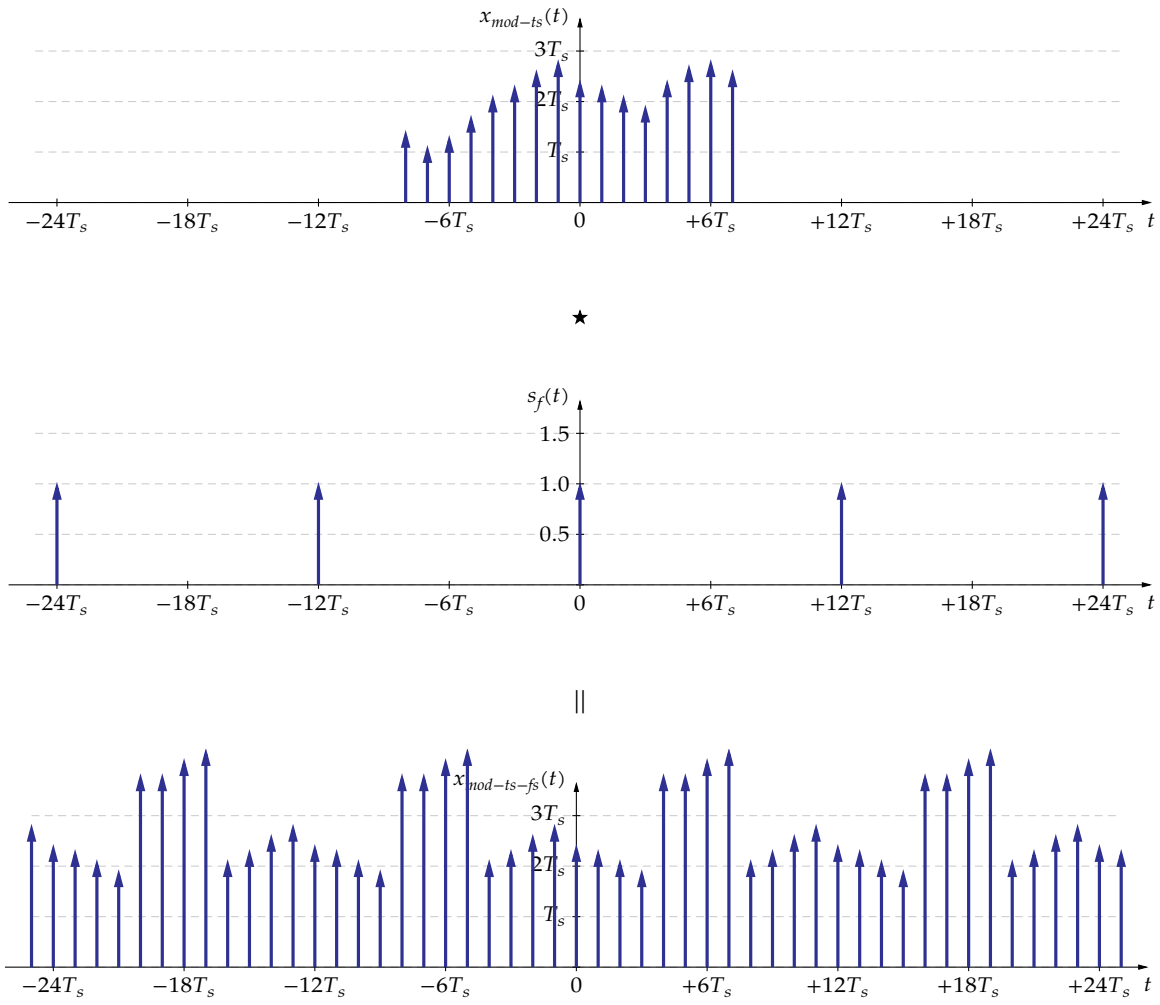


Figure 3.8: Graphical illustration of the effect of frequencysampling with too low a frequency resolution.

- This theorem also tells us, that if we want to generate the discrete spectrum of a time-limited signal, we can easily increase the frequency resolution by adding zeros at the end of the signal. This technique is called *zero padding*.

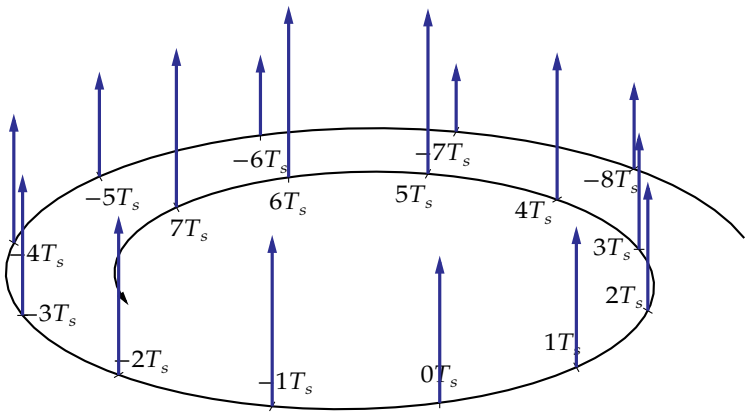
Exercises
Some exercises on Shannon's frequency sampling theorem

Exercise 3.4.3-1: Assume a signal that has been sampled at $f_s = 10\,\text{kHz}$, with a time-span of 2 seconds. What's the frequency pitch ω_f we need to take for the DFT to ensure a correct time-reconstruction.

Exercise 3.4.3-2: Given a system with a sample rate of 1 MHz. If we use a frequency pitch of 10 kHz, then what is the maximal signal time length we can treat in a single DFT run? How many sample points do we need to cover this time span?

3.4.4 Destructive circular convolution

It is I, Leclerc!
The problem that Shannon's frequency sampling theorem tries to avoid is destructive circular convolution. One also could call this, time aliasing. A time-alias is a timepoint 'in disguise'. This means mixing up early and late time-samples. The 'circular' in this term refers to the circular folding of the time axis as has been illustrated below for $N_L = 16$ and $N = 12$. The term convolution will become clear later on.



Examples We could illustrate this phenomenon using some examples. However, we will not do this, as destructive circular convolution is more easily treated once we deal with DSP filtering techniques. Besides, circular convolution effects are not so common in our daily life.

3.4.5 Computational complexity of the DFT

As this is our first fully numerical technique, we can also think about implementing it. Implementation is straightforward, but what will it cost us in terms of calculation time (and energy?).

Reconsider the equations for the DFT:

$$x_p[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_p[k] \, e^{j\frac{2\pi kn}{N}}$$
$$X_p[k] = \sum_{n=0}^{N-1} x_p[n] \, e^{-j\frac{2\pi kn}{N}}$$

Let's assume an optimal DFT, i.e. $N_L = N$. In addition, given a specific DFT length we can precompute the exponential factors of the DFT and its inverse. For the forward transform, the only live calculations that need take place are the multiplications of $x_p[n]$ with the exponential factors and the summation. For the inverse transform, similarly only the multiplications of $X_p[k]$ with the exponential factors and the summation needs being counted.

Therefore, the total time required by a DFT and the inverse DFT equals

$$T_{DFT} = N^2(T_{MULT} + T_{ADD})$$
$$T_{IDFT} = N^2(T_{MULT} + T_{ADD})$$

3.4.6 Properties of the DFT

Almost all of the Fourier Transform properties also hold for the other members of the family. Though we did not treat the properties of the Fourier series and the Discrete-time Fourier Transform, we will do so for the DFT, as we will encounter many of them frequently. Check section **??** on page ?? for comments on these properties.

Let's start by making the following assumptions:

$$x_p[n] \xrightarrow{\text{DFT}} X_p[k]$$
$$y_p[n] \xrightarrow{\text{DFT}} Y_p[k]$$
$$a, b \in \mathbb{R}$$
$$u \in \mathbb{R}_0$$

Linearity

$$ax_p[n] + by_p[n] \xrightarrow{\text{DFT}} aX_p[k] + bY_p[k]$$

Time-frequency symmetry

$$X_p[n] \xrightarrow{\text{DFT}} Nx_p[-k]$$

Cyclic Time/Frequency shifting

$$x_p[n-i] \xrightarrow{\text{DFT}} X_p[k] \, e^{-j\frac{2\pi ki}{N}}$$
$$x_p[n] \, e^{+j\frac{2\pi in}{N}} \xrightarrow{\text{DFT}} X_p[k-i]$$

Parseval's theorem

$$\sum_{n=0}^{N-1} x_p[n] y_p^*[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_p[k] Y_p^*[k]$$

If we substitute $y_p[k]$ by $x_p[k]$, Parseval tells us that:

$$\sum_{n=0}^{N-1} \left|x_p[n]\right|^2 = \frac{1}{N} \sum_{k=0}^{N-1} \left|X_p[k]\right|^2$$

Convolution/Multiplication

$$x[n] \star y[n] \xrightarrow{\text{DFT}} X[k]Y[k]$$
$$x[n] \times y[n] \xrightarrow{\text{DFT}} \frac{1}{N} X[k] \star Y[k]$$

3.5 Density functions

To treat the topic of density functions, we start again in the analog domain, considering energy and power in the continuous-time domain first, to continue in the discrete-time domain and end with seeing how we can use the discrete-time domain to make an estimate of the continuous-tiem domain.

3.5.1 Spectral density functions

We start in the continuous-time domain and consider two spectral density functions:

- the energy spectral density (ESD) function, and
- the power spectral density (PSD) function.

Both can be calculated in two ways: one suited for deterministic (known) signals and another suited for stochastic (unknown) signals.

Energy spectral density

Straight to the target

An interesting characteristic of a signal is its energy. For a signal $x(t)$ we have defined its energy E_x as

$$E_x = \int_{-\infty}^{+\infty} |x(t)|^2 \, \mathrm{d}t \tag{3.12}$$

An obvious question is: how is this energy distributed over the signal's frequency components? Parseval's theorem (see page **??**) provides the answer:

$$E_x = \frac{1}{2\pi} \int_{-\infty}^{+\infty} |X(\omega)|^2 \, \mathrm{d}\omega = \int_{-\infty}^{+\infty} \frac{|X(\omega)|^2}{2\pi} \, \mathrm{d}\omega$$

This means that the square of the magnitude of the spectrum of the signal divided by 2π can be considered to be the *energy spectral density* (ESD) function of the signal:

$$\text{ESD}(\omega) = \frac{|X(\omega)|^2}{2\pi}$$

To obtain the signal energy in a particular frequency band $[\omega_l, \omega_u]$, one has to integrate this ESD function over this angular frequency band.

We might also choose to work with ordinary frequencies f (in Hz) instead of with angular frequencies (in rad/s). In that case, the equations for the ESD boils down to:

$$\text{ESD}(f) = |X(\omega)|^2$$

In the angular case, the unit of the ESD is signal units squared times seconds squared per radian. In the ordinary case, the unit of the ESD is signal units squared times seconds squared or equivalently signal units squared times second per Hertz. E.g. if the signal has the unit Volts:

$$[\text{ESD}(\omega)] = \text{V}^2\text{s}/(\text{rad/s}) \qquad [\text{ESD}(f)] = \text{V}^2\,\text{s/Hz}$$

Taking the scenic route

A different route to the same result involves the autocorrelation function. Given a signal $x(t)$, the autocorrelation of the signal is defined as:

$$r_x(t) = x(t) \star x^*(t) = \int_{-\infty}^{+\infty} x(\tau)x^*(t+\tau)\,\mathrm{d}\tau$$

It is obvious that the energy of the signal E_x (see (3.12)) is a special case of this autocorrelation function. Indeed:

$$E_x = r_x(0)$$

Let's elaborate on this autocorrelation function:

$$\begin{aligned} r_x(t) &= x(t) \star x^*(t) \\ &\downarrow \text{(see exercise 3.5.1-3)} \\ &= x(t) \star x^*(-t) \end{aligned}$$

Then let's calculate its Fourier transform:

$$\begin{aligned} R_x(\omega) = \mathcal{F}(r_x(t)) &= \mathcal{F}(x(t) \star x^*(-t)) \\ &\downarrow \text{time convolution property} \\ &= \mathcal{F}(x(t)) \cdot \mathcal{F}(x^*(-t)) \\ &\downarrow x(t) \xrightarrow{\mathcal{F}} X(\omega) \Rightarrow x^*(-t) \xrightarrow{\mathcal{F}} X^*(\omega) \text{ (see exercise 3.5.1-4)} \\ &= X(\omega) \cdot X^*(\omega) = |X(\omega)|^2 \end{aligned}$$

Therefore, we can conclude that the ESD can also be calculated as:

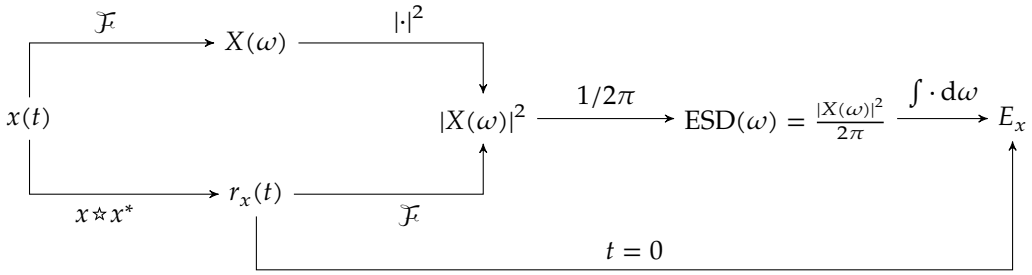
$$\text{ESD}(\omega) = \frac{R_x(\omega)}{2\pi}$$

This way of calculating the ESD is most suited for signals for which we cannot calculate a Fourier transform, e.g. noise signals. It can be shown that the derivation above can be generalized for *stationary random processes*.[5]

[5]The theory of random processes is not a subject of this course. It is a generalization of the concept of a random

Summary

The schematic drawing below summarizes the different options to calculate a signal's energy:



Power spectral density

Another interesting characteristic is a signal's average power, i.e. the energy averaged over the observation period. If the signal is time unlimited, this leads to the following definition of its average power P_x:

$$P_x = \lim_{T \to +\infty} \int_{-T/2}^{T/2} \frac{|x(t)|^2}{T} \, \mathrm{d}t$$

A similar reasoning as for the energy spectral density, leads to multiple ways to calculate the *power spectral density* (PSD) function, identical to the ESD except for the extra factor $1/T$ and the limit transition.

$$\mathrm{PSD}(\omega) = \lim_{T \to +\infty} \frac{\mathrm{ESD}(\omega)}{T}$$

Concerning units, the angular PSD has as unit signal units per radian per second; the ordinary PSD has as unit signal units per Hertz. As an example, if the signal has the unit Volts:

$$[\mathrm{PSD}(\omega)] = \mathrm{V}^2/(\mathrm{rad/s}) \qquad [\mathrm{PSD}(f)] = \mathrm{V}^2/\mathrm{Hz}$$

Special case: time-limited signals

In case the signal $x(t)$ is time limited the definitions simplify themselves, limiting the time integrals to the considered support interval and avoiding the limit transition.

Exercises

Exercise 3.5.1-1: Prove the following equality for a real signal $x(t)$:

$$x(t) \star\!\!\star x(t) = x(t) \star x(-t)$$

Exercise 3.5.1-2: Prove the following equality for a real signal $x(t)$:

$$x(-t) \overset{\mathcal{F}}{\longrightarrow} X^*(\omega)$$

Exercise 3.5.1-3: (*) Prove the following equality for a complex signal $x(t)$:

$$x(t) \star\!\!\star x^*(t) = x(t) \star x^*(-t)$$

Hint: try to solve Exercise 3.5.1-1 first.

variable in which time dependence is added. Noise can be seen as a random process. Many common noise sources are stationary.

Exercise 3.5.1-4: (*) Prove the following equality for a complex signal $x(t)$:

$$x^*(-t) \xrightarrow{\mathcal{F}} X^*(\omega)$$

Hint: try to solve Exercise 3.5.1-2 first.

Exercise 3.5.1-5: (**) A discrete-time signal $x[n]$ is often a sampled version of an analog signal $x(t)$ sampled at $t = n\,T_s$.
Try to relate the power spectral mass of $x[n]$ to the power spectral density of $x(t)$.

3.5.2 Spectral mass functions

Let's review the story of the two spectral densities, but now in the discrete time and frequency domain. We can discern:

- the energy spectral mass (ESM) function, and
- the power spectral mass (PSM) function.

As we no longer have a continuous frequency axis, the energy (or power) is concentrateed in specific spectral lines. Therefore, we can no long speak about densities, but use the term *mass*.

Given the discrete nature of time and frequency, we limit ourselves to time and bandwidth limited signals, and assume both time and frequency domain are periodic.[6] Again, both can be calculated in two ways: one suited for deterministic (known) signals and another suited for stochastic (unknown) signals.

Energy spectral mass

Straight to the target
An interesting characteristic of a signal is its energy. For a time-limited signal $x[n]$ that is defined for $n = 0, 1, \ldots, N-1$, we have defined its energy E_x as

$$E_x = \sum_{n=0}^{N-1} |x[n]|^2$$

An obvious question is: how is this energy distributed over the signal's frequency components? Parseval's theorem (see page 57) provides the answer:

$$E_x = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 = \sum_{k=0}^{N-1} \frac{|X[k]|^2}{N}$$

This means that the square of the magnitude of the spectrum of the signal divided by N can be considered to be the *energy spectral mass* (ESM) function of the signal:

$$\mathrm{ESM}[k] = \frac{|X[k]|^2}{N}$$

[6] As an exercise you might want to make derivation below for the case of the DtFT, i.e. discrete (non-periodic) time signals and continuous but periodic frequency spectra.

To obtain the signal energy in a particular frequency interval $[k_l, k_u]$, one has to sum this ESM over this frequency interval. The unit of the ESM is signal units squared. E.g. if the signal has the unit Volts, the ESM has unit V^2. To stress that the values of the function hold for a single frequency bin, sometimes the unit V^2 *per bin* is used.

Taking the scenic route

A different route to the same result involves the autocorrelation function. Given a time-limited signal $x[n]$ that is defined for $n = 0, 1, \ldots, N-1$, the autocorrelation of the signal is defined as:

$$r_x[n] = x[n] \star x^*[n] = \sum_{i=0}^{N-1} x[i]x^*[n+i]$$

assuming $x[n + kN] = x[n], \forall k \in \mathbb{Z}$.

It is obvious that the energy of the signal E_x (see (3.12)) is a special case of this autocorrelation function. Indeed:

$$E_x = r_x[0]$$

Let's elaborate on this autocorrelation function:

$$\begin{aligned} r_x[n] &= x[n] \star x^*[n] \\ &\downarrow \text{ (see exercise 3.5.2-3)} \\ &= x[n] \star x^*[-n] \end{aligned}$$

Then let's calculate its discrete Fourier transform:

$$\begin{aligned} R_x[k] = \text{DFT}\left(r_x(n)\right) &= \text{DFT}\left(x[n] \star x^*[-n]\right) \\ &\downarrow \text{ time convolution property} \\ &= \text{DFT}\left(x[n]\right) \cdot \text{DFT}\left(x^*[-n]\right) \\ &\downarrow \; x[n] \xrightarrow{\text{DFT}} X[k] \Rightarrow x^*[-n] \xrightarrow{\text{DFT}} X^*[k] \text{ (see exercise 3.5.2-4)} \\ &= X[k] \cdot X^*[k] = |X[k]|^2 \end{aligned}$$

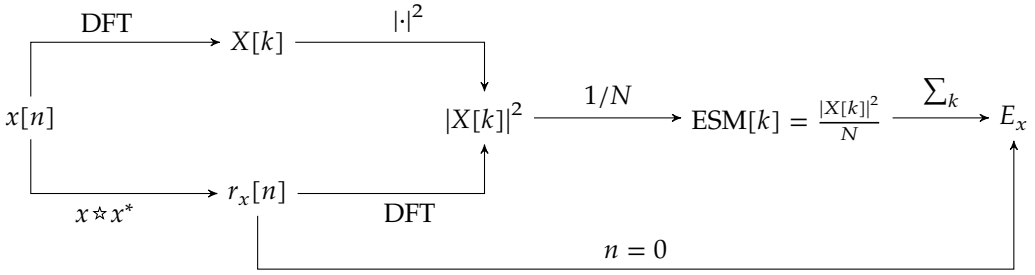
Therefore, we can conclude that the ESM can also be calculated as:

$$\text{ESM}[k] = \frac{R_x[k]}{N}$$

This way of calculating the ESM is most suited for signals for which we cannot calculate a Fourier transform, e.g. noise signals. It can be shown that the derivation above can be generalized for *stationary random processes*.[7]

Summary

The schematic drawing below summarizes the different options to calculate a signal's energy:



[7]The theory of random processes is not a subject of this course. It is a generalization of the concept of a random variable in which time dependence is added. Noise can be seen as a random process. Many common noise sources are stationary.

Power spectral mass

Another interesting characteristic is a signal's average power, i.e. the energy averaged over the observation period. This leads to the following definition of its average power P_x:

$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} |x[n]|^2$$

A similar reasoning as for the energy spectral mass function, leads to multiple ways to calculate the *power spectral mass* (PSM) function, identical to the ESM except for the extra factor $1/N$.

$$\text{PSM}[k] = \frac{\text{ESM}[k]}{N}$$

Exercises

Exercise 3.5.2-1: Prove the following equality for a real signal $x[n]$:

$$x[n] \star\!\!\star x[n] = x[n] \star x[-n]$$

Note that we are dealing with circular convolution and circular correlation and assume $x[n]$ to be periodic with period N.

Exercise 3.5.2-2: Prove the following equality for a real signal $x[n]$:

$$x[-n] \xrightarrow{\text{DFT}} X^*[k]$$

Exercise 3.5.2-3: (*) Prove the following equality for a complex signal $x[n]$:

$$x[n] \star\!\!\star x^*[n] = x[n] \star x^*[-n]$$

Hint: try to solve Exercise 3.5.2-1 first.

Exercise 3.5.2-4: (*) Prove the following equality for a complex signal $x[n]$:

$$x^*[-n] \xrightarrow{\mathcal{F}} X^*[k]$$

Hint: try to solve Exercise 3.5.2-2 first.

3.6 Epilogue

Now, take your time to review the Fourier family diagram of Figure 3.1 on page 36 again. By now it should be clear to you that all the transforms of the Fourier family are very related. You should also be able to appreciate the subscripts p (for periodic) and a (for aperiodic).

When discussing the diagram the first time, we were able to recognize several regions in the diagram: time vs. frequency and analytic vs. numerical. At this moment, we can distinguish some more regions (see Figure 3.9 on the facing page). You should be able to explain them all.

The journey we took through the diagram can be made in any direction. In fact, we covered the three basic techniques you need to traverse the diagram in any direction you would like:

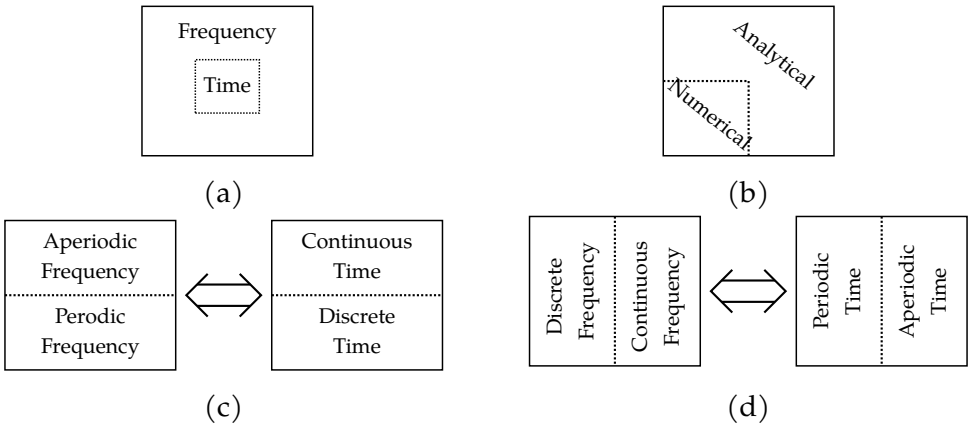


Figure 3.9: Distinct regions in the Fourier family diagram: (a) time vs. frequency, (b) analytic vs. numeric, (c) continuous vs. discrete time, (d) continuous vs. discrete frequency.

- considering limit cases,
- time sampling/interpolation, and
- frequency sampling/interpolation.

In those derivations, the handling of extra multiplication factors 2π, N, T_s and ω_f made the derivations — though correct — sometimes a little bit tricky. The key was, that we tried to model (and make drawings of) the signals appearing in any of the Fourier family's transforms in continuous time and frequency. This kind of unique reference domain, i.e. continuous time and frequency, is necessary to allow comparision of the effects of all the transforms.

The transitions with the multiplication factors involved have been gathered in Figure 3.10 on the next page, taking into account the proper definition of the DtFT, and in Figure 3.11 on page 65, taking into account the real definition of the DtFT.

Finally, a mathematical remark: the Fourier transform and its derivatives are *orthogonal signal decompositions*. In a finite-dimensional vector space a vector can be decomposed into basis vectors spanning the vector space. In the very same way a signal can be decomposed in the infinite-dimensional function space. The exponentials (or sines or cosines) are basis functions for this function space.

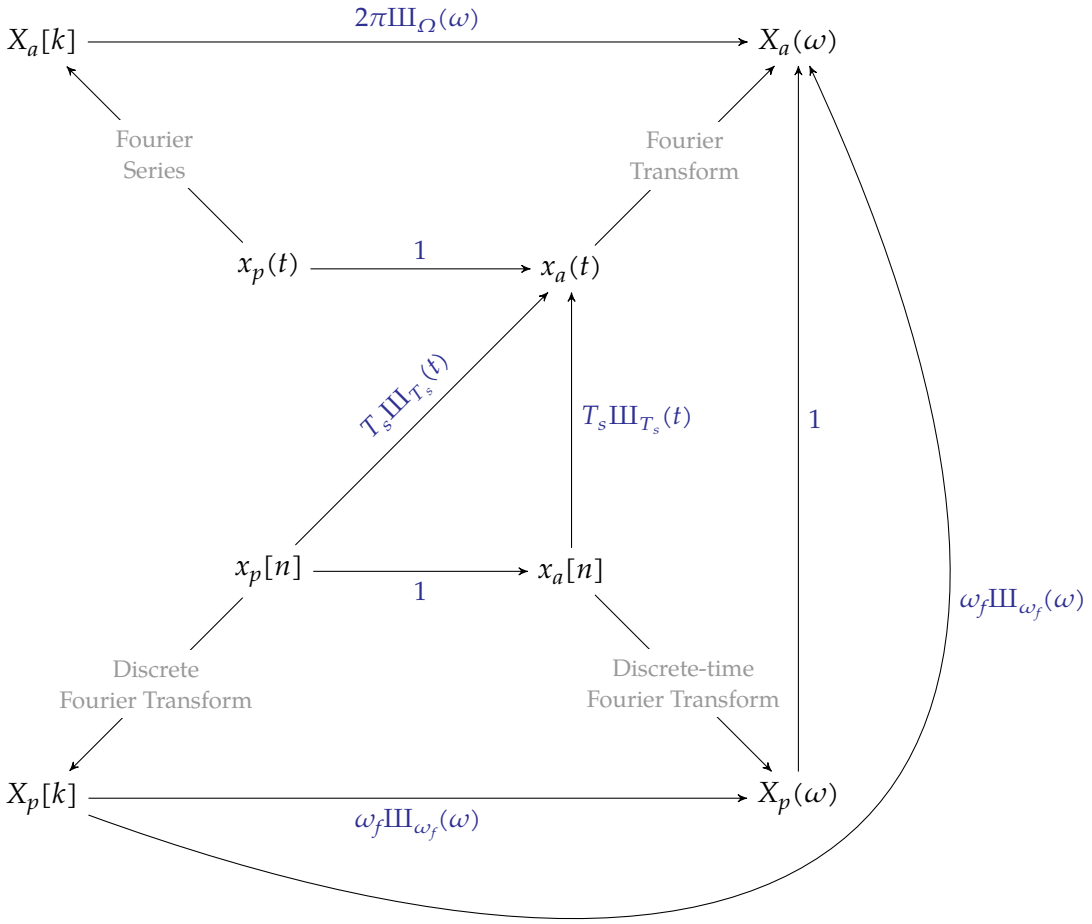


Figure 3.10: Multiplication factors to convert signals to the continuous time and frequency reference domain (using the proper definition for the DtFT)

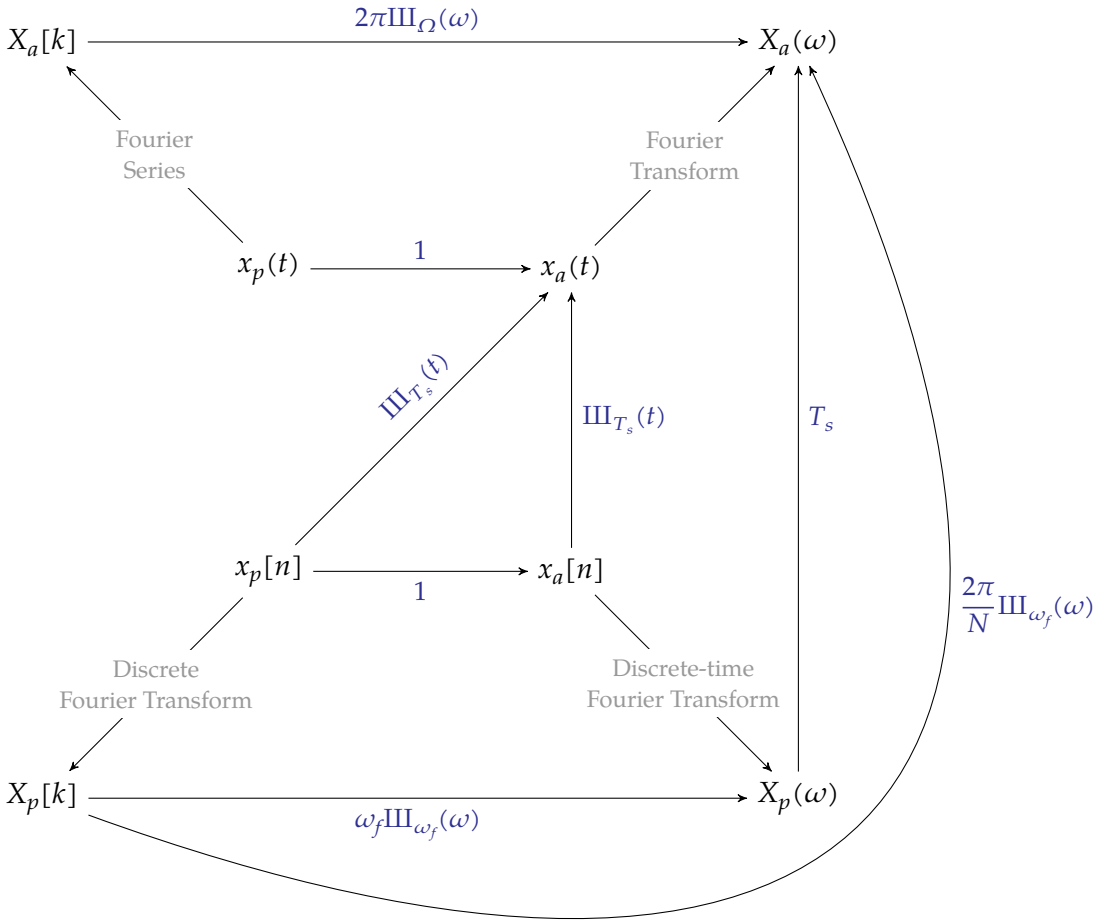


Figure 3.11: Multiplication factors to convert signals to the continuous time and frequency reference domain (using the real definition of the DtFT)

Chapter 4

The Discrete Fourier Transform in practice

We devoted an entire part of this course to signals and signal transforms. Still we need to dive a little deeper into the subject to be able to use signals and signal transforms with confidence.

In this chapter, you will learn about:

- using the one-dimensional Discrete Fourier Transform,
- the multidimensional Fourier Transform and Discrete Fourier Transform,
- information encoding and the human signal perception (hearing and vision),
- analysis windows, and
- the gain of the DFT.

After having read this chapter, some questions will still be left unanswered:

- how do I treat color images using a multidimensional DFT?
- how does the human ear and eye work?
- what did Bill Gates and Linus Torvalds do to receive a cameo in this chapter?

After having read/studied this chapter, you are expected to be able to

- calculate the DFT and the inverse DFT when given a signal (or its spectrum): by hand and using Octave/Matlab,
- understand the effect of zero-padding,
- understand the physical interpretation of one- and two-dimensional Hermitian Fourier pairs,
- understand and explain the effect of analysis windows in terms spectral leak, frequency resolution, dynamic range and noise bandwidth,
- use analysis windows, calculate their frequency effect, and
- understand, explain and use the gain of the DFT (processing gain, and coherent and incoherent integration gain).

4.1 One-dimensional DFT

Let's start by zooming in on the DFT, using a simple example. We will first do this using pencil and paper. However, pencil and paper fall short when having to calculate larger examples. We

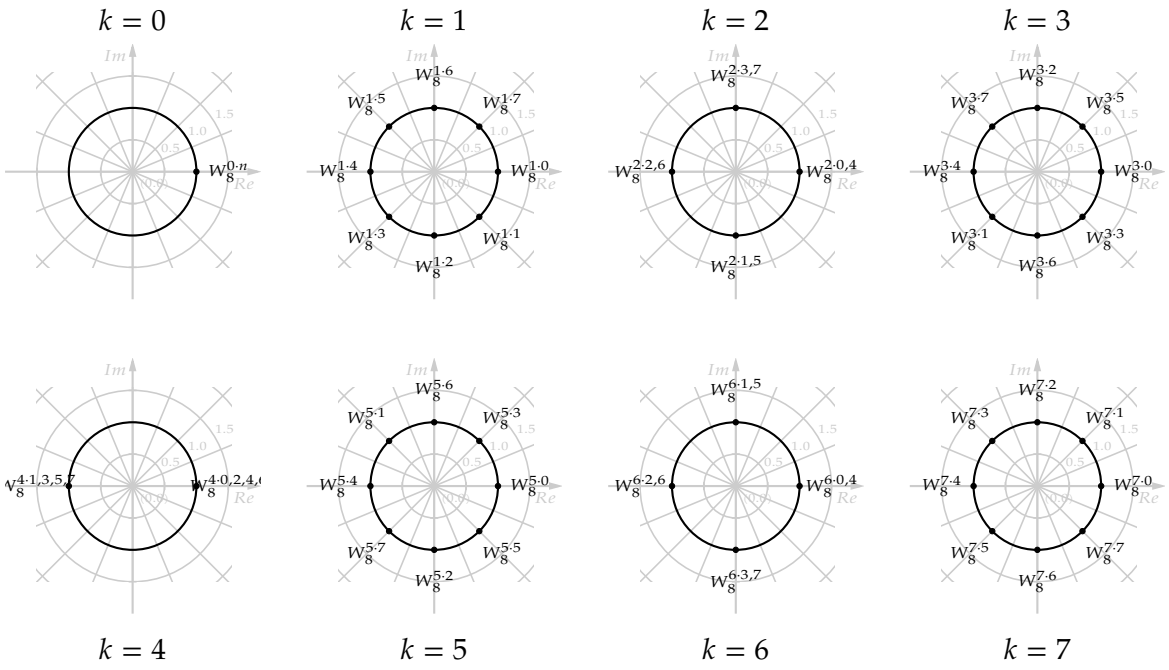


Figure 4.1: Fourier factors W_8^{kn} in the complex plane, for all relevant values of k

therefore secondly will demonstrate how to use your computer to perform the calculations for you. However, the most important message of this section is the interpretation of the resulting data w.r.t. the frequency axis. That will be the third topic.

4.1.1 Using pencil and paper

Consider a signal with the following time sequence:

$$x[n] = \begin{cases} [2,3,5,4,-1,1,-5,-3], & n = 0,1,\ldots 7 \\ 0 & \text{otherwise} \end{cases}$$

Let's calculate the 8-point DFT of this signal. The definition of the DFT tells us that:

$$X[k] = \sum_{n=0}^{7} x[n] \underbrace{\mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{8}}}_{W_8^{kn}}$$

In Figure 4.1, the *Fourier factors* (W_8^{kn}) have been drawn in the complex plane for different values of k. Whenever you have got to calculate a DFT by hand, drawing these small Fourier factor pictures will help you considerably.

Calculating the 8-point DFT is as simple as multiplying $x[n]$ with the appropriate factor W_N^{kn} for every k and n, and adding the resulting products per value of k.
Using a table is a convenient way to organize the data. This has been illustrated in Table 4.1 on the next page. Obviously, besides pencil and paper, a simple calculator eases the job considerably.

As you can see, the calculations very quickly become very lengthy and error prone. For $N = 10...16$, we could still turn the page sideways and sweat this through, but beyond $N = 16$ a modern calculator or computer would come in very handy.

		k							
		0	1	2	3	4	5	6	7
n	x[n]	W_8^{kn}							
0	2	1.00+j0.00	1.00+j0.00	1.00+j0.00	1.00+j0.00	1.00+j0.00	1.00+j0.00	1.00+j0.00	1.00+j0.00
1	3	1.00+j0.00	0.71−j0.71	0.00−j1.00	−0.71−j0.71	−1.00+j0.00	−0.71+j0.71	0.00+j1.00	0.71+j0.71
2	5	1.00+j0.00	0.00−j1.00	−1.00+j0.00	0.00+j1.00	1.00+j0.00	0.00−j1.00	−1.00+j0.00	0.00+j1.00
3	4	1.00+j0.00	−0.71−j0.71	0.00+j1.00	0.71−j0.71	−1.00+j0.00	0.71+j0.71	0.00−j1.00	−0.71+j0.71
4	−1	1.00+j0.00	−1.00+j0.00	1.00+j0.00	−1.00+j0.00	1.00+j0.00	−1.00+j0.00	1.00+j0.00	−1.00+j0.00
5	1	1.00+j0.00	−0.71+j0.71	0.00−j1.00	0.71+j0.71	−1.00+j0.00	0.71−j0.71	0.00+j1.00	−0.71−j0.71
6	−5	1.00+j0.00	0.00+j1.00	−1.00+j0.00	0.00−j1.00	1.00+j0.00	0.00+j1.00	−1.00+j0.00	0.00−j1.00
7	−3	1.00+j0.00	0.71+j0.71	0.00+j1.00	−0.71+j0.71	−1.00+j0.00	−0.71−j0.71	0.00−j1.00	0.71−j0.71
		$X[k]$							
		6.00+j0.00	−0.54−j16.36	1.00−j3.00	6.54+j3.64	−4.00+j0.00	6.54−j3.64	1.00+j3.00	−0.54+j16.36

Table 4.1: Tabular calculation of the 8-point DFT

4.1.2 Using a computer

Given the tabular nature of the calculations, one might be tempted to employ a spreadsheet to perform the calculations. In itself, this is not a bad idea (and you should try it on a rainy day, it will ease your quest to understand the DFT). However, for large values of N, even your spreadsheet will become too slow for your patience. This is not due to using a spreadsheet (though a straightforward C or C++ implementation will be a lot faster), but due to the quadratic computational complexity of the DFT.

Luckily, Cooley and Tukey (in 1965) picked up Gauss's idea on exploiting the regularity of a continued interlaced signal decomposition and implemented a fast version of the DFT: the *Fast Fourier Transform* (*FFT*). We will not go into detail on the nature of the FFT algorithm right now, but the things to remember are:

- the FFT does exactly the same job as a DFT: it is not an approximate version, so there is no loss of accuracy (on the contrary: round-off errors are smaller for the FFT);
- the FFT has an $N \log_2 N$ computational complexity, as opposed to the DFT's N^2 complexity.

The two facts above are the cause that you will not regularly find DFT algorithms. Instead, all mathematical software packages (and DSP libraries) have FFT routines on board.

For the particular DSP processor you encounter in developing a DSP application, look for the OEM's standard library: it will contain a most efficient FFT.

Even very good free algorithms exist (e.g., FFTW [FJ08]). Search the web or ask your local FFT expert for advice.

In most mathematical software packages, the routines you want to use are named:

- `fft`,
- `ifft`.

In any case, programming an FFT yourself for a real-world application is the last thing you should be tempted to do (unless it's for the pure fun of it).

It is time for some fiddling with OCTAVE/MATLAB [Eat07, MAT15]. Make sure you have the signal processing, audio and image processing toolboxes installed. We'll use the simple example on page 68 once again.

Start the OCTAVE/MATLAB-interpreter. After the start-up messages, the interpreter's command prompt will appear.

First, enter the time vector $x[n]$.

```
x = [ 2; 3; 5; 4; -1; 1; -5; -3];
```

Then, calculate its discrete Fourier transform $X[k]$:

```
X = fft(x)
==> X =

    6.00000 +  0.00000i
   -0.53553 - 16.36396i
    1.00000 -  3.00000i
    6.53553 +  3.63604i
   -4.00000 +  0.00000i
    6.53553 -  3.63604i
    1.00000 +  3.00000i
   -0.53553 + 16.36396i
```

It's as easy as that. We used a column vector to get the output as a column vector as well (which makes it easier to fit the output on paper). However, using a row vector is equally good.

Plotting the magnitude and phase spectrum is also a piece of cake:

```
subplot(2,1,1);
plot( 0:7, abs(X), 's' );
subplot(2,1,2);
plot( 0:7, angle(X), 's' );
```

The help facility should get you going very quickly in understanding the above.

```
help fft
==> <output cut>

help ifft
==> <output cut>

help subplot
==> <output cut>

help plot
==> <output cut>
```

Unleash the engineer in yourself and make yourself acquainted with this powerful mathematical tool!

Exercises
Some exercises on applying the DFT. Note that by following the scheme of the first four exercises, you can easily make extra exercises yourself.

Exercise 4.1.2-1: Calculate the DFT of the following signal by hand:

$$x = [-1 \quad 3.2 \quad 2 \quad 5 \quad -1.2 \quad 1.3]$$

Exercise 4.1.2-2: Verify your result obtained above by applying the DFT using Octave/Matlab.

Exercise 4.1.2-3: Plot the magnitude and the phase of the result you obtained (using Octave/Matlab).

Exercise 4.1.2-4: Verify your result by using the iDFT and see whether you can find the original sequence again.

Exercise 4.1.2-5: (*) Try equipping the magnitude and phase plots with a title, axis labels and a grid.

Exercise 4.1.2-6: (*) Try plotting the magnitude on a decibel scale.

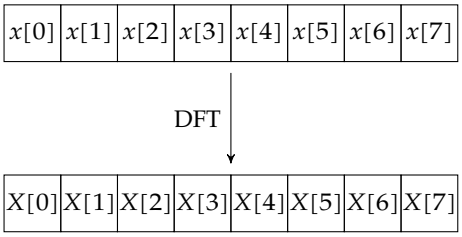
Exercise 4.1.2-7: (*) Try fixing the plots' frequency axis range to a range you choose.

4.1.3 Interpreting the results w.r.t. to the frequency axis

Mapping k to ω So, the result of our DFT is a vector of numbers $X[k]$ that is related to $x[n]$ and is equally long:

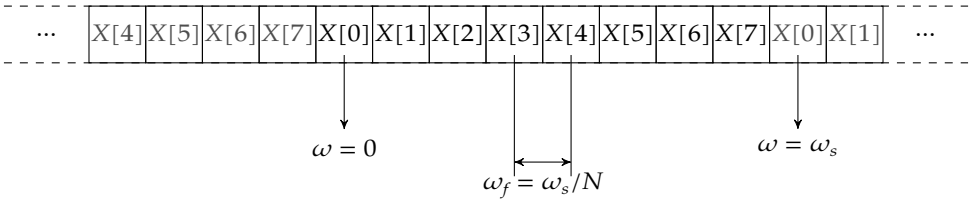
$$x[n] \xrightarrow{\text{DFT}} X[k]$$

Graphically (for $N = 8$):



Obtaining that row of numbers $X[k]$ is easy. However, interpreting the meaning of those numbers is essential to be able to make effective use of the DFT. More specifcally: what frequency does a particular $X[k]$ (e.g. with $k = 3$) belong to?

The corner stone of understanding this relationship is realizing that the result of the DFT is a periodic spectrum with period ω_s and pitch ω_f. Therefore the first bin (for $k = 0$) corresponds to $\omega = 0$ and the first repetition of the value (for $k = N$, i.e. 8 in our case) corresponds to $\omega = \omega_s$.



Closer analysis of this diagram leads to the fundamental equation:

$$\omega = \frac{k}{N}\omega_s$$

Therefore, the value $X[3]$ belongs to $\omega = 3/8\,\omega_s$. The value $X[6]$ corresponds to $\omega = 6/8\,\omega_S$, but equally well to $\omega = -2/8\,\omega_s$ (due to aliasing).

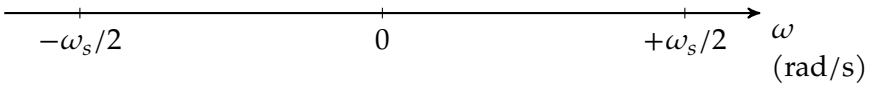
The other way round, given a value for ω, we can determine which value of k we need to take. E.g., assuming $\omega_s = 1\,\text{kHz}$ in the example above, we can easily derive that $\omega = 250\,\text{Hz}$ corresponds to $k = 2$, using the equation:

$$k = N\frac{\omega}{\omega_s}$$

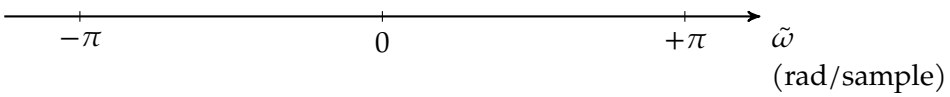
What happens if the value of k ends up not to be integer? In that case there is no answer..... yet. In that case, we need a technique called zero padding that we will treat in section 4.4.

Normalized frequencies Very often, DSP calculations are carried out while making abstraction of the actual sample frequency. The sampled data is just considered as a row of numbers with a distance in between them equal to '1'. This way of working corresponds to setting $T_s = 1$. This means: $\omega_s = 2\pi$.

A DtFT or DFT spectrum normally has its primary period in between $-\omega_s/2$ and $\omega_s/2$. The unit of the frequency axis is rad/s.



Given $T_s = 1$ this means that the primary period is located in between $-\pi$ and π. To indicate the use of normalized frequencies, often a tilde is typeset over the frequency variable ω.



The unit of the $\tilde{\omega}$ frequency axis is rad/sample. Understanding this, requires a detailed dimensional analysis of the argument of a periodic function with period 2π, e.g. a sine wave:

$$\sin\Big(\underbrace{\omega \cdot n \cdot T_s}_{=\alpha}\Big)$$

Let's first try to find out which unit ω has. If we use $[\cdot]$ as the dimension operator, the requirement is that $[\alpha] = \text{rad}$.

Given

$$[\alpha] = [\omega] \cdot [n] \cdot [T_s]$$

it is easy to see that

$$\begin{aligned}[\omega] &= \frac{[\alpha]}{[n] \cdot [T_s]} \\ &= \frac{\text{rad}}{\text{sample} \cdot \frac{\text{s}}{\text{sample}}} \\ &= \text{rad/s}\,.\end{aligned}$$

Now, we can proceed to determine the unit of $\tilde{\omega}$:

$$\alpha = \underbrace{\omega \cdot T_s}_{=\tilde{\omega}} \cdot n\,.$$

And therefore,

$$\begin{aligned}[\tilde{\omega}] &= [\omega] \cdot [T_s] \\ &= \frac{\text{rad}}{\text{s}} \cdot \frac{\text{s}}{\text{sample}} \\ &= \text{rad/sample}\,.\end{aligned}$$

4.2 The fast Fourier transform

Remember the definition of the discrete Fourier transform:

$$\begin{aligned}X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{kn} \\ x[n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn}\end{aligned} \tag{4.1}$$

with

$$W_N = \mathrm{e}^{-\frac{\mathrm{j}2\pi}{N}}$$

and

$$W_N^{kn} = (W_N)^{kn} = \mathrm{e}^{-\frac{\mathrm{j}2\pi kn}{N}}$$

The constants W are often called *twiddle factors*.

We can organize these calculations in a smart way to reduce the amount of computational work, in two ways:

- decimation in time, and
- decimation in frequency.

For convenience reasons, let's assume N to be even.

This idea was first formulated by Johann Carl Friedrich Gauss (in the 19th century), and later implemented by Cooley and Tukey (in 1965).

4.2.1 Decimation in time

4.2.1.1 Derivation

We can split (4.1) into two subseries, by gathering the even-numbered time points and the odd-numbered time points:

$$\begin{aligned} X[k] &= \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_N^{k2n} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{k(2n+1)} \\ &= \underbrace{\sum_{n=0}^{\frac{N}{2}-1} x[2n] W_{\frac{N}{2}}^{kn}}_{A[k]} + W_N^k \underbrace{\sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_{\frac{N}{2}}^{kn}}_{B[k]} \end{aligned} \tag{4.2}$$

Expression $A[k]$ corresponds to the DFT of the half-length signal, composed by the even-numbered time samples. Expression $B[k]$ corresponds to the DFT of the half-length signal, composed by the odd-numbered time samples.

In a normal DFT, we'd only calculate the spectrum for $k = 0, 1, \ldots, N-1$ with N the length of the time-domain signal. In the case of $A[k]$ and $B[k]$ we need to evaluate the DFT expression for values of k outside of that primary range. Given the periodicity of the DFT, this is easily solved without any extra evaluations of the expressions of $A[k]$ and $B[k]$ by:

$$\begin{aligned} A\left[k\right] &= A\left[k - \frac{N}{2}\right], \quad k = \frac{N}{2}, \ldots, N-1 \\ B\left[k\right] &= B\left[k - \frac{N}{2}\right], \quad k = \frac{N}{2}, \ldots, N-1 \end{aligned}$$

In this way, the calculation of an N-point DFT has been reduced to calculating two $N/2$-point DFTs and some extra multiplications and additions.

Considering (4.2) for $N = 2$ allows further simplification:

$$X[k] = x[0] + W_N^k x[1]$$

4.2.1.2 Computational effort

The computational effort of the original DFT can be approximated as:

$$T_{DFT} = N^2 (T_{add} + T_{mul})$$

The computational effort of the FFT can be approximated as:

$$T_{DFT} = \underbrace{2\,(N/2)^2\,(T_{add} + T_{mul})}_{\text{Work at level } N/2} + \underbrace{N(T_{add} + T_{mul})}_{\text{Work at level } N}$$

Note that the first term corresponds to the work to be done at level $N/2$, while the second term corresponds to the work to be done at the (current) level N

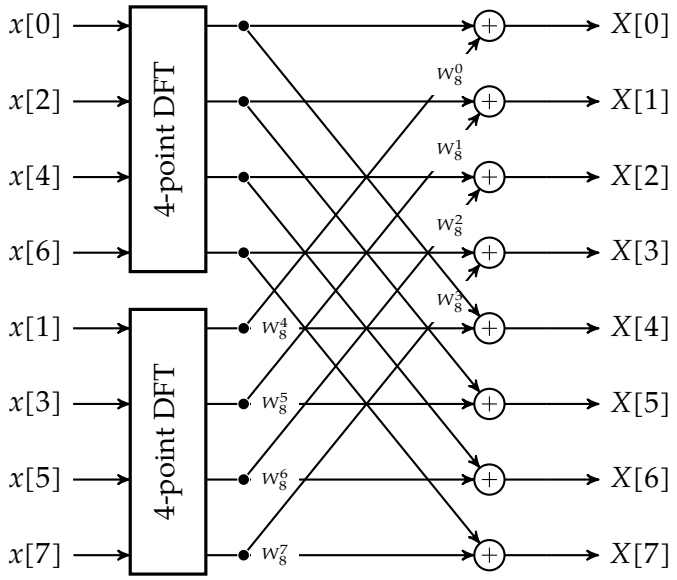


Figure 4.2: Signal flow representation of (4.2) for $N = 8$. A label on a signal path indicates a signal gain on the path. Signal paths without label have gain 1 by default.

4.2.1.3 Graphical representation

If we consider (4.2) graphically for $N = 8$ (as an example), we obtain the signal flow graph of Figure 4.2.

If we further expand the placeholders for the 4-points DFT in Figure 4.2 using the same principle and recursively keep doing that, we obtain the flow graph of a decimation-in-time *Fast Fourier transform* (*FFT*). This has been done in Figure 4.3. The typical 4-arrow pattern as visible in the leftmost column, is called a *butterfly*, for apparent reasons.

Note that the order of the samples on the left of the diagram is bit reversed. Many digital signal processors have hardware features to allow for bit-reversal addressing, which makes accessing the samples in the correct order easy.
Also note that many of the coefficients turn out to be trivial (e.g. $W_N^0 = 1$).

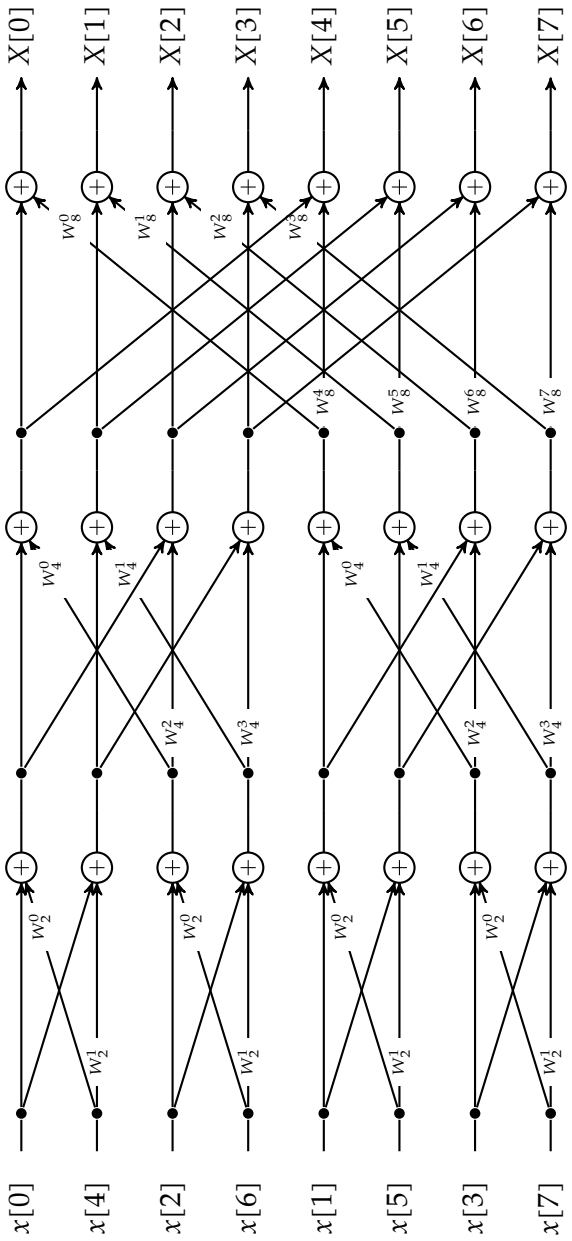


Figure 4.3: Fully expanded signal flow representation of (4.2) (decimation in time) for $N = 8$. A label on a signal path indicates a signal gain on the path. Signal paths without label have gain 1 by default.

4.2.2 Decimation in frequency

4.2.2.1 Derivation

We can split (4.1) into two subseries, by gathering the first half of time points and the second half of time points:

$$\begin{aligned} X[k] &= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_N^{kn} + \sum_{n=\frac{N}{2}}^{N-1} x[n]W_N^{kn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_N^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x[n+N/2]W_N^{k(n+N/2)} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_N^{kn} + W_N^{\frac{kN}{2}} \sum_{n=0}^{\frac{N}{2}-1} x[n+N/2]W_N^{kn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_N^{kn} + (-1)^k \sum_{n=0}^{\frac{N}{2}-1} x[n+N/2]W_N^{kn} \end{aligned} \tag{4.3}$$

The latter step is based on the fact that:

$$W_N^{\frac{kN}{2}} = \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}\frac{kN}{2}} = \mathrm{e}^{\mathrm{j}\pi k} = (-1)^k$$

Our goal is to recognize half-length DFTs in (4.3). We therefore consider the case for k even and k odd separately. This allows reworking the twiddle factors to allow for recognizing the half-length DFTs.

When k is even, i.e $k = 2m$, we can continue simplifying (4.3):

$$\begin{aligned} X[2m] &= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_N^{2mn} + (-1)^{2m} \sum_{n=0}^{\frac{N}{2}-1} x[n+N/2]W_N^{2mn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_{\frac{N}{2}}^{mn} + \sum_{n=0}^{\frac{N}{2}-1} x[n+N/2]W_{\frac{N}{2}}^{mn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} \left(x[n] + x[n+N/2]\right) W_{\frac{N}{2}}^{mn} \end{aligned} \tag{4.4}$$

This expression corresponds to the DFT of the half-length signal, obtained by summing the first and the second half of the time samples.

When k is odd, i.e. $k = 2m+1$, we also can continue simplifying (4.3):

$$\begin{aligned} X[2m+1] &= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_N^{(2m+1)n} + (-1)^{2m+1} \sum_{n=0}^{\frac{N}{2}-1} x[n+N/2]W_N^{(2m+1)n} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_N^n W_{\frac{N}{2}}^{mn} - \sum_{n=0}^{\frac{N}{2}-1} x[n+N/2]W_N^n W_{\frac{N}{2}}^{mn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} \left(x[n] - x[n+N/2]\right) W_N^n W_{\frac{N}{2}}^{mn} \end{aligned} \tag{4.5}$$

This expression corresponds to the DFT of the half length signal obtained by subtracting the second half of the time samples from the first half and multiplying with W_N^n.

As in the decimation-in-time case we can calculate $X[k]$ for values of $k \geq N/2$ using:

$$X[k] = X\left[k - \frac{N}{2}\right], \quad k = \frac{N}{2}, \ldots, N-1$$

In this way, the calculation of an N-point DFT has been reduced to calculating two $N/2$-point DFTs and some extra multiplications and additions.

4.2.2.2 Computational effort

The computational effort for the former can be approximated as:

$$T_{DFT} = N^2(T_{add} + T_{mul})$$

The computational effort of the latter can be approximated as:

$$T_{DFT} = \left(2\left(\frac{N}{2}\right)^2 + N\right)(T_{add} + T_{mul})$$

The careful reader will have noticed that we overestimate the number of multiplications slightly (as the multiplications with W_N^n are only required for the odd spectral components).

4.2.2.3 Graphical representation

If we consider (4.4) and (4.5) graphically for $N = 8$ (as an example), we obtain the signal flow graph of Figure 4.4.

If we further expand the placeholders for the 4-points DFT in Figure 4.2 using the same principle and recursively keep doing that, we obtain the flow graph of a decimation-in-frequency *Fast Fourier transform* (*FFT*). This has been done in Figure 4.5. The typical 4-arrow pattern as visible in the leftmost column, is also called a *butterfly*, for apparent reasons.

Note that the order of the spectral components on the right of the diagram is bit reversed. Many digital signal processors have hardware features to allow for bit-reversal addressing, which makes accessing the spectral lines in the correct order easy.
Also note that many of the coefficients turn out to be trivial (e.g. $W_N^0 = 1$).

4.3 Boosting the DFT/FFT

In case of real signals, we can exploit the nature of the DFT/FFT to double its performance. The technique presented works independently from the fact whether we use the DFT or the FFT algorithm. Therefore, in this section, one might replace DFT by FFT in all places.

Let's assume in this section that N is even. This eases the discussion and suits our needs sufficiently in view of the performance doubling effect.

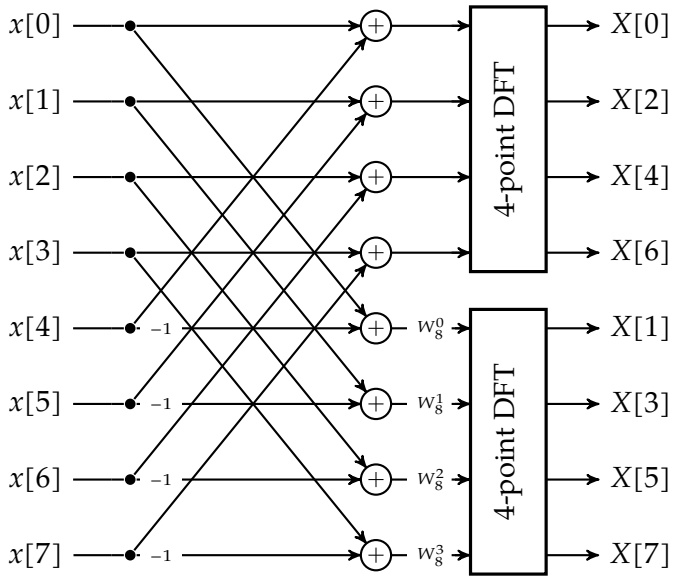


Figure 4.4: Signal flow representation of (4.4) and (4.5) for $N = 8$. A label on a signal path indicates a signal gain on the path. Signal paths without label have gain 1 by default.

4.3.1 Observation: we're working too hard

An observation we made earlier is that the DFT of a real signal is hermitic, i.e.

$$X^*[k] = X[k]$$

or

$$\overline{X[-k]} = X[k]$$

Note that $X[k]$ is periodic, therefore, the latter equation can also be written as $\overline{X[N-k]} = X[k]$.

We translated this observation earlier in a polar form, stating that the magnitude spectrum is even and the phase spectrum is odd.

In fact, this means that even when calculating only half of the DFT coefficients of a real signal, we know all there is to know. The other half can be completed based on the hermiticity mentioned above.

Knowing that, we could easily restrict ourselves to calculating the first $N/2$ points and derive the second half.

Specifically, we calculate

$$X[k] = \sum_{n=0}^{N-1} x[n]\, \mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{N}}, \quad k = 0, 1, \ldots \frac{N}{2} - 1$$

and derive

$$X[N-1-k] = \overline{X[k]}, \quad k = 0, 1, \ldots \frac{N}{2} - 1.$$

However, this is suboptimal, as the FFT algorithm is optimized for calculating N frequency points when starting with N time points. However, there is a solution that combines optimal use of the FFT and halving the computational work for a real signal.

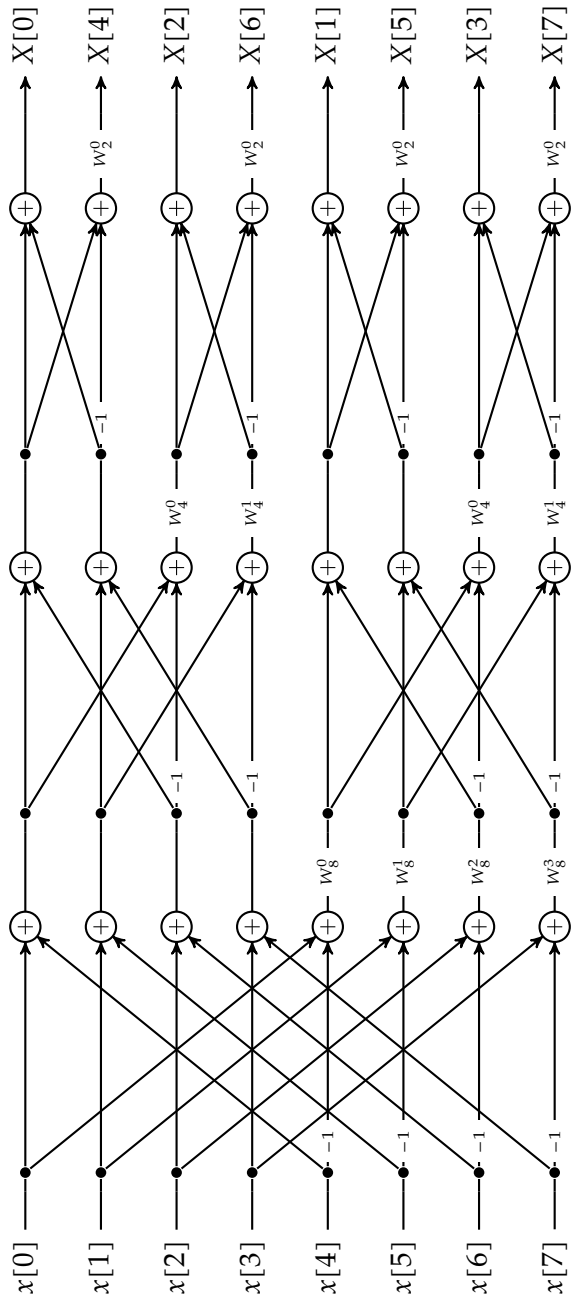


Figure 4.5: Fully expanded signal flow representation of (4.4) and (4.5) (decimation in frequency) for $N = 8$. A label on a signal path indicates a signal gain on the path. Signal paths without label have gain 1 by default.

4.3.2 Symmetry to the rescue

Symmetry properties of the DFT of a real signal We start by considering the definition of the DFT

$$X[k] = \sum_{n=0}^{N-1} x[n]\, \mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{N}}$$

Using Euler's formula, we can rewrite this as:

$$\begin{aligned} X[k] &= \underbrace{\sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi kn}{N}\right)}_{X_A[k](} - \mathrm{j} \underbrace{\sum_{n=0}^{N-1} x[n] \sin\left(\frac{2\pi kn}{N}\right)}_{X_B[k]} \\ &= X_A[k] - \mathrm{j} X_B[k] \end{aligned}$$

Note that if $x[n]$ is a real signal, $X_A[k]$ and $X_B[k]$ are both real and an even and odd function of k respectively.

Symmetry properties of the DFT of a real signal in an imaginary disguise Similarly, consider another real signal $y[n]$, but let's put it in an *imaginary disguise* by multiplying it with j.

$$v[n] = \mathrm{j} y[n]$$

It's DFT/FFT is:

$$\begin{aligned} V[k] &= \sum_{n=0}^{N-1} v[n] \cos\left(\frac{2\pi kn}{N}\right) - \mathrm{j} \sum_{n=0}^{N-1} v[n] \sin\left(\frac{2\pi kn}{N}\right) \\ &= \underbrace{\sum_{n=0}^{N-1} y[n] \sin\left(\frac{2\pi kn}{N}\right)}_{Y_B[k](} + \mathrm{j} \underbrace{\sum_{n=0}^{N-1} y[n] \cos\left(\frac{2\pi kn}{N}\right)}_{Y_A[k]} \\ &= Y_B[k] + \mathrm{j} Y_A[k] \end{aligned}$$

Not that if $y[n]$ is a real signal, $Y_A[k]$ is even and $Y_B[k]$ is odd, and both are real functions of k.

Telling twins apart If we add two (complex) signals $x_1[n]$ and $x_2[n]$ into a sum signal $y[n]$

$$y[n] = x_1[n] + x_2[n]$$

there's no way in telling these two signals apart again from the sum $y[n]$ alone. It's like meeting perfect twins: telling them apart is in general impossible. However, if the twins have significant and noticeable differences, distinguishing them is easy.

This also holds for signals: if one of the signals is even $x_e[n]$ and the other is odd $x_o[n]$ (a significant and noticeable difference), we can sum them and afterwards tell them apart again. Let's investigate this. Assume

$$y[n] = x_e[n] + x_o[n] \tag{4.6}$$

with $x_e[n] = x_e[N-n]$ and $x_o[n] = -x_o[N-n]$.

In view of this, we can calculate an expression for $y[-n]$. Indeed:

$$\begin{aligned} y[N-n] &= x_e[N-n] + x_o[N-n] \\ &= x_e[n] - x_o[n] \end{aligned} \tag{4.7}$$

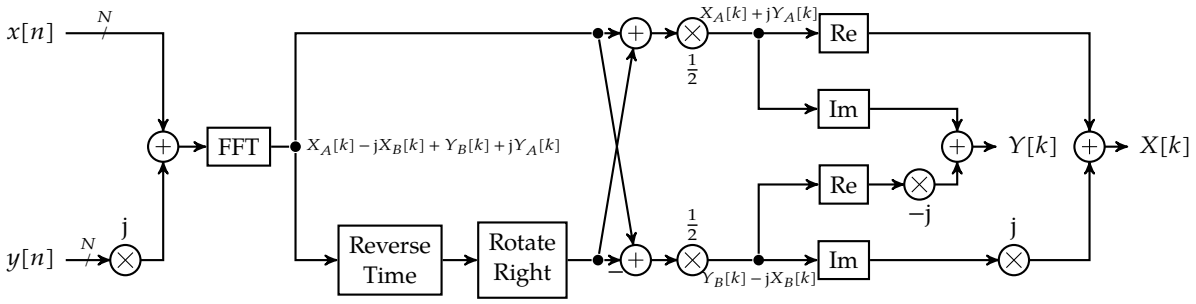


Figure 4.6: Flow graph to analyze two real signals using a single DFT run. The input signals are vectors of length N.

Combining (4.6) and (4.7) leads to a set of simultaneous equations

$$\begin{cases} y[n] = x_e[n] + x_o[n] \\ y[N-n] = x_e[n] - x_o[n] \end{cases}$$

that we can solve for $x_e[n]$ and $x_o[n]$, leading to

$$\begin{cases} x_e[n] = \dfrac{y[n] + y[N-n]}{2} \\ x_o[n] = \dfrac{y[n] - y[N-n]}{2} \end{cases} \tag{4.8}$$

4.3.3 Giving a boost to our work efficiency

If we calculate the DFT/FFT of the sum of $x[n]$ and $v[n] = \mathrm{j}y[n]$, and use the notation of section 4.3.2, we obtain:

$$\begin{aligned} x[n] + v[n] = x[n] + \mathrm{j}y[n] &\xrightarrow{\text{DFT}} \underbrace{X_A[k] - \mathrm{j}X_B[k]}_{\text{DFT}(x[n])} + \underbrace{Y_B[k] + \mathrm{j}Y_A[k]}_{\text{DFT}(\mathrm{j}y[n])} \\ &\xrightarrow{\text{DFT}} \underbrace{X_A[k]}_{\text{even}} + \underbrace{Y_B[k]}_{\text{odd}} + \mathrm{j}\left(\underbrace{Y_A[k]}_{\text{even}} - \underbrace{X_B[k]}_{\text{odd}} \right) \\ &\xrightarrow{\text{DFT}} \underbrace{X_A[k] + \mathrm{j}Y_A[k]}_{\text{even}} + \underbrace{Y_B[k] - \mathrm{j}X_B[k]}_{\text{odd}} \end{aligned}$$

Now, this is an interesting result, as we see that distinguishing the even and odd parts (using (4.8)) and within those parts, distinguishing the real and imaginary parts, allows us telling all the components apart.

This leads to the idea of Figure 4.6.

4.4 Zero padding

Padding a signal or a spectrum with zeros means adding zeros to it. We call this 'zero padding'. As we will see, this operation has an interesting effect on the resulting (forward/inverse) DFT.

4.4.1 Time-domain zero padding

Consider a time- and bandwidth-limited signal.[1] The nonzero parts ($0 \leq t \leq T$, and $-\omega_B \leq \omega \leq \omega_B$) have been symbolically indicated by the colored area in Figure 4.7.

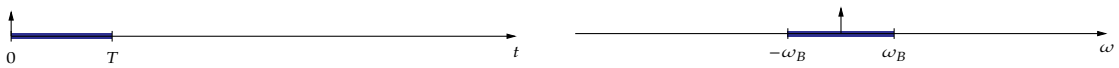


Figure 4.7: An arbitrary time- and bandwidth-limited signal

If we select a sample grid T_s (step (a) in Figure 4.8), the sample frequency ω_s (imposing aliasing effects) is also fixed. The next thing we can do (step (b)) is select a number of points N, such that the full analysis period $[0, T_f]$ encompasses the signal's time span $[0, T]$. The value of N determines T_f (step (c)) and also automatically determines the frequency grid ω_f. In this way, the frequency resolution of your DFT is fixed.

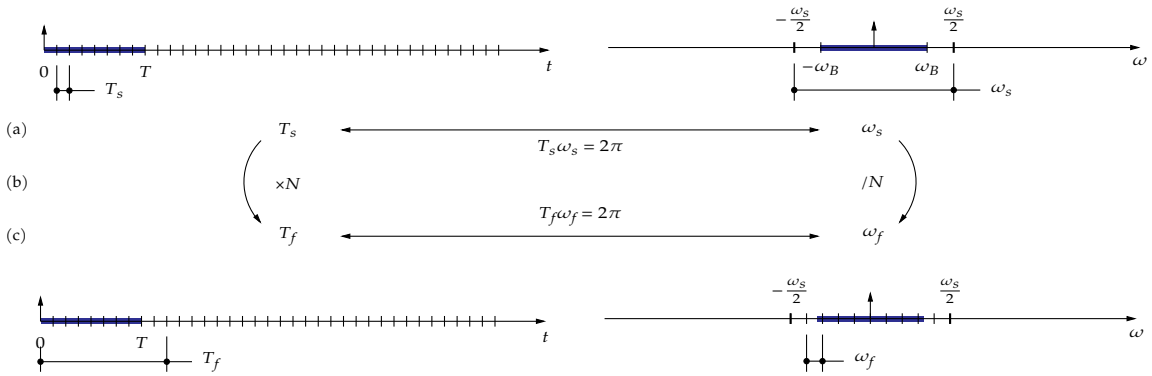


Figure 4.8: DFT parameter selection process without zero padding

Now, let's apply zero padding. We start by selecting the same sample grid T_s (step (a) in Figure 4.9 on the next page), imposing the very same sample frequency ω_s. Now let's assume (step (d)), that we select a number of points N' that is twice as big as the original number: $N' = 2N$. This value determines T_f' (step (e)) and also determines the frequency grid ω_f'. In this way the frequency resolution of your DFT is doubled!

This is the effect of time-domain zero padding: it increases the frequency resolution of your DFT. Obviously, we can pick any number for N' as long as it's bigger than N.

Now, aren't the spectral values of the points that were in the original DFT grid affected by this operation? No. We'll illustrate this for $N' = 2N$. Consider a particular frequency point $k = l$ in the original DFT. Its value equals

$$X_N[l] = \sum_{n=0}^{N-1} x[n]\, \mathrm{e}^{-\mathrm{j}2\pi\frac{ln}{N}}$$

[1] One can theoretically prove that a signal cannot be time- and bandwidth-limited at the same time. However, let's assume that the signal is such that the amount of energy falling outside the considered time and bandwidth limits is neglectable.

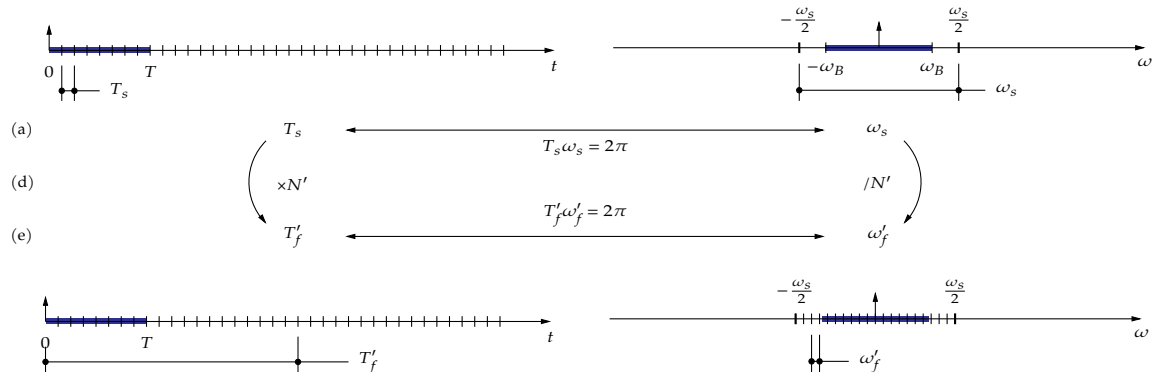


Figure 4.9: DFT parameter selection process with zero padding

Now let's calculate the very same frequency point in the new DFT grid. The position of the frequency component we calculated above now lies at $k = l' = 2l$. Its value equals

$$\begin{aligned} X_{N'}[l'] &= \sum_{n=0}^{N'-1} x[n]\, \mathrm{e}^{-\mathrm{j}2\pi\frac{l'n}{N'}} = \sum_{n=0}^{2N-1} x[n]\, \mathrm{e}^{-\mathrm{j}2\pi\frac{2ln}{2N}} \\ &\downarrow \quad x[n] = 0, \forall n \geq N \\ &= \sum_{n=0}^{N-1} x[n]\, \mathrm{e}^{-\mathrm{j}2\pi\frac{ln}{N}} \overset{!!}{=} \end{aligned}$$

Conclusion: the original spectral values are not affected.

Remarks Zero padding offers an interesting way to use the DFT to generate graphs for the Discrete-time Fourier Transform.

- If the signal you want to analyze is time limited, you can apply the DFT with a sufficient number of zeros padded at the end to make the frequency graph seem continuous.
- If the signal is periodic, just take as many samples to cover exactly an integer multiple of periods.
- If the signal is not time limited, and not periodic, you will have to revert to analytical techniques to avoid aliasing effects. You will learn more about these effects in section 4.7 on page 97.
- Practice is more involved than theory! Be aware that some experience is required when applying this technique. Make sure to exercise!

Exercises

Some exercises on using the DFT combined with zero padding

Exercise 4.4.1-1: Perform zeropadding on the following signal until it has length 8, and calculate the corresponding zero-padded DFT

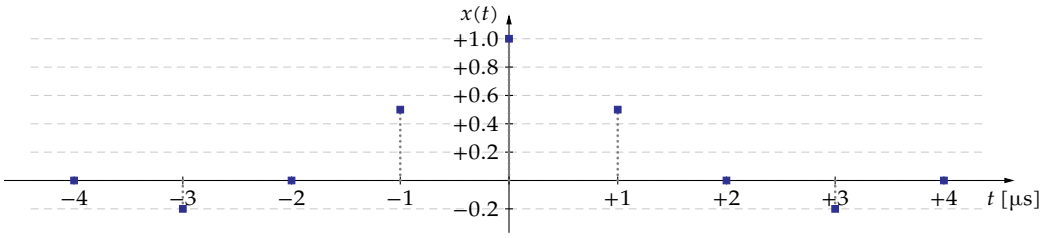
$$x = [\underbrace{-1}_{n=0} \quad 3.2 \quad 2 \quad 5 \quad -1.2 \quad 1.3]$$

Exercise 4.4.1-2: Pad the example signal on page 68 with M zeros, and plot the magnitude and phase of the newly obtained signal. Do this for:

- $M = 8$
- $M = 24$
- $M = 56$
- $M = 248$

Now compare these results with the original plots. What do you observe?

Exercise 4.4.1-3: (*) Generate a graph (using OCTAVE/MATLAB) of the DtFT of the following sampled function:



Make sure you understand the frequency scale of the resulting DtFT!
Hint: You might want to consult the Fourier family transition diagram of Figure 3.11 on page 65.

4.4.2 Frequency-domain zero padding

Consider a time- and bandwidth-limited signal. The nonzero parts ($0 \leq t \leq T$, and $-\omega_B \leq \omega \leq \omega_B$) have been symbolically indicated by the gray area of Figure 4.10.

Figure 4.10: An arbitrary time- and bandwidth-limited signal

If we select a frequency sample grid ω_f (step (a) in Figure 4.11 on the next page), the full analysis period T_f is also fixed. The next thing we can do (step (b)) is select a number of points N, such that the primary frequency period $[-\omega_s/2, \omega_s/2]$ encompasses the signal's bandwidth $[-\omega_B, \omega_B]$. The value of N determines ω_S (step (c)) and also automatically determines the time-domain sample grid T_s. In this way, the time-domain resolution of your inverse DFT is fixed.

Now, let's apply zero padding. We start by selecting the same frequency grid ω_f (step (a) in Figure 4.12), imposing the very same full analysis period T_f. Now let's assume (step (d)), that we select a number of points N' that is twice as big as the original number: $N' = 2N$. This value determines ω_s' (step (e)) and also determines the sample grid T_s'. In this way the time-domain resolution of your inverse DFT is doubled!

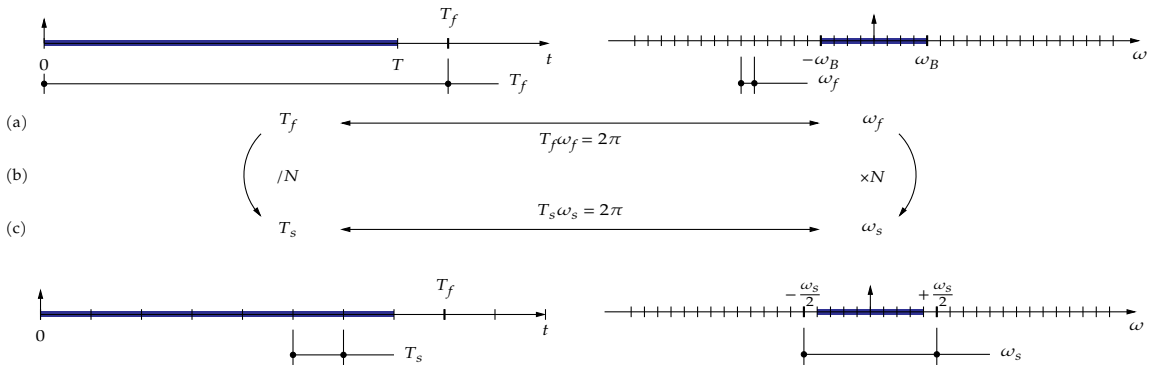


Figure 4.11: Inverse DFT parameter selection process without zero padding

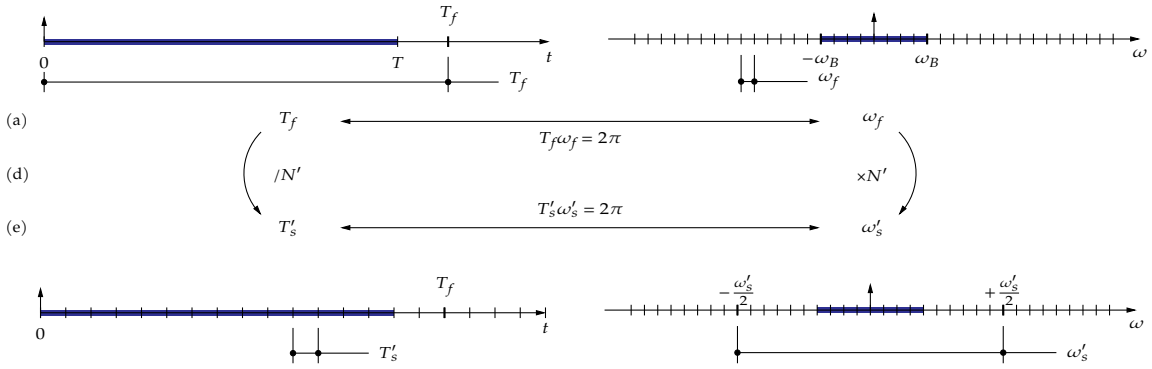


Figure 4.12: Inverse DFT parameter selection process with zero padding

This is the effect of frequency-domain zero padding: it increases the time-domain resolution of your inverse DFT. Obviously, we can pick any number for N' as long as it's bigger than N.

Now, aren't the time-domain values of the points that were in the original inverse DFT grid affected by this operation? No, if we make abstraction of the following detail: they are scaled by a factor N/N'. Try to prove this yourself. Take a look at the similar proof in the section on time-domain zero padding on page 83 if you lack the inspiration.

Remarks Zero padding offers an interesting way to use the DFT to generate graphs for the Discrete-time Fourier Transform.

- When padding a spectrum with zeros, make sure you add them at the right spot, i.e. at frequencies beyond the original $\omega_s/2$. Usually, this means inserting zeros in the middle instead of appending them.
- If the original time-domain signal was real, make sure to keep the symmetry in the spectrum when padding it with zeros.
- Practice is more involved than theory! Be aware that some experience is required when applying this technique. Make sure to exercise!

Exercises
Some exercises on frequency-domain zero padding

Exercise 4.4.2-1: Consider the following spectrum:

$$X = [\underset{\smile}{\underset{k=0}{1}} \quad 2+\mathrm{j} \quad 2-\mathrm{j}]$$

Pad this spectrum with 5 zeros and calculate the zero-padded inverse DFT.

Exercise 4.4.2-2: Pad the DFT of the example signal on page 68 with M zeros, and plot the time-domain representation of the newly obtained signal. Do this for:

- $M = 8$
- $M = 24$
- $M = 58$
- $M = 248$

Now compare these results with the time-domain signal. What do you observe?

4.5 Multidimensional DFT

When imagining real-world signals you are likely to find yourself thinking of measurements of scalar physical quantities as a function of time. Some examples of such signals are:

- for audio applications: air (sound) pressure (as a function of time);
- for control applications in chemical process technology: pressure, temperature, flow, concentration, fluid level in a tank, etc. (all as a function of time);
- for medical applications: heartbeat rate, blood pressure, body temperature, concentration of blood gases, etc. (all as a function of time);
- for automotive applications: speed, power, torque, intake air temperature, gasoline level, turbo pressure, concentration of pollutants in the exhaust gases, etc. (once again as a function of time).

The signals mentioned above are all functions of a one-dimensional variable, time. All transforms of the Fourier Transform family can be readily applied by straightforward application of the definitions. Nothing new to be noted here.

However, though we developed the Fourier Transform family in the framework of one-dimensional signals, the definitions are easily extended to multidimensional signals. The technique consists of repeatedly applying the Fourier Transform, introducing a frequency variable for every independent variable of the signal.

Graphically:

$$x(t_1,t_2) \overset{\mathcal{F}_{t_1}}{\longrightarrow} X_1(\omega_1,t_2) \overset{\mathcal{F}_{t_2}}{\longrightarrow} X_{1,2}(\omega_1,\omega_2) \overset{\mathcal{F}^{-1}_{\omega_2}}{\longrightarrow} X_1(\omega_1,t_2) \overset{\mathcal{F}^{-1}_{\omega_1}}{\longrightarrow} x(t_1,t_2)$$

or, treating t_1 and t_2 in the opposite order:

$$x(t_1,t_2) \overset{\mathcal{F}_{t_2}}{\longrightarrow} X_2(t_1,\omega_2) \overset{\mathcal{F}_{t_1}}{\longrightarrow} X_{1,2}(\omega_1,\omega_2) \overset{\mathcal{F}^{-1}_{\omega_1}}{\longrightarrow} X_2(t_1,\omega_2) \overset{\mathcal{F}^{-1}_{\omega_2}}{\longrightarrow} x(t_1,t_2)$$

4.5.1 Derivation

So, applying the Fourier Transform to a two-dimensional signal $x(t_1, t_2)$ corresponds to a straightforward two-step process:

Step 1

$$X_1(\omega_1, t_2) = \int_{-\infty}^{+\infty} x(t_1, t_2)\, \mathrm{e}^{-\mathrm{j}\omega_1 t_1} dt_1$$

Step 2

$$\begin{aligned} X_{1,2}(\omega_1, \omega_2) &= \int_{-\infty}^{+\infty} X(\omega_1, t_2)\, \mathrm{e}^{-\mathrm{j}\omega_2 t_2} dt_2 \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x(t_1, t_2)\, \mathrm{e}^{-\mathrm{j}\omega_1 t_1} dt_1\, \mathrm{e}^{-\mathrm{j}\omega_2 t_2} dt_2 \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x(t_1, t_2)\, \mathrm{e}^{-\mathrm{j}(\omega_1 t_1 + \omega_2 t_2)} dt_1 dt_2 \end{aligned} \tag{4.9}$$

Please, note that treating t_1 and t_2 in reverse order, also leads to (4.9). Hence, the order is not important. Also note that for real signals $x(t_1, t_2)$, $X_{1,2}(\omega_1, \omega_2)$ is Hermitian, i.e.

$$X_{1,2}(-\omega_1, -\omega_2) = \overline{X_{1,2}(\omega_1, \omega_2)}$$

Remember: we made the same conclusion for the one-dimensional variants of the Fourier family.

4.5.2 Definition

This leads to the definition of the two-dimensional Fourier Transform:

Two-dimensional Fourier Transform

$$\begin{aligned} x(t_1, t_2) &= \frac{1}{4\pi^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} X_a(\omega_1, \omega_2)\, \mathrm{e}^{\mathrm{j}(\omega_1 t_1 + \omega_2 t_2)} d\omega_1 d\omega_2 \\ X_a(\omega_1, \omega_2) &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x(t_1, t_2)\, \mathrm{e}^{-\mathrm{j}(\omega_1 t_1 + \omega_2 t_2)} dt_1 dt_2 \end{aligned}$$

Analogously, we can define the two-dimensional DFT:

Two-dimensional Discrete Fourier Transform

$$\begin{aligned} x_p[n_1, n_2] &= \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X_p[k_1, k_2]\, \mathrm{e}^{\mathrm{j}2\pi\left(\frac{k_1 n_1}{N_1} + \frac{k_2 n_2}{N_2}\right)} \\ X_p[k_1, k_2] &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_p[n_1, n_2]\, \mathrm{e}^{-\mathrm{j}2\pi\left(\frac{k_1 n_1}{N_1} + \frac{k_2 n_2}{N_1}\right)} \end{aligned} \tag{4.10}$$

So, why would we need multidimensional signals? Clearly, the world as we humans perceive it — besides time — also has three spatial dimensions. Multidimensional signals are hence very common in our world. Typical examples are:

Figure 4.13: Grayscale picture of an artwork in front of a building in Hoboken

- pictures: a 2D projected impression of our 3D world;
- motion pictures: a combination of the above and the dimension time.

4.5.3 Example

As an example, consider Figure 4.13, a picture of the Hoboken campus artwork. Again, let's use Octave/Matlab to help us out. A good exercise to make sure you understand what happens below, is to take your own favorite picture and run through the steps below. Check regularly how the intermediate data looks like. The `size()` function may help you in that.

We will start by converting the original full-color picture to a grayscale representation in order not to *complicate the picture* (no PUN intended) by having to deal with color.

```
colorpicture = imread( "artwork.jpg" );
imshow( colorpicture );
graypicture = rgb2gray( real( colorpicture ) );
```

This creates a matrix with grayscale values from 0 to 255. Next, we'll normalize the picture to have greyscale values in the range $[-0.5, 0.5]$ (to avoid having a very large DC-value).

```
graypicture /= 255;
graypicture -= 0.5;
```

Applying the two-dimensional DFT of (4.10), is extremely simple.

```
graypicture_fft = fft( fft( graypicture.' ).' );
          % or = fft2( graypicture )
```

To generate a 3D plot, we need to generate a mesh first (i.e. a pair of matrices whose element-wise combination corresponds to the coordinates of every point of your DFT matrix).

```
k1 = 1:rows( graypicture_fft );
```

```
k2 = 1:columns( graypicture_fft );
[K1,K2] = meshgrid( k1, k2 );
```

Then, lets create the mesh plot itself:

```
mesh( K1, K2, 20*log10(abs( graypicture_fft(k1,k2) ) ) );
mesh( K1, K2, angle( graypicture_fft(k1,k2) ) );
```

This yields the magnitude and phase spectrum of Figure 4.14 on the next page.

However, our brain is not very suited to interpret these kind of 3D representations. Even more, our brain is even less capable of recognizing patterns in these graphs. We're far better at finding patterns in flat (2D) representations. Therefore, we map the real magnitude and phase spectrum values onto grayscale values, correct the grayscales to map more or less (try different values for K in the code below) to the range $[0, 1]$ and map every Fourier component to a pixel.

```
mag = abs( graypicture_fft ) / K;
imshow( mag );
pha = angle( graypicture_fft ) / (2 * pi);
pha += 0.5;
imshow( pha );
```

This yields the magnitude and phase spectrum pictures of Figure 4.15(a) and (b).

As you can see, recognizing patterns is much easier in this kind of representation.

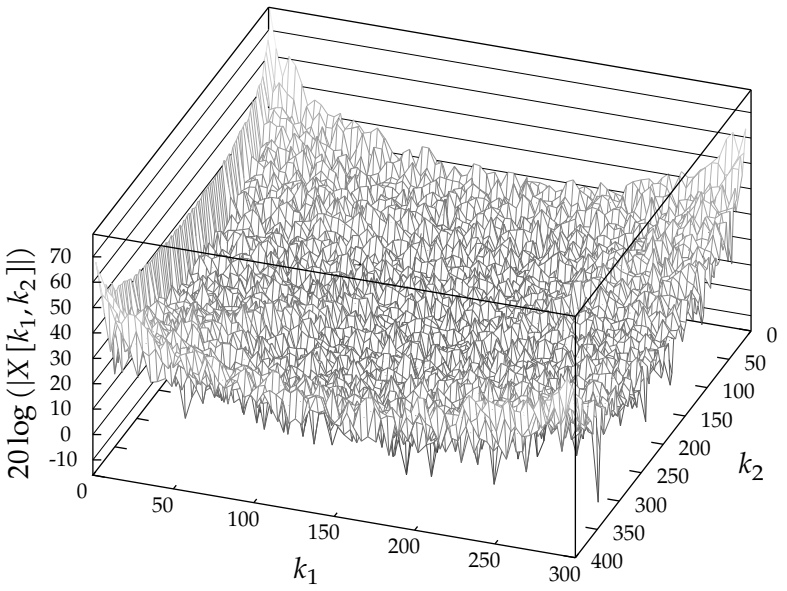
Interpreting such Fourier pictures requires some understanding of the physical interpretation of a Fourier component, very much like we investigated the physical interpretation of the one-dimensional Fourier series in section **??** on page ??.

4.5.4 Physical interpretation of the Fourier components

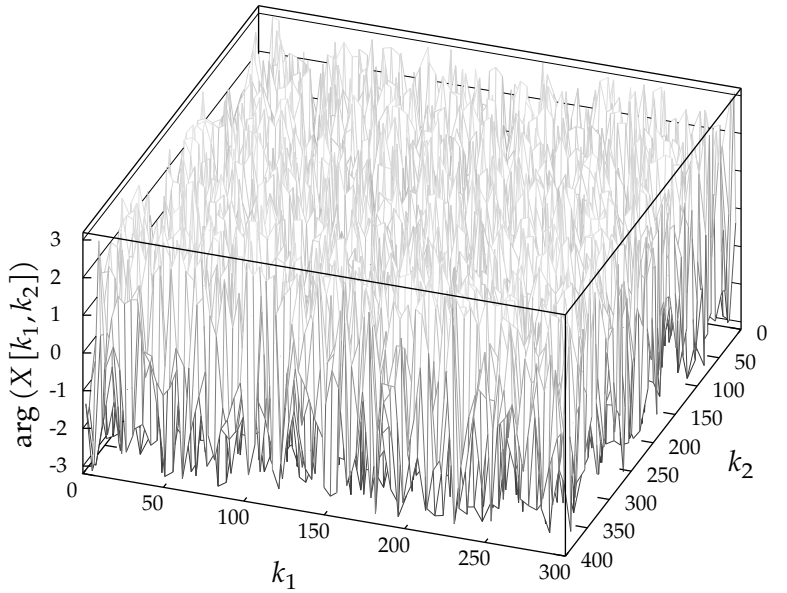
In section 4.5.1 on page 88, we observed that real images yield Hermitian Fourier spectra. Therefore, we need to consider Fourier components in Hermitian pairs to see how they translate to real picture "basis functions".

Instead of making a theoretical analysis (which requires some analytical geometry to fully understand the form of the basis functions), we will show some examples for a 64×64 picture. Take a look at Figure 4.16 on page 93. A grayscale value of -1 has been depicted using a fully black pixel, and a grayscale value of 1 using a fully white pixel.

Example	k_1	k_2	$X[k_1, k_2]$	$X[N-k_1, N-k_2]$
(a)	1	1	$N^2/2$	$N^2/2$
(b)	4	2	$N^2/2$	$N^2/2$
(c)	0	4	$N^2/2$	$N^2/2$
(d)	2	0	$N^2/2$	$N^2/2$
(e)	4	61	$N^2/2$	$N^2/2$
(f)	48	9	$N^2/2$	$N^2/2$
(g)	64	64	$N^2/2$	$N^2/2$

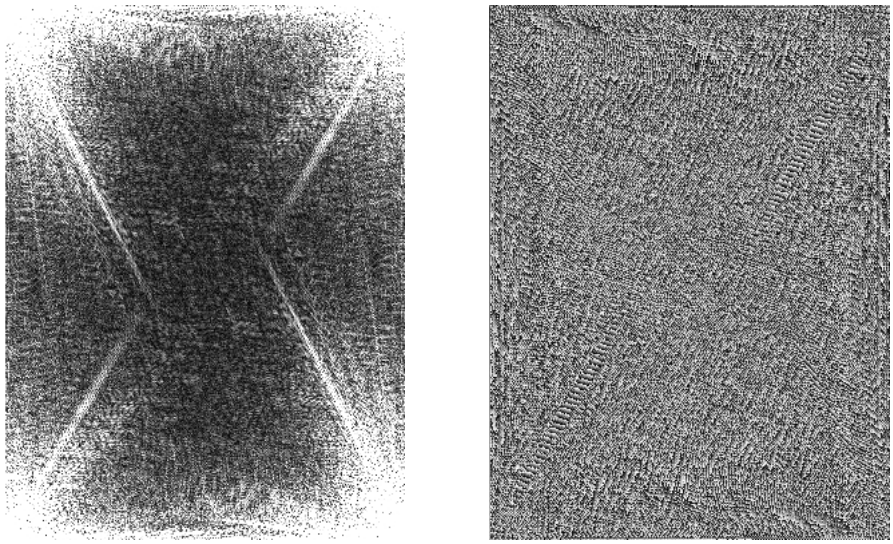


(a)



(b)

Figure 4.14: DFT of the artwork picture of Figure 4.13: (a) magnitude spectrum, and (b) phase spectrum



(a) (b)

Figure 4.15: DFT of the artwork picture of Figure 4.13 displayed as grayscale pictures: (a) magnitude spectrum, and (b) phase spectrum

As can be verified on the example pictures, the value of k_1 corresponds to the number of cycles along the n_1 axis, and the value of k_2 to the number of cycles along the n_2 axis. What also can be seen is that the vector connecting $(0, 0)$ with (k_1, k_2) is always perpendicular to the equi-grayscale lines in the (n_1, n_2)-space.[2]

4.6 Information encoding

4.6.1 Natural information encoding

A physical quantity can be measured using an appropriate sensor. This sensor encodes the measured information onto a particular aspect of a (signal) waveform. This can be the instantaneous signal amplitude, the signal's magnitude spectrum or phase spectrum (or real or imaginary parts of the spectrum), or more involved combinations.

As any signal needs to pass through a system (and as this is a course on digital signal processing, let's assume this is a DSP system), the signal will also be affected by the system. To some extent this is exactly what we want: we want to "improve" the signal. However, a signal in its entirety cannot be improved. As we will see later, we're only able to improve certain aspects of the signal. In most cases, the other aspects of the signal will deteriorate.

Therefore, it is crucial to fully understand which signal aspects carry the information and therefore are crucial to your application.

If the sensor determines the encoding, we label this as so-called *natural information encoding*.

[2]This perpendicularity property is only readily visible if $N_1 \equiv N_2$.

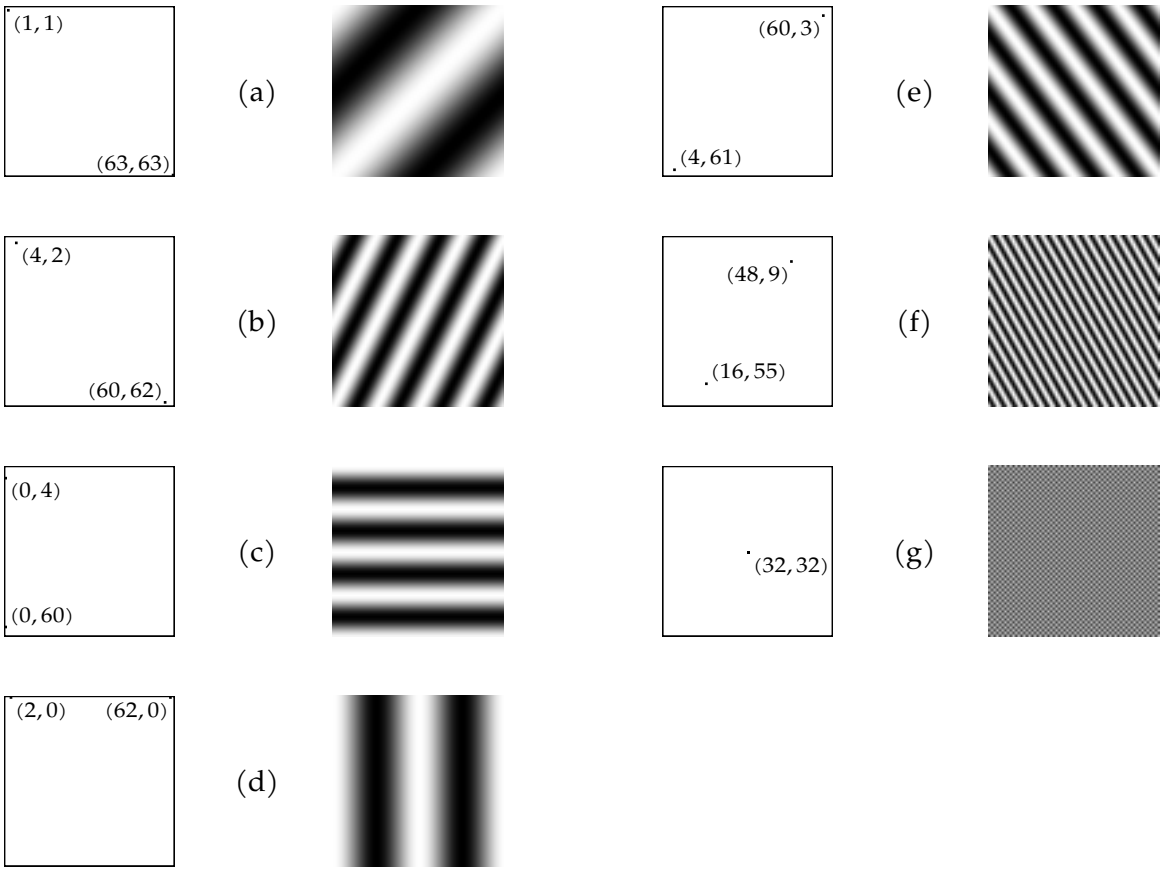


Figure 4.16: Basis functions (right column) corresponding to some examples of Hermitian Fourier pairs (left column)

4.6.2 Engineered information encoding

The information encoding is not always a datum. Good engineers as we are, we're capable of matching the selected encoding to the application. Telecommunications are full of encoding schemes (AM, PM, FM, (D)PSK, TDM, (O)FDM, a.s.o.) to match the communication channel ensuring optimal transmission and reception.

If the engineer determines the encoding, we speak about *engineered information encoding*.

4.6.3 Aspects of human signal perception

To illustrate the importance of knowing which part of the signal carries the most important information, let's take a brief look at human signal perception using two of our senses:

- hearing, and
- vision.

Knowing at least something about human perception is crucial in many regular DSP applications dealing with audio or images. We will not attempt to analyze both systems in depth as one might fill an entire book on this subject alone. However, some examples will shed some light on the matter.

4.6.3.1 Hearing

Consider the two waveforms defined below. Their frequency components lie entirely in the hifi audio range. In fact, their spectra are identical up to some phase difference in the harmonics.

$$x_1(t) = \frac{1}{2}\left(\sin(2\pi 400t) + \frac{1}{2}\sin(2\pi 800t) + \frac{1}{3}\sin(2\pi 1200t) + \frac{1}{4}\sin(2\pi 1600t)\right)$$

$$x_2(t) = \frac{1}{2}\left(\sin(2\pi 400t) + \frac{1}{2}\sin(2\pi(800t - 0.1)) + \frac{1}{3}\sin(2\pi(1200t + 0.1)) + \frac{1}{4}\sin(2\pi(1600t - 0.1))\right)$$

(4.11)

(4.12)

If you take a look at Figure 4.17 on the next page, you will appreciate the significant difference in the time domain. Yet, they sound almost identical.

You can try yourself using the chunk of Octave/Matlab code below:

```
fs = 44100;

function [x] = x1( t )
  tone = 400;
  x = 0.5 * ( 1/1 * sin( 2*pi*tone*t ) \
            + 1/2 * sin( 2*pi*tone*t*2 ) \
            + 1/3 * sin( 2*pi*tone*t*3 ) \
            + 1/4 * sin( 2*pi*tone*t*4 ) );
end
```

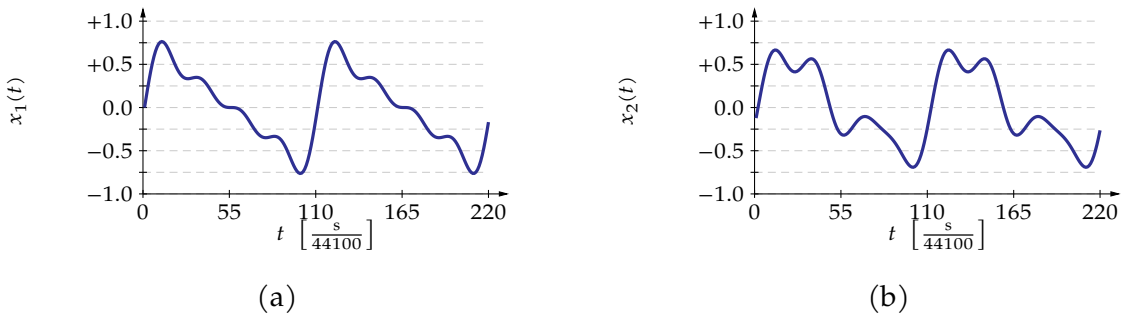


Figure 4.17: Time-domain plots of the signals of (a) equation (4.11), and (b) equation (4.12)

```
10
11 function [x] = x2( t )
12   tone = 400;
13   x = 0.5 * ( 1/1 * sin( 2*pi*tone*t) \
14             + 1/2 * sin( 2*pi*tone*( 2*t-0.1/tone ) ) \
15             + 1/3 * sin( 2*pi*tone*( 3*t+0.1/tone ) ) \
16             + 1/4 * sin( 2*pi*tone*( 4*t-0.1/tone ) ) );
17 end
18
19 t = linspace( 0, 1, fs+1 );
20 sound( x1(t), fs );
```

This phenomenon is generally known as the phase-deafness of the human hearing system. However, this is quite an overstatement as the human hearing system is not completely deaf to the phase. If you listen carefully, you will even hear the difference between the two test-signals above. In the end, phase changes can also rearrange the time sequence of an audio signal if it is non-stationary.

As an example of this, select 20 s of your favorite song and pump it through the following Octave/Matlab script:

```
1  % load your .wav file
2  [x,fs] = auload( "SchubertDieForelle.wav" );
3
4  % calculate the DFT (FFT) of your signal
5  xspectrum = fft( x );
6  mag = abs( xspectrum );
7  pha = angle( xspectrum );
8
9  % randomize the phase of signal with 0%, 1%, 2%, 5%, 10%, 20%, 50%, 100%
10 % assemble the result and check it
11
12 for i = [ 0.0 0.01 0.05 0.1 0.2 0.5 1.0 ]
13
14   printf( "Phase␣randomized␣by␣%g%%:\n", i * 100 );
15
16   printf( "-␣calculating\n" );
17   fflush( stdout );
18
19   % take a copy of the phase and randomize it
20   phamod = pha;
21   phamod -= i * 2*pi * rand( size( x ) );
22
23   % recompose the spectrum using the modified phase
24   xspectrummod = mag .* cos( phamod ) + j * mag .* sin( phamod );
25
26   % recreate the time-domain signal
27   xmod = ifft( xspectrummod );
```

```
28
29    printf( "- playing\n" );
30    fflush( stdout );
31
32    % send the signal to the soundcard
33    sound( xmod, fs );
34
35  end
```

A 1% phase change will be hardly audible, but anything above that is quite noticeable. Going above 50% almost completely averages your song...(a good way to create rich-content sound samples). It must be said that systems that randomize the phase by large amounts (like in the example above) are not very likely to be encountered in practice.

Feel free to experiment with the script above. E.g., you might check for yourself the following things:

- can you hear the influence of a linear phase change?

$$\Delta\phi = k_{LIN}\omega \qquad \text{for different values of } k_{LIN}$$

- can you hear the influence of an exponential phase change?

$$\Delta\phi = k_{LOG}10^{\omega} \qquad \text{for different values of } k_{LOG}$$

4.6.3.2 Vision

Consider the pictures of two arbitrary persons in Figure 4.18.

(a) (b)

Figure 4.18: Grayscale picture of (a) Linus Torvalds, and (b) Bill Gates

An obvious question would be: what's more important about these figures: their magnitude or their phase? A simple experiment shows us the conclusion. We calculated the DFT of the two and combined Linus' magnitude with Bill's phase and vice versa. The result can be seen in Figure 4.19 on the next page.

The conclusion is obvious: the phase is most crucial in image/video applications. The way our mind recognizes pictures is by detecting edges. These edges are created by aligning the edges

(a) (b)

Figure 4.19: Grayscale pictures composed out of: (a) Linus' magnitude and Bill's phase, and (b) Bill's magnitude and Linus' phase

of the sinusoidal basis functions. Therefore phase information is an important factor in creating high-quality images/video.

4.7 Analysis Windows

4.7.1 Analyzing signals using the DFT

In chapter 3, we have learnt how to analyze a signal using the Fourier transforms. One of the conclusions was that in order to make the analysis tractable for a computer, we must discretize time and frequency, leading to the *discrete Fourier transform* (*DFT*). However, there were a few strings attached:

- discretizing time assumes that the signal exhibits a periodic spectrum,
- discretizing frequency assumes that the signal is periodic in time.

We can deal with the former by only considering bandwidth-limited signals and sampling fast enough. Shannon's theorem is the foundation for this.

The latter consequence deserves some more attention. To make things clear, we need to consider three separate cases:

- the signal is periodic
- the signal is nonperiodic and not sufficiently limited in time
- the signal is nonperiodic and sufficiently limited in time

The latter two cases immediatly trigger a question: what's sufficient? We'll get to that in a minute. To make the labeling of the cases a bit simpler, we will call the second case nonperiodic time-limited signals and the latter case nonperiodic time-unlimited signals.

4.7.1.1 Comparing continuous and discrete frequency spectra

Assessing the net effect of the DFT when compared to the Fourier transform, requires comparing continuous frequency spectra with discrete frequency spectra.

In chapter 3 we derived the DFT from the Fourier transform in two steps:

1. we derived the DtFT from the Fourier transform by equating the area covered by the original signal $x_a(t)$ to the area covered by the continuous-time model of our discrete time signal
$$x_m(t) = x[n] \cdot T_s \text{III}_{T_s}(t).$$
However, in the end, we decided to get rid of the factor T_s, resulting in the fact that this factor pops up in the spectrum as:
$$X_m(\omega) = T_s \cdot X_p(\omega) \tag{4.13}$$

2. we derived the DFT from the DtFT by equating the area coverd by the original DtFT spectrum $X_p(\omega)$ to the area covered by the continuous-frequency model of our discrete frequency spectrum
$$X_p = (\omega) = X_p[k] \cdot \omega_f \text{III}_{\omega_f}(\omega) \tag{4.14}$$
ensuring a one-to-one relationship between the time-domain signals:
$$x_p[n] = x_a(nT_s)$$

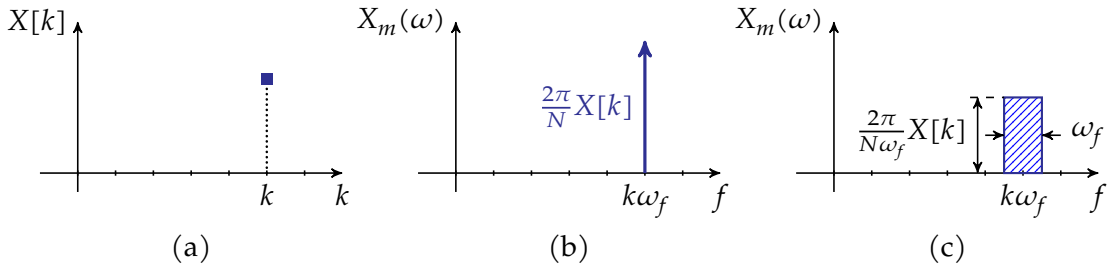
Concatenating (4.13) and (4.14) results in a conversion factor $C = T_s\omega_f = 2\pi/N$.

This has been summarized in Figure 3.11 on page 65. This idea of equating areas helps us in comparing continuous frequency spectra with discrete frequency spectra.

Continuous spectrum containing (infinite) Dirac impulses
As the continuous spectra contain Dirac impulses, we can directly compare the two spectra, just taking into account the conversion factor $C = 2\pi/N$.

Continuous spectrum that is finite
If the original spectrum is finite, then the Dirac impulses in the the model $X_m(f)$ (see above) can be considered to be equal area representations of a rectangle with width ω_f.

For a single discrete frequency component at discrete frequency k, the equivalence we use has been drawn below. From (a) to (b) in case the spectrum contains (infinite Dirac impulses) and further to (c) in case of a finite spectrum. We will refer to the former one as the impulse representation and to the latter one as the staircase representation.



4.7.1.2 Periodic signals

Aligned analysis window

If we want to analyze a periodic signal with period T using the DFT, the best we can do is to align our DFT time-period NT_s to an integral multiple of our signal's period.

The procedure is very simple: according to Shannon's theorem, the bandwidth ω_B dictates an appropriate value for T_s:

$$\frac{2\pi}{T_s} = \omega_s \geq 2\omega_B \quad \Rightarrow \quad T_s \leq \frac{\pi}{\omega_B} \tag{4.15}$$

The signal's period T dictates the number of points N to use:

$$NT_s = kT \quad \Rightarrow \quad N = k\frac{T}{T_s} \tag{4.16}$$

with $k \in \mathbb{Z}_0^+$.
Choosing the combination of T_s and N (or N given T_s) is referred to as selecting the *analysis window*.
Of course we need to select a value for T_s that obeys (4.15) and that makes N integer according to (4.16).

Doing this, the DFT will "see" a correct signal. Indeed, the DFT assumes the signal is periodic with a period NT_s. Therefore, aligning this period with the signal's period will avoid additional artifacts. In fact, what we did is use the DFT to calculate a (bandwidth-limited) Fourier series.

Let's illustrate this with an example. Consider a sinusoidal signal $x(t)$ (see Figure 4.20(a) and (b) for a time-domain and a frequency-domain representation) with a mean value of 0.5 and a frequency f_0 of 1 Hz.

$$\begin{aligned} x_1(t) &= 0.5 + \sin\left(2\pi(f_0 t + 0.05)\right) \\ &= 0.5 + \sin\left(2\pi(t + 0.05)\right) \end{aligned}$$

If we sample this signal with a frequency of 16 Hz during 4 s (i.e. $N = 64$), we have a perfectly aligned analysis window. This has been depicted in Figure 4.20(c). The DFT spectrum in Figure 4.20(d) shows what we expected: The DFT calculates a correct Fourier series equivalent. Also note that the DFT-spectral amplitude of the discrete-time sine corresponds to the Fourier spectral amplitude of the continuous-time sine. Indeed, to convert a DFT spectrum to a continuous-time spectrum, we need to replace every frequency point of our DFT with a Dirac impulse with weight $2\pi/N$ (see section 3.6). The DFT amplitude of the sine at frequency

$f = 1\,\text{Hz}$ equals 32. The corresponding Fourier amplitude can readily be calculated:

$$|X(2\pi 1\,\text{Hz})| = \underbrace{\frac{2\pi}{N}}_{C}|X[4]| = \frac{2\pi}{64}32 = \pi$$

Disaligned analysis window

Now assume that (intentionally or accidentally) we didn't align the analysis window with the signal's period. We can mimick this situation by increasing the frequency a little bit, e.g., to $f_0 = 1.125\,\text{Hz}$.

$$\begin{aligned} x_2(t) &= 0.5 + \sin\left(2\pi(f_0 t + 0.05)\right) \\ &= 0.5 + \sin\left(2\pi(1.125t + 0.05)\right) \end{aligned}$$

You can find the time-domain graph in Figure 4.20(e) and the spectrum in Figure 4.20(f).

We can be sure to encounter some problems. Indeed, the DFT is only able to "see" frequencies that are a multiple of 0.25 Hz. This is a phenomenon we call *picket fencing*. The new frequency does not align to that grid. So maybe our DFT just sees nothing?

The DFT spectrum has been calculated and drawn in Figure 4.20(h). Probably this is an unexpected result to you. What happened, is that the frequency $f = 1.125\,\text{Hz}$ has leaked to the neighbouring frequencies. This is a phenomenon called *spectral leak*. Maybe you didn't notice, but the zero-frequency value has also leaked to the neighbouring frequencies. Also note the fact that the leakage caused the height of the spectral peaks to diminish.

Though spectral leak is inherent to the DFT, an additional aggrevating factor are the discontinuities (jumps) to appear at the analysis window edges.

Conclusion

The conclusion seems obvious: in case of periodic signals: align the analysis window, with a multiple of the signal's period. However, the clue is that we don't know the signal's period! So: alignment is not an option in practice. There are two solutions:

- sample the signal using many periods, such that the energy of the edge-effects of the analysis window disalignment, is small compared to the main frequency-content energy;
- use an analysis window function (we will treat them in section 4.7.4).

4.7.1.3 Nonperiodic time-unlimited signals

Let's consider an aperiodic signal that is not compact (short) enough to fit inside the analysis window.

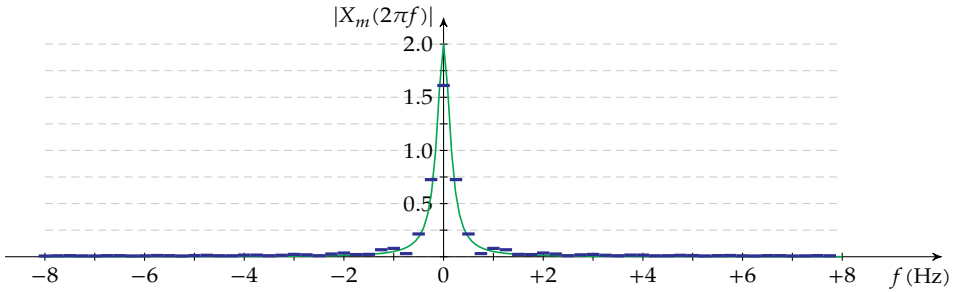
The exponential pulse $x(t)$ (see Figure 4.21(a) and (b) for a time-domain and a frequency-domain representation) is an example of such a signal:

$$x(t) = \mathrm{e}^{-|t-4|}$$

If we sample this signal with a frequency of 16 Hz in the interval $3\,\text{s} \leq t < 7\,\text{s}$ (i.e. $N = 64$), we

get the situation of Figure 4.21(c). We already indicated the time repetition that will be assumed by the DFT.

To compare the discrete spectrum with the continuous spectrum, let's generate the staircase representation, starting from Figure 4.21(d). Using $2\pi/N/\omega_f$ as conversion factor, we obtain the staircase representation (in blue) and the original continuous spectrum (in green).



Clearly they don't match: the original spectrum is monotonously decreasing, while the discrete one oscillates.

The DFT spectrum above shows what we expected: spectral leak. How can we be sure that the effect we see is due to spectral leak? We will investigate this in section 4.7.2.

Again, we can diminish the spectral leak by using a special analysis window (we will treat these in section 4.7.4).

4.7.1.4 Nonperiodic time-limited signals

In this case, let's assume our signal does not extend outside of our analysis window.[3]

Consider as an example a cosine burst signal $x(t)$ (see Figure 4.21(e) and (f) for a time-domain and a frequency-domain representation) with a burst frequency f_0 of 1.125 Hz.

$$x(t) = \begin{cases} 0 & \text{if } t < 3.8889 \\ \cos\left(2\pi \cdot 1.125 \cdot (t-5)\right) & \text{if } 3.8889 \leq t \leq 6.1111 \\ 0 & \text{if } 6.1111 < t \end{cases}$$

If we sample this signal with a frequency of 16 Hz in the interval $3\,\text{s} \leq t < 7\,\text{s}$ (i.e. $N = 64$), we get the situation of Figure 4.21(g). We already indicated the time repetition that will be assumed by the DFT.

To compare the discrete spectrum with the continuous spectrum, let's generate the staircase representation, starting from Figure 4.21(h). Using $2\pi/N/\omega_f$ as conversion factor, we obtain the representation (in blue) and the original continuous spectrum (in green).

[3] A fundamental problem in this case is that all time-limited signals have an unbounded spectrum. Therefore, we cannot comply to Shannon's time-sampling theorem: we will suffer aliasing. We will not investigate this aspect any further.

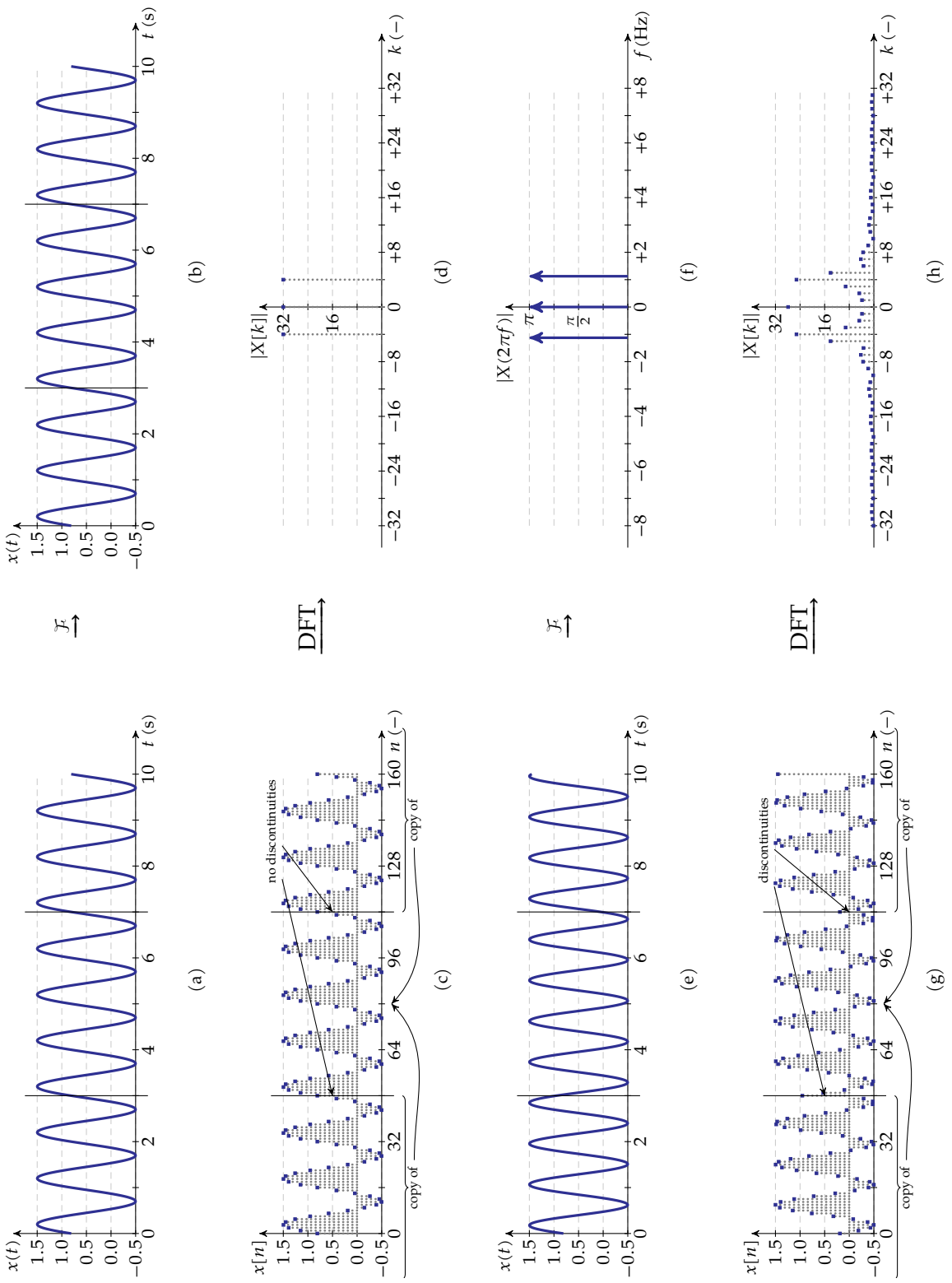


Figure 4.20: Illustration of the effect of aligned and disaligned analysis windows w.r.t. picket fencing and spectral leak: original signal ($f_0 = 1\,\text{Hz}$ — (a) time domain, (b) spectrum; sampled version ($f_0 = 1\,\text{Hz}$) i.c.w. aligned window — (c) time domain, (d) spectrum; modified signal ($f_0 = 1.125\,\text{Hz}$) — (e) time domain, (f) spectrum; sampled version ($f_0 = 1.125\,\text{Hz}$) i.c.w. disaligned window — (g) time domain, (h) spectrum.

.

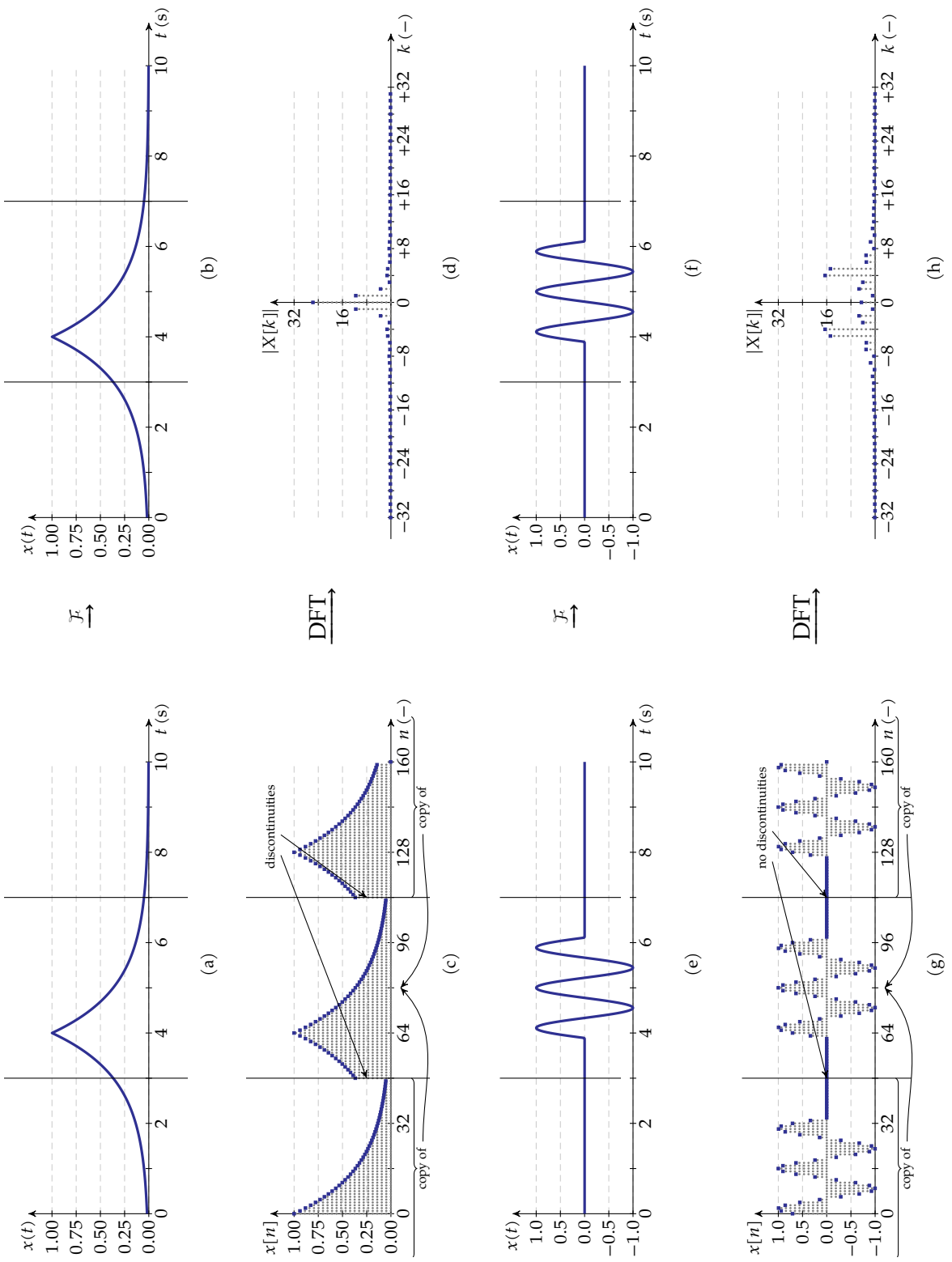
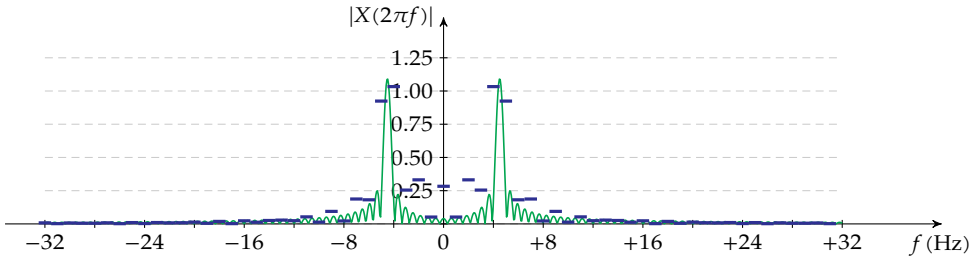


Figure 4.21: Illustration of picket fencing and spectral leak related to using the DFT to analyze a time-unlimited signal — (a and c) time domain, (b and d) spectrum; and a time-limited signal — (e and g) time domain, (f and h) DFT spectrum



Clearly, they don't match: the discrete spectrum contains far larger oscillations that the original spectrum. The DFT spectrum above shows what we expected: aliasing, but also spectral leak. We will investigate this further in section 4.7.2.

The aliasing can be avoided using an appropriate low-pass filter. And, again, we can diminish the spectral leak by using a special analysis window (we will treat these in section 4.7.4).

4.7.2 Spectral leak

We've seen several examples of spectral leak effects on the spectrum obtained by a DFT.

Note that the occurrence of discontinuities at the analysis window's borders is not a sastisfactory explanation for spectral leak. The last example (illustrating nonperiodic time-limited signals) clearly did not suffer from discontinuities, yet the DFT spectrum showed some clear spectral leak. It is time to analyze this phenomenon in depth.

The effect of the DFT in the time-domain is simple: it selects a limited time-window of length N out of the total signal. Mathematically this corresponds to multiplying our discrete-time signal with a rectangular window function of length N:

$$r_N[n] = \begin{cases} 1 & \text{if } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

In the frequency domain, time-multiplication corresponds to convolution. Therefore, let's start by calculating the discrete-time Fourier transform (DtFT) of a general window function:

$$\begin{aligned} R_N(\omega) = \text{DtFT}(r_N[n]) &= \sum_{n=-\infty}^{+\infty} r_N[n]\, \mathrm{e}^{-\mathrm{j}\omega n T_s} \\ &= \sum_{n=0}^{N-1} \mathrm{e}^{-\mathrm{j}\omega n T_s} \\ &\downarrow \quad \text{geometric series with ratio } \mathrm{e}^{-\mathrm{j}\omega T_s} \\ &= \frac{1-\mathrm{e}^{-\mathrm{j}\omega N T_S}}{1-\mathrm{e}^{-\mathrm{j}\omega T_s}} \\ &= \frac{\mathrm{e}^{-\mathrm{j}\frac{\omega N T_S}{2}}}{\mathrm{e}^{-\mathrm{j}\frac{\omega T_S}{2}}} \frac{\mathrm{e}^{\mathrm{j}\frac{\omega N T_s}{2}} - \mathrm{e}^{-\mathrm{j}\frac{\omega N T_s}{2}}}{\mathrm{e}^{\mathrm{j}\frac{\omega T_s}{2}} - \mathrm{e}^{-\mathrm{j}\frac{\omega T_s}{2}}} \\ &= \mathrm{e}^{-\mathrm{j}\frac{\omega(N-1)T_s}{2}} \frac{\sin\left(\frac{\omega N T_s}{2}\right)}{\sin\left(\frac{\omega T_s}{2}\right)} \end{aligned}$$

The exponential part up front is a linear phase part that is due to our rectangular window not being symmetrical around n = 0.[4] The sine fraction is the interesting part. Note the similarity to the Fourier transform of a continuous-time rectangular function.

The 'aliased sinc' spectrum of the rectangular window function can be found in Figure 4.22 on the next page. It is the so-called *Dirichlet kernel*. Do note the aliasing effect embedded in the kernel!

Let's get back to our attempt to try to understand the nature of spectral leak. Now, consider the spectrum of an arbitrary signal to be a Riemann sum of Dirac impulses. By isolating one of these Dirac impulses and convolving that with the Dirichlet kernel, we can see how a single frequency translates itself into a the spectrum of the time-limited signal. Convolving the Dirichlet kernel with a Dirac impulse, simply implies shifting the center of the Dirichlet kernel to the frequency under consideration (see Figure 4.22(b)).

Take a moment to fully understand the essence of this observation: this means that any single frequency component ω_1 in the signal causes a myriad of frequencies to appear in the spectrum according to the shape of the Dirichlet kernel.

Let's take this one step further: the magnitude of the resulting frequencies at a distance $\Delta\omega = \omega - \omega_1$ from the original frequency ω_1 can be written as a frequency dependent 'cross-frequency gain' A_{in}:

$$A_{in}(\omega) = \frac{\sin\left(\frac{(\omega-\omega_1)NT_s}{2}\right)}{\sin\left(\frac{(\omega-\omega_1)T_s}{2}\right)}$$

It should not take you long to understand this, when analyzing Figure 4.22(c).

Now, let's look at it from a different perspective. Consider a particular frequency ω_2. E.g., consider the frequency we used in Figure 4.22(c) as target frequency. This frequency is the target of all frequency components present in the original input signal. The cross-frequency gain seen by this target frequency ω_2 w.r.t. all input frequencies ω is given by:

$$A_{out}(\omega) = \frac{\sin\left(\frac{(\omega_2-\omega)NT_s}{2}\right)}{\sin\left(\frac{(\omega_2-\omega)T_s}{2}\right)}$$

and has been indicated in Figure 4.22(d). The symmetry of the Dirichlet kernel made drawing it very easy. This Dirichlet shape can be considered to be a narrow-band filter that maps the spectrum of the input signal to the frequency under consideration. For this reason, the Dirichlet kernel is also called the *Dirichlet filter function*.

We're getting close to understanding the link between spectral leak and the DFT. The DFT discretizes the frequency axis into a set of uniformly spaced frequencies. In view of the above, any such frequency is influenced by the integral of the entire input signal's spectrum multiplied by the Dirichlet filter function.

Therefore, a DFT grid-frequency is often called a *DFT bin*. It collects (integrates) frequencies according to its Dirichlet filter function.

[4] We assumed our window to start at $n = 0$. The more generic case where it starts at $n = n_0$ can easily be derived using the time-shift property of the DtFT.

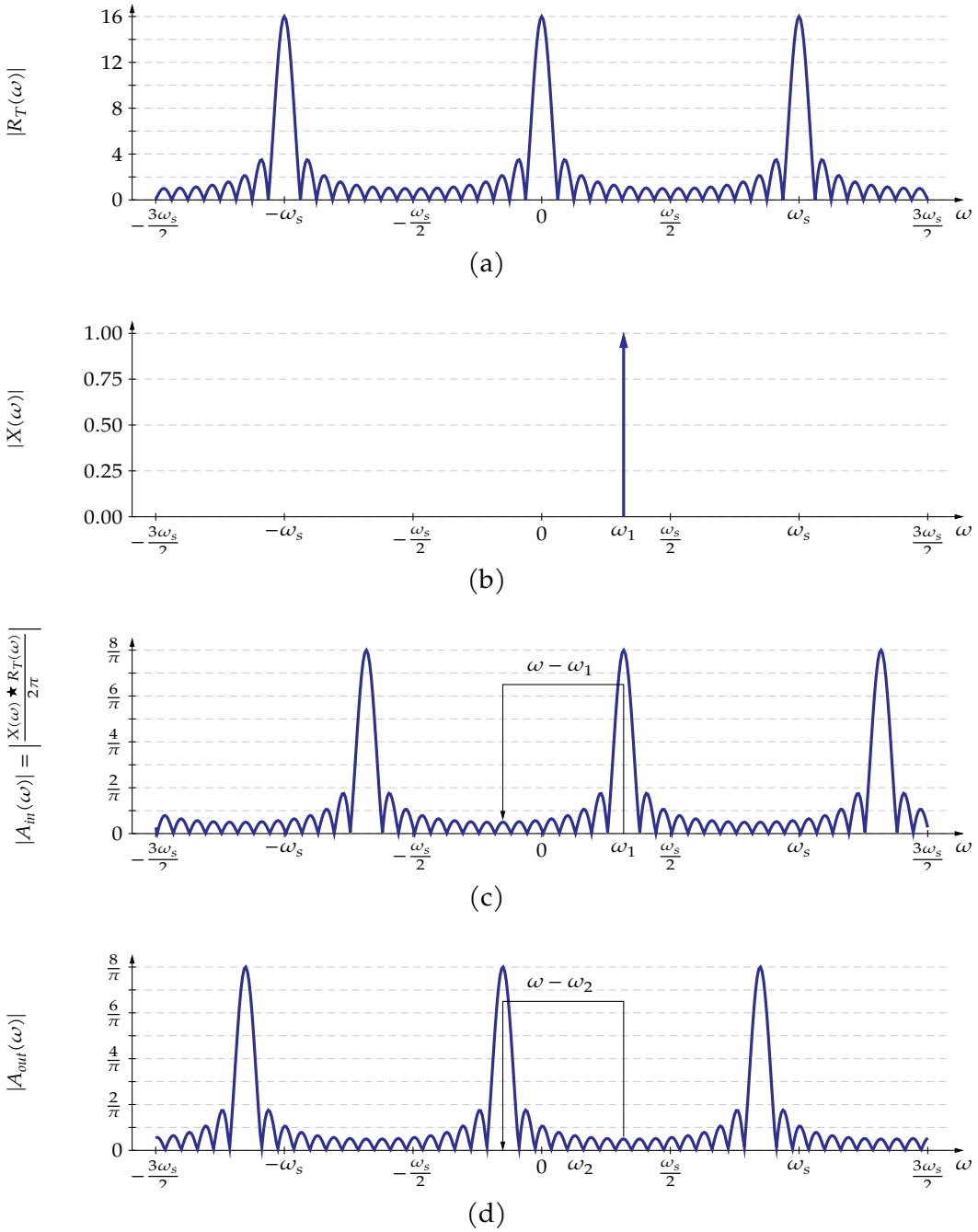
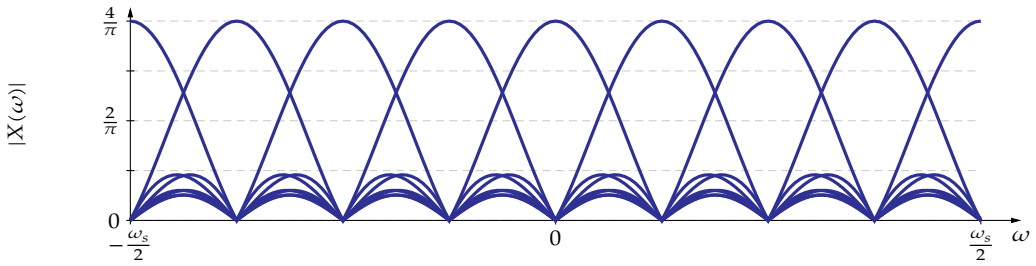


Figure 4.22: Discrete-time Fourier transform of (a) a rectangular window with length $N = 16$ (Dirichlet kernel), (b) a single frequency ω_1 of an arbitrary signal, (c) the resulting spectrum by convolution with the Dirichlet kernel, (d) the cross frequency gain from any frequency to ω_2.

So, how come that input frequencies on the DFT grid, result in no spectral leak? The answer is simple: the zeros of the Dirichlet filter function are exactly located at the neighbouring grid frequencies.

To conclude this section, consider the superposition of all DFT bin filter functions for a DFT of length 8 below:



The shape of this figure illustrates why sometimes one refers to spectral leak as *scalloping loss*.

So, now that we understand spectral leak, is there anything we can do to avoid it? Not fully, but to a certain extent, yes, by apply a non-rectangular window function.

4.7.3 Frequency resolution and dynamic range

The literature on analysis window functions is vast.[5] The problem with window functions, is that there is no "best" window function. Window functions, as we will see in a minute, trade frequency resolution for dynamic range. On that trade-off curve, many solutions exist.

But, before diving into some window examples, let's look at the concepts of *frequency resolution*, *dynamic range* and *noise bandwidth*.

The shape of the Dirichlet kernel is a typical spectral shape, shared by all window functions. It has a main lobe, and several side lobes.

Frequency resolution

The wider the main lobe, the more difficult it will be to distinguish two spectral components that are close to each other. We denote this by saying that a window function with a wide main spectral lobe, has a poor frequency resolution.

Dynamic range

The bigger the side lobes, the more difficult it will be to distinguish a small spectral component on a distance from a large spectral component. Indeed: the main lobe of the small component may be drowned by a side lobe of the large spectral component. We denote this by saying that a window function with large side lobes, has a poor dynamic range. We define the dynamic range as the ratio between the height of the main lobe and the height of the biggest secondary lobe. The dynamic range varies a little bit with the window length. Therefore, we will only give approximate values.

[5] One sure way to end up in the hall of fame of DSP is to invent a new exotic window function and to define a figure of merit that proves it to be the best.

Noise bandwidth

Our DFT will not only sample the signal's spectrum, but also the signal's noise spectrum. A single DFT bin integrates this noise, according to it's bin filter function (i.e. the window's spectral kernel). Therefore, assuming white noise (i.e. noise with a constant amplitude over the entire spectrum) the integral of the kernel $K(\omega)$ is a measure for the amount of noise power that will be picked up by a DFT bin. The total kernel area is often expressed in terms of a noise bandwidth B_{noise}, such that:

$$B_{noise} \cdot \max_{\omega} |K(\omega)|^2 = \int_{\omega=-\omega_s/2}^{\omega_s/2} |K(\omega)|^2 \, \mathrm{d}\omega$$

The higher the noise bandwidth, the less the DFT will be capable of detecting a specific frequency that is drowned in noise. For a rectangular window of length N, we can prove that:

$$B_{noise,rect} = \frac{\omega_s}{N}$$

Note that this is a very logical result: as if every bin integrates its 'own' noise. This is our reference value for the noise bandwidth of all other windows. We will indicate the relative noise bandwidth, w.r.t. the reference value:

$$\beta_{noise} = \frac{B_{noise}}{B_{noise,rect}}$$

The relative noise bandwidth varies also a little bit with the window length, therefore only approximate values will be given.

4.7.4 Analysis window functions

We will limit our description to the following windows:

- Medium to High-resolution windows
  - rectangular (or boxcar) window
  - Hamming window
  - Hann window
  - Blackman window
  - Bartlett window
  - Kaiser window
- Low-resolution, high-dynamic range windows
  - Nuttall window
  - Blackman-Harris window
  - Blackman-Nuttall window
- Very low-resolution windows
  - Flat top window

They are all based on a common idea: diminish the influence of the analysis window edges.

If you combine Octave/Matlab's FFT with the concept of zero-padding, and the Fourier transition diagram of Figure 3.11 on page 65, then creating the graphs below should almost be a no-brainer. Besides, most of the analysis windows presented here are part of Octave/Matlab's *signal processing* toolbox.

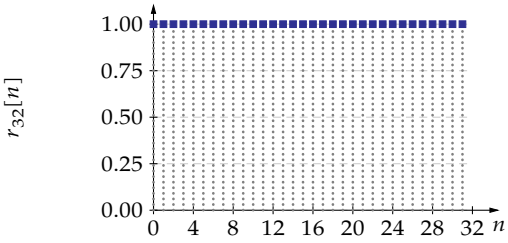
Remarks

- You will notice that all window functions are symmetric. This is not a requirement per se, but it helps in creating window functions with good properties.
- In some cases the preservation of symmetry in the signal you are analyzing is important. In such case you must make sure that the symmetry of the window is well aligned with the symmetry of the signal.
- For illustration's purpose we added some graphs for $N = 32$. Off course any other window length is possible.
- You will notice that, despite the claimed symmetry of our windows, the final value concluding the window does not correspond to the starting value of the window. In many mathematical tools, the symmetry is made full by adding a final value that equals the initial value of the window. MATLAB and OCTAVE do so. Check the mathematical tool (or library) you are using before blindly applying window functions.

4.7.4.1 Rectangular window

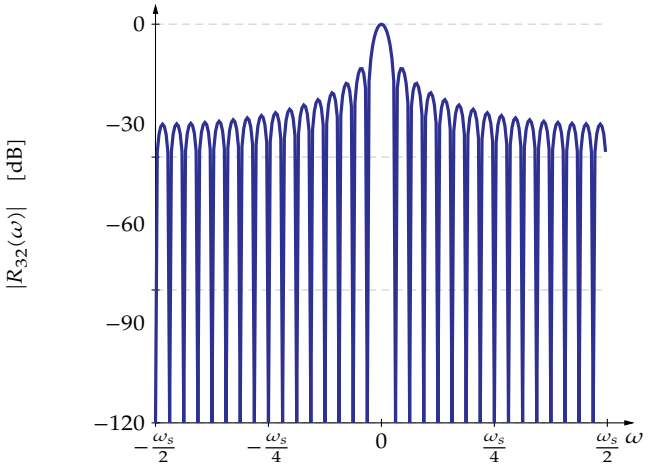
We already studied the rectangular window. For completeness, we incorporate it into our list here. It is also known as the 'box car window'. Think of your childhood drawing capabilities when trying to understand the term 'box car'.

$$r_N[n] = \begin{cases} 1 & \text{if } 0 \le n < N \\ 0 & \text{otherwise} \end{cases}$$



$\beta_{noise} = 1$

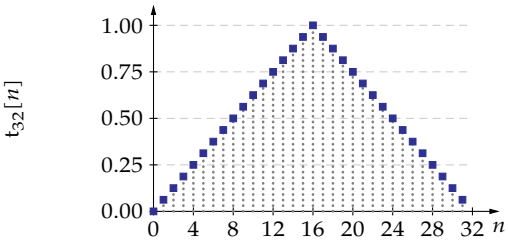
$DR \approx 13.4\,\text{dB}$



4.7.4.2 Bartlett window

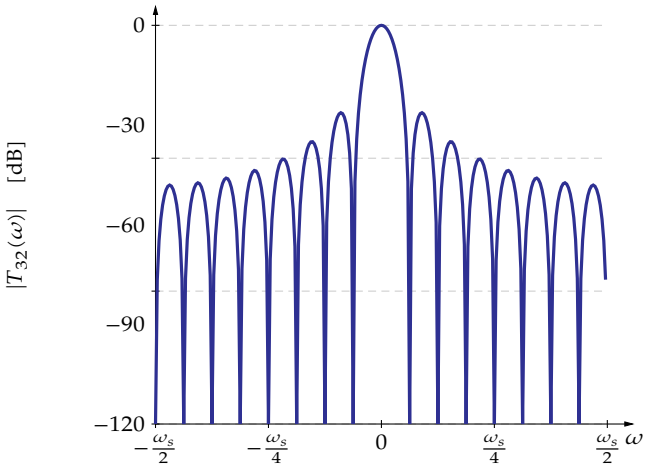
The Bartlett window is a triangular window. For reasons of convenience, we assume N to be even.

$$t_N[n] = \begin{cases} \frac{n}{N/2} & \text{if } 0 \le n \le N/2 \\ 1 - \frac{n-N/2}{N/2} & \text{if } N/2 < n \le N-1 \\ 0 & \text{otherwise} \end{cases}$$



$\beta_{noise} \approx 1.335$

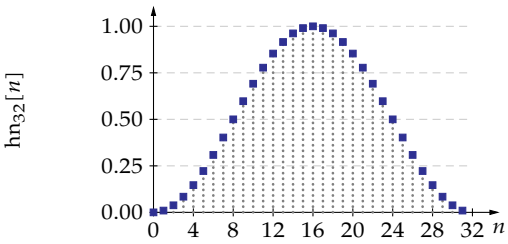
$DR \approx 26.3\,\text{dB}$



4.7.4.3 Hann window

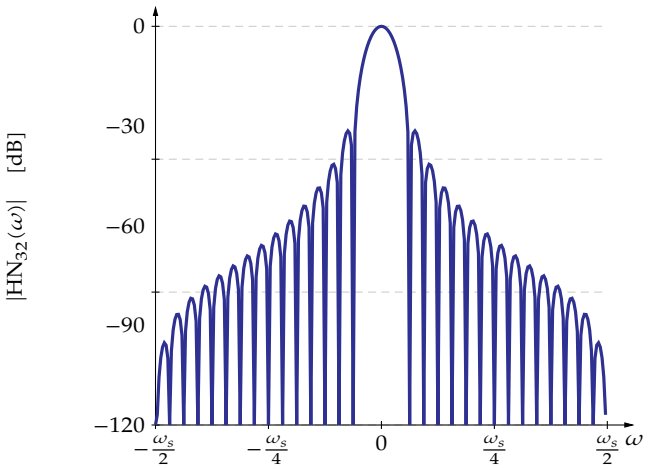
The Hann window (often called 'Hanning' window) is also known as the cosine window.

$$\text{hn}_N[n] = \begin{cases} 0.5 - 0.5\cos\left(\frac{2\pi n}{N}\right) & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$



$\beta_{noise} = 1.5$

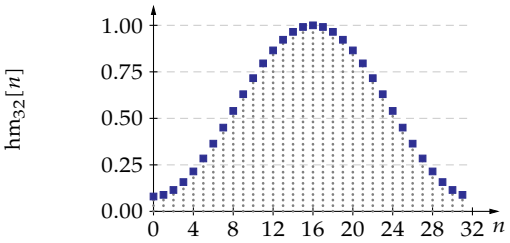
$DR \approx 31.5\,\text{dB}$



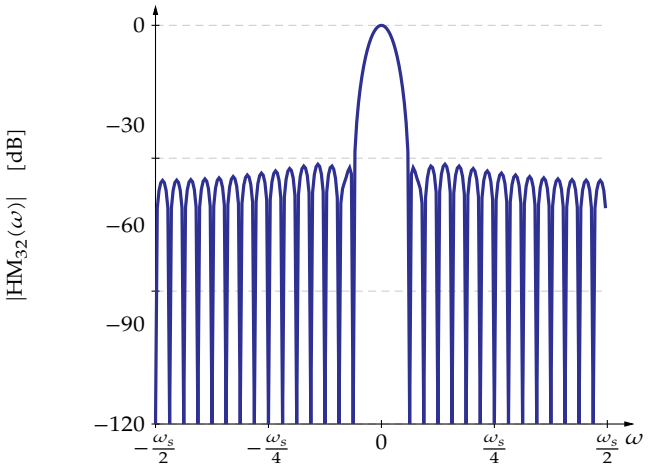
4.7.4.4 Hamming window

The Hamming window is a window of the raised cosine window family.

$$\text{hm}_N[n] = \begin{cases} 0.53836 - 0.46164\cos\left(\frac{2\pi n}{N}\right) & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$



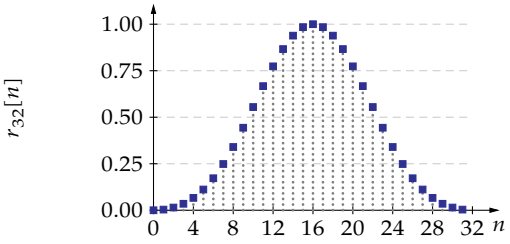
$$\beta_{noise} \approx 1.36$$
$$DR \approx 41.8\,\text{dB}$$



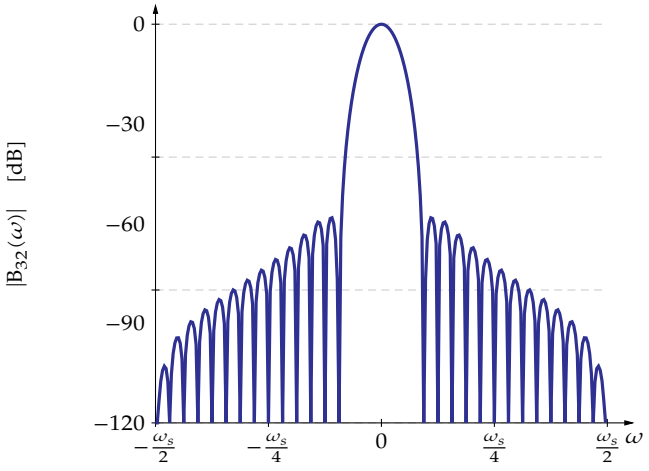
4.7.4.5 Blackman window

The Blackman window is another window of the raised cosine window family.

$$\text{bm}_N[n] = \begin{cases} 0.42 - 0.5\cos\left(\frac{2\pi n}{N}\right) + 0.08\cos\left(\frac{4\pi n}{N}\right) & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$



$$\beta_{noise} \approx 1.727$$
$$DR \approx 58.3\,\text{dB}$$



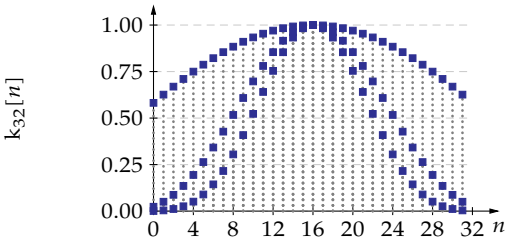
4.7.4.6 Kaiser windows

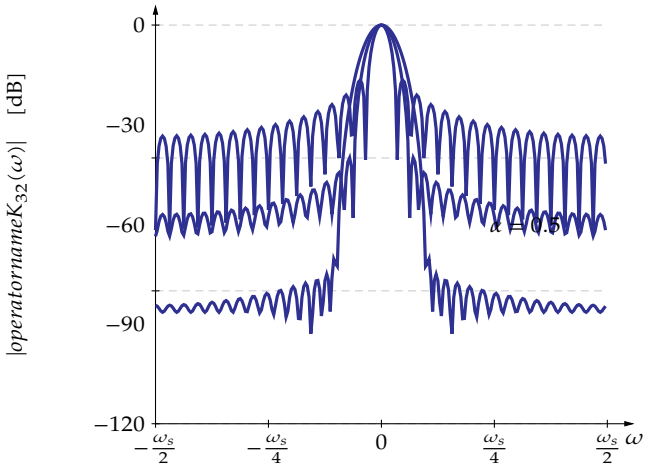
The Kaiser window is an interesting window, for it is controlled by a parameter α that allows generating a myriad of windows on the resolution vs. dynamic range trade-off:

$$\text{k}_N[n] = \begin{cases} \frac{I_0\left(\pi\alpha\sqrt{1-\left(\frac{2n}{N}-1\right)^2}\right)}{I_0(\pi\alpha)} & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

with I_0 the zeroth-order modified Bessel function of the first kind:

$$I_0(x) = 1 + \sum_{k=1}^{+\infty} \frac{x^{2k}}{2^{2k}(k!)^2}$$





$$\beta_{noise} \approx \begin{cases} 1.022 & \text{if } \alpha = 0.5 \\ 1.414 & \text{if } \alpha = 1.75 \\ 1.795 & \text{if } \alpha = 3 \end{cases}$$

$$DR \approx \begin{cases} 16.6\,\text{dB} & \text{if } \alpha = 0.5 \\ 40.0\,\text{dB} & \text{if } \alpha = 1.75 \\ 70.4\,\text{dB} & \text{if } \alpha = 3 \end{cases}$$

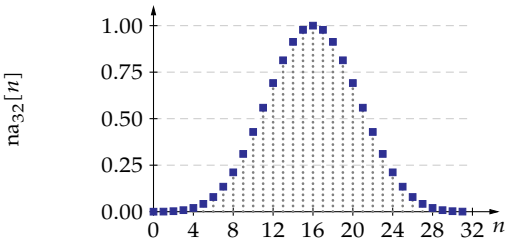
4.7.4.7 Nuttall window

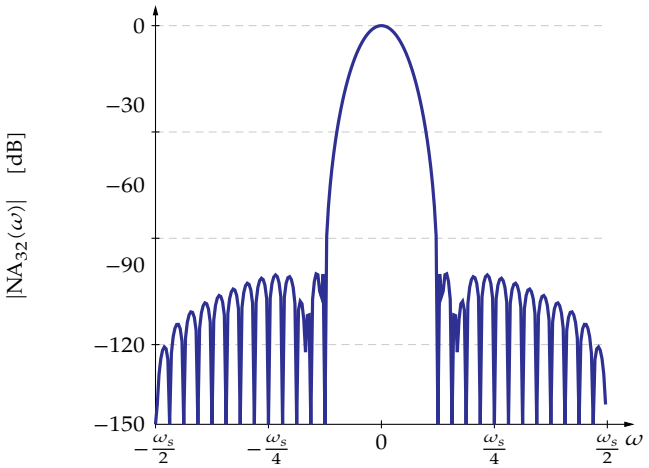
The Nuttall window is a four-term cosine window with high dynamic range.

$$\text{na}_N[n] = \begin{cases} a_0 - a_1 \cos\left(\frac{2\pi n}{N}\right) + a_2 \cos\left(\frac{4\pi n}{N}\right) - a_3 \cos\left(\frac{6\pi n}{N}\right) & \text{if } 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases}$$

with

$$a_0 = 0.355768 \qquad a_1 = 0.487396 \qquad a_2 = 0.144232 \qquad a_3 = 0.012604$$





$$\beta_{noise} \approx 2.02$$
$$DR \approx 93.4\,\text{dB}$$

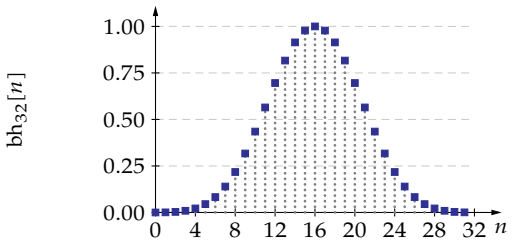
4.7.4.8 Blackman-Harris window

The Blackman-Harris window is a four-term cosine window with high dynamic range.

$$\mathrm{bh}_N[n] = \begin{cases} a_0 - a_1 \cos\left(\frac{2\pi n}{N}\right) + a_2 \cos\left(\frac{4\pi n}{N}\right) - a_3 \cos\left(\frac{6\pi n}{N}\right) & \text{if } 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases}$$

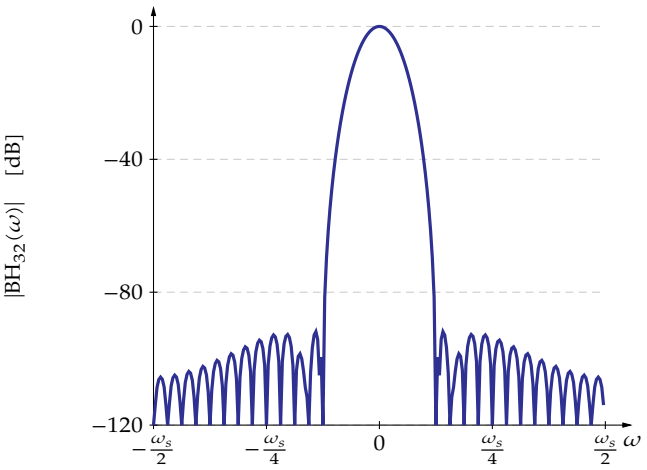
with

$$a_0 = 0.35875 \qquad a_1 = 0.48829 \qquad a_2 = 0.14128 \qquad a_3 = 0.01168$$



$\beta_{noise} \approx 2.00$

$DR \approx 91.9\,\mathrm{dB}$



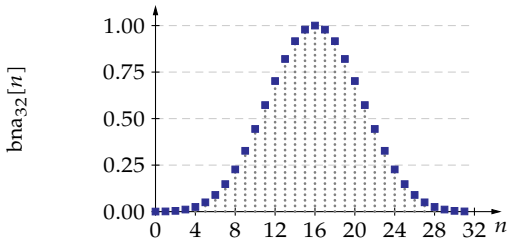
4.7.4.9 Blackman-Nuttall window

The Blackman-Nuttall window is a four-term cosine window with high dynamic range.

$$\mathrm{bna}_N[n] = \begin{cases} a_0 - a_1 \cos\left(\frac{2\pi n}{N}\right) + a_2 \cos\left(\frac{4\pi n}{N}\right) - a_3 \cos\left(\frac{6\pi n}{N}\right) & \text{if } 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases}$$

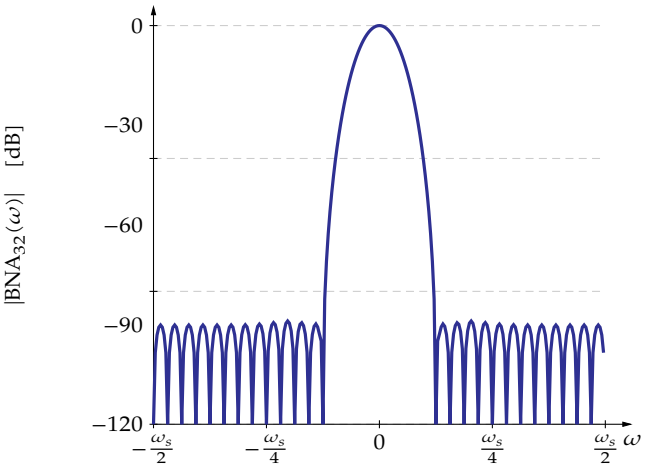
with

$$a_0 = 0.3635819 \qquad a_1 = 0.4891775 \qquad a_2 = 0.1365995 \qquad a_3 = 0.0106411$$



$\beta_{noise} \approx 1.976$

$DR \approx 88.9\,\mathrm{dB}$



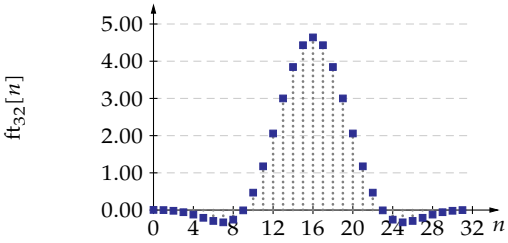
4.7.4.10 Flat top window

The flat top window is a five-term cosine window with the beneficial property of having an almost perfectly flat primary lobe response over the main DFT bin. This property can be important if one wants to know the amplitude of a single sinusoidal signal with high precision without being bothered by scalloping loss. By consequence, the window also has a very low resolution, and will be greatly affected by noise.

$$\text{ft}_N[n] = \begin{cases} a_0 - a_1 \cos\left(\frac{2\pi n}{N}\right) + a_2 \cos\left(\frac{4\pi n}{N}\right) - a_3 \cos\left(\frac{6\pi n}{N}\right) + a_4 \cos\left(\frac{8\pi n}{N}\right) & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

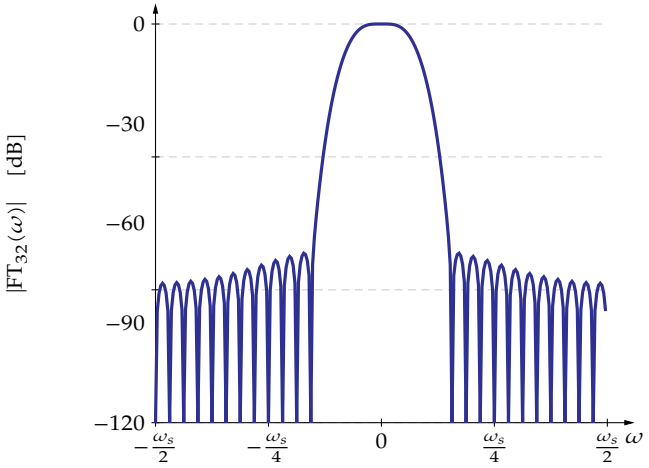
with

$$a_0 = 1 \qquad a_1 = 1.93 \qquad a_2 = 1.29 \qquad a_3 = 0.388 \qquad a_4 = 0.032$$



$\beta_{noise} \approx 3.77$

$DR \approx 69.0\,\text{dB}$



Exercises

Some exercises on window functions

Exercise 4.7.4.10-1: Apply a Blackman window (with $N = 10$) to the following sample sequence. To allow checking your result, the values have been chosen such that the result forms a linear sequence.

$$x[n] = [0, 24.86767, 9.96164, 5.88481, 4.71015, 5.00000, 7.06522, 13.73122, 39.84656, 223.80899]$$

Exercise 4.7.4.10-2: The following sequence is symmetrical (in time). Choose an appropriate Hann window to multiply the signal with, without destroying its symmetry.

$$x[n] = [0, 1, 2, 3, 4, 5, 4, 3, 2, 1, 0]$$

Exercise 4.7.4.10-3: The following sequence is symmetrical (in time). Choose an appropriate Nuttall window to multiply the signal with, without destroying its symmetry.

$$x[n] = [1, 2, 3, 4, 4, 3, 2, 1]$$

Exercise 4.7.4.10-4: (*) Find another analysis window in literature (or define your own!), and calculate its relative noise bandwidth and its dynamic range. Did you find a high-resolution or a high-dynamic range window? In case you invented your own window, is it worth attaching your name to it?

4.8 Power spectral density estimation

We treated one of the interesting characteristics of a signal, its power spectral density, in (Chapter 3) on the Fourier Transformation. At that time, we did not fully detail the relationship between the *power spectral density function* (PSD) and the *power spectral mass function* (PSM) of its sampled version. Yet, there is a clear relationship. Let's investigate this.

Assume we are exploring a signal $x(t)$ that is time limited with support $t \in [0, T]$. The average power of that signal on the interval $[0, T]$ can be calculated to be:

$$P_x = \frac{1}{T} \int_0^T |x(t)|^2 \, \mathrm{d}t \tag{4.17}$$

This average power can be written as an integral of the power spectral density function[6]:

$$P_x = \int_{-\infty}^{+\infty} \mathrm{PSD}(\omega) \, \mathrm{d}\omega \tag{4.18}$$

with

$$\mathrm{PSD}(\omega) = \frac{|X(\omega)|^2}{2\pi T}$$

Let's assume that we don't have the function $x(t)$ available, but only a sampled version $x[n]$ of it with sampling period T_s, such that

$$x[n] = x(nT_s)$$

with $n = 0, 1, \ldots, N-1$ and $T = NT_S$.

Using this sampled version, we can calculate an estimate of the average power using a piecewise rectangular approximation of (4.17) and transform this calculation subsequently to the frequency domain:

$$\begin{aligned}
P_x &\approx \frac{1}{T} \sum_{n=0}^{N-1} |x[n]|^2 \, T_s \\
&\downarrow \quad \text{Parseval's theorem: } \sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 \\
&= \frac{T_s}{T} \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 \\
&= \frac{1}{N^2} \sum_{k=0}^{N-1} |X[k]|^2 \\
&= \sum_{k=0}^{N-1} \mathrm{PSM}[k]
\end{aligned} \tag{4.19}$$

[6] If in doubt, start your reasoning from Parseval's theorem again.

with

$$\mathrm{PSM}[k] = \frac{|X[k]|^2}{N^2}$$

The sum of (4.19) could be seen as a piecewise rectangular approximation of (4.18):

$$P_x = \sum_{k=0}^{N-1} \frac{\mathrm{PSM}[k]}{\omega_f} \omega_f$$

with ω_f the frequency pitch of the DFT.

In view of this, we can propose the following approximation for the power spectral density function of $x(t)$:

$$\mathrm{PSD}(\omega) \approx \frac{\mathrm{PSM}\left[\left\lfloor \frac{\omega}{\omega_f} \right\rfloor\right]}{\omega_f}$$

We therefore call $\mathrm{PSM}[k]/\omega_f$ an *estimator* for the PSD.

Given the obvious relationship $\omega_f = \omega_s/N$ and $\mathrm{ESM}[k] = N\,\mathrm{PSM}[k]$ the following alternative is also often encountered:

$$\mathrm{PSD}(\omega) \approx \frac{\mathrm{ESM}\left[\left\lfloor \frac{\omega}{\omega_f} \right\rfloor\right]}{\omega_s} = \frac{1}{\omega_s} \frac{\left|X\left[\left\lfloor \frac{\omega}{\omega_f} \right\rfloor\right]\right|^2}{N} \tag{4.20}$$

In case normalized frequencies are used, this reduces to:

$$\mathrm{PSD}(\omega) \approx \frac{1}{2\pi} \frac{\left|X\left[\left\lfloor \frac{\omega}{\omega_f} \right\rfloor\right]\right|^2}{N} \tag{4.21}$$

4.8.1 Improving the estimate

One problem remains: if the signal one wants to observe is long, the number of sample points N can be quite large, and the straight estimation of the PSD according to (4.21) can be computationally very expensive (proportional to N^2 in case of the regular DFT and proportional to $N \log_2 N$ in case of the superior FFT).

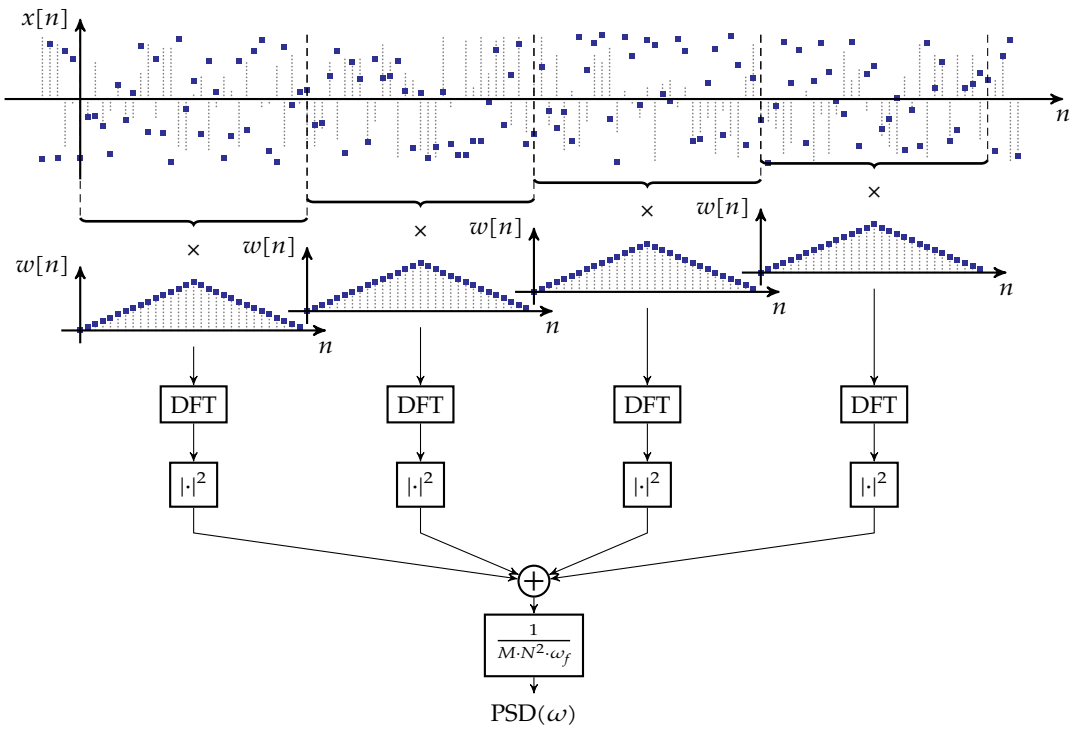
In case the signal is stationary (i.e. the signal does not change it's overall spectral density over time, we can solve this by only considering a particular subframe of the samples, as long as we take enough samples.[7]

This offers a new opportunity of taking multiple subframes and average, such that the influence of noise diminishes. In that way we are effectively trading frequency resolution for accuracy.

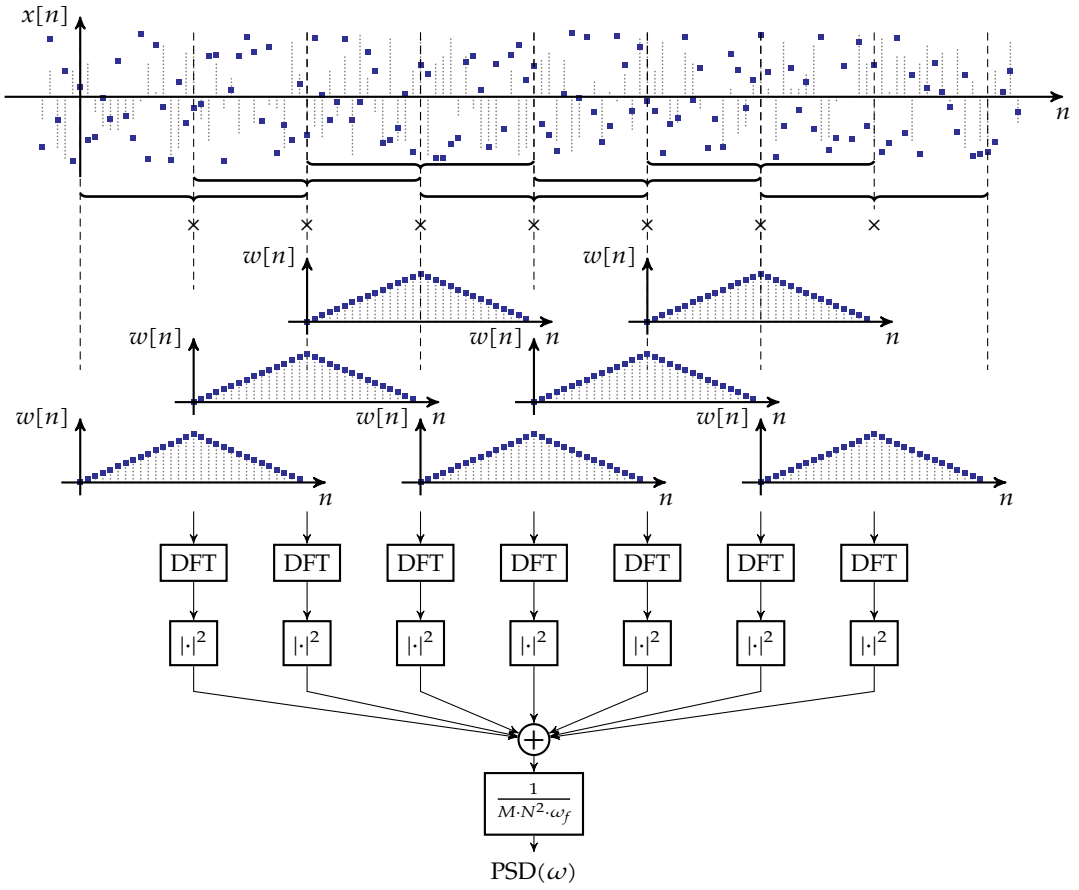
This leads to two common methods:

- taking non-overlapping subframes: Bartlett's method
- taking overlapping subframes: Welch's method

[7] This description of stationary signal intends to give you an intuitive feeling of what 'stationary' means. It is in no way a rigorous mathematical definition. In addition, we don't specify what 'enough samples' means. This would take us too far.



(a) PSD estimation using Bartlett's method, making use of a Bartlett window ($M = 4$)



(b) PSD estimation using Welch's method, making use of a Bartlett window (50% overlap, $M = 7$)

Figure 4.23: Illustration of PSD estimation with noise suppression by incoherent averaging, using M frames, a window length $N = 30$ and normalized frequencies

4.8.2 PSD estimation using Bartlett's method

Bartlett was the first to propose averaging multiple subframes in estimating the PSD. As cutting subframes means multiplication with a rectangular window, Dirichlet patterns are to be expected and therefore applying a proper window function to each of the subframes makes sense. Of course, a Bartlett window is a good candidate, but another window is equally good. The principle has been illustrated in Figure 4.23a.

4.8.3 PSD estimation using Welch's method

Welch observed that applying a window function to the data prior to calculating the DFT, underuses the information of the samples near the window edges. Therefore, he proposed to let the frames overlap. The overlap amount can be freely chosen, as can be the window function used to shape the frames prior to calculating the DFT. The principle has been illustrated in Figure 4.23b.

Remarks

- Note that the averaging spectra is only beneficial in the case of power spectra, not in the case of ordinary spectra. The former is sometimes called incoherent averaging, the latter coherent averaging. Averaging ordinary spectra will result not only in attenuation of the noise, but also in attenuation of the signal, because of destructive interference. This is to be avoided. The destructive interference will not occur with power spectra, because only positive numbers are added in the calculation of the average.
- Both Bartlett's method and Welch's method have been implemented in MATLAB/OCTAVE and are available as the function `pwelch`. Explore the help facililty for more information.

4.9 Gain of the DFT

The DFT has a very beneficial property: it has the ability of recovering very low amplitude signals that are buried in noise. We call this ability: the *gain* of the DFT.[8]

4.9.1 Theory

The effect of *processing gain* allows the DFT to dig up a small signal out of a noise wasteland. How is this possible? Consider a sinusoidal signal contaminated by white noise with a power spectral density of $\eta \text{V/Hz}^{0.5}$:

$$x_n(t) = \underbrace{A\sin(\omega_0 t)}_{x(t)} + n_{white,\eta}(t)$$

Assume we sampled the noisy signal with a sampling frequency ω_s and are calculating an N point DFT (after applying an appropriate analysis window function).

[8] Sometimes this is also referred to as the *processing gain* of the DFT.

As we've seen in section 4.7.2, a single DFT bin behaves like a small-band filter with an approximate bandwidth $\beta_{noise}\frac{\omega_s}{N}$.

The key thing to understand is that:

- the sine wave has all its energy concentrated on two frequencies (represented by two Dirac impulses), while
- noise is spread over the entire primary frequency period.

Therefore, increasing N means that the DFT bin containing one of the signal's main frequencies, still sees the full strength (or in this case, the full weakness) of the sine wave, while the noise energy it picks up is limited by the decreased bandwidth of the DFT bin.

The RMS value of a sine wave with amplitude A amounts to:

$$x_{\text{RMS}} = \frac{A}{\sqrt{2}}$$

The total corresponding power amounts to the square of that value:

$$P = x_{\text{RMS}}^2 = \frac{A^2}{2}$$

Each of the two frequency components (for $+\omega_0$ and $-\omega_0$) carries half of that power, i.e. $A^2/4$. Therefore, the RMS value of the two frequency components of the sinewave separately, equals:

$$x_{+\omega_0,\text{RMS}} = x_{-\omega_0,\text{RMS}} = \frac{A}{2}$$

The DFT bin around $+\omega_0$ only sees one of the two components while the RMS value of the noiseband seen by the DFT bin containing that frequency equals:

$$n_{\text{RMS}} = \sqrt{\beta_{noise}\frac{\omega_s}{N}\eta^2}.$$

The ratio of the two gives us the *signal-to-noise ratio* as seen by a single DFT bin:

$$\text{S/N}_{DFT} = \frac{\frac{A}{2}}{\sqrt{\beta_{noise}\frac{\omega_s}{N}\eta^2}} = \sqrt{N}C$$

with

$$C = \frac{A}{2\eta\sqrt{\beta_{noise}\omega_S}}.$$

We can safely assume that β_{noise} is almost independent of N, and therefore that C is a constant. Conclusion: we can increase the signal-to-noise ratio of the DFT by increasing N. This is the so-called *processing gain* of the DFT.

Increasing the number of DFT points (by sampling faster in the same time window, or by considering a longer time window) with a factor of k corresponds to a gain equaling:

$$G_{\text{DFT}} = \frac{\sqrt{kN}C}{\sqrt{N}C} = \sqrt{k}.$$

On a logarithmic decibel scale, this corresponds to:

$$G_{\mathrm{DFT,dB}} = 10 \log(k)$$

i.e. a 3 dB increase whenever N is doubled ($k = 2$).

Let's illustrate this with an example.

4.9.2 Example

Consider a sinusoidal signal buried in noise:

$$x(t) = \sin(51.2\pi t) + n(t) \qquad (V) \tag{4.22}$$

with $n(t)$ a Gaussian white-noise source with standard deviation of $\sqrt{2}$ (see Figure 4.24).

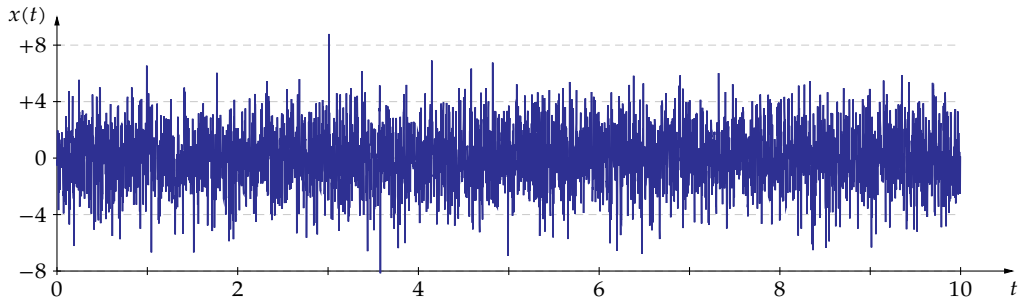
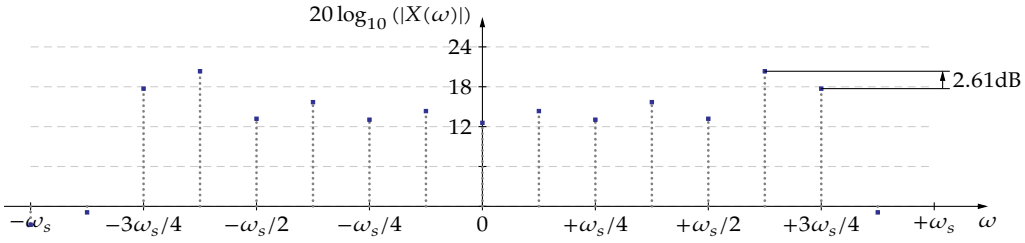


Figure 4.24: The noisy signal of equation (4.22)

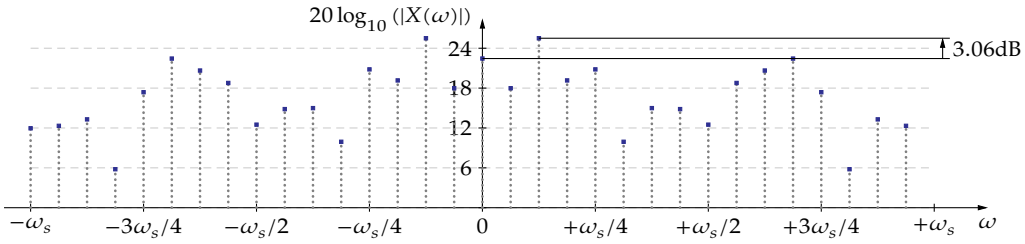
Using a sampling frequency $f_s = 409.7$ Hz and sampling this signal with $N = 16$ (see Figure 4.25(a)) yields a DFT-spectrum that does not quite reveal a sinusoidal component. Doubling the number of points considered in the DFT makes the signal emerge from the noise (see Figure 4.25(b-e)).

Please, observe that whenever we double the number of points, the increase in signal-to-noise ratio is not *exactly* 3 dB (as expected). At first, this is due to the fact that the signal is still drowning in the noise. For increased values of N, this is due to the fact that the noise we applied is not perfectly white (selecting N noise samples out of an infinite number of white-noise samples, make the selection non-white). This behavior can also be observed in the graph of Figure 4.26 on page 122.

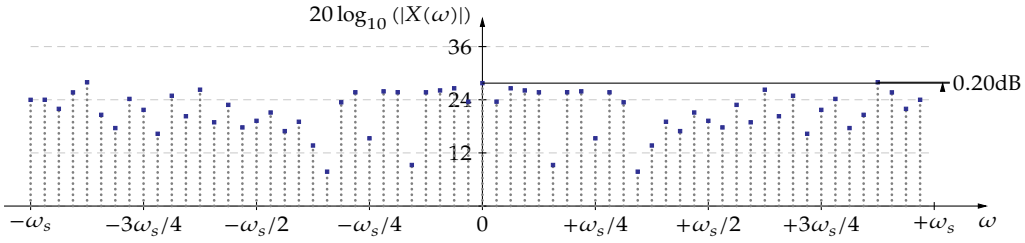
This causes both the detected signal and the noise floor to be dependent on this imperfectness.



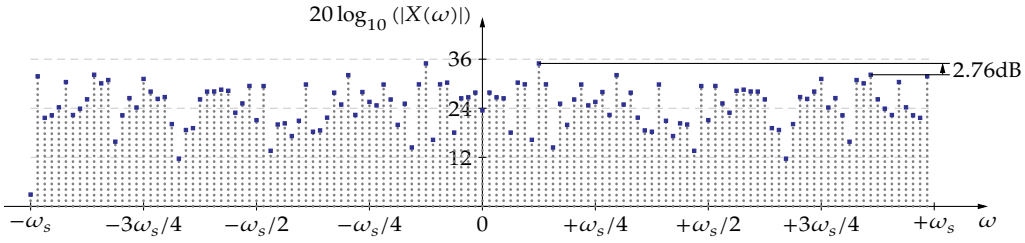
(a)



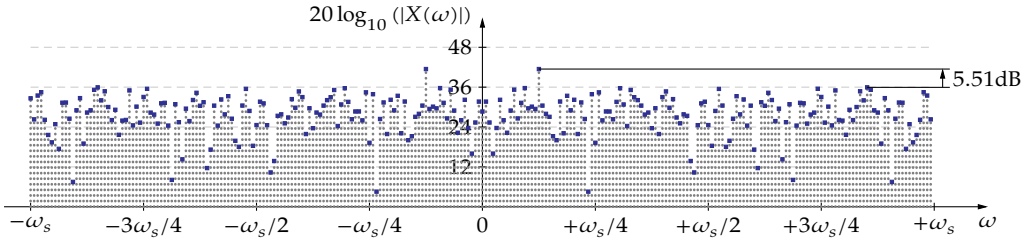
(b)



(c)



(d)



(e)

Figure 4.25: DFTs of the noisy signal of Figure 4.24 ($N = 16 \ldots 256$)

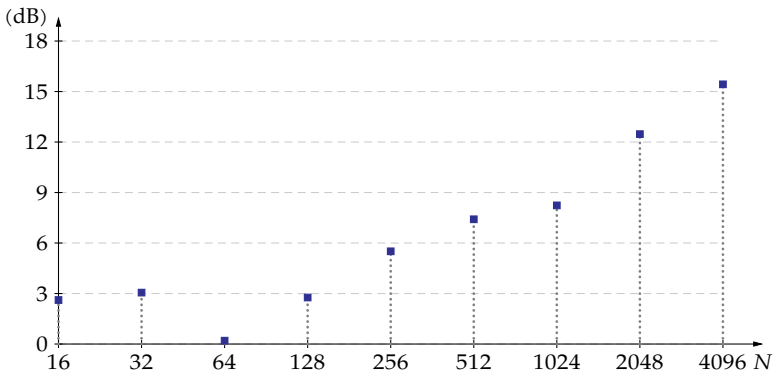


Figure 4.26: S/N_{DFT} of the DFT of the noisy signal of Figure 4.24 as a function of N.

Chapter 5

Sampling, Quantization and Reconstruction

In this chapter, you will learn:

- how to convert an analog signal into a digital signal,
- how to be smarter than (or at least as smart as) Shannon in doing so,
- how quantization affects your signal,
- how to convert it back to an analog signal,
- what nonidealities affect AD and DA converters.

After having read this chapter, some questions will still be left unanswered:

- what can we do with these digitized signals?
- how do these DACs and ADCs work?

After having read/studied this chapter, you are expected to be able to

- compose an appropriate conversion setup (selecting sampling frequency, appropriate filters, ...), and
- read and understand the basic information of a converter datasheet.

5.1 Introduction

In chapter 3, traversing the boundary of continuous to discrete time was merely a mathematical operation:

- sampling: multiplication of the time-domain signal with a Dirac comb,
- reconstruction: low-pass filtering the frequency-domain signal.

We totally neglected quantization.

It's time to think about how we could achieve these operations using real-world electronics. There's one big problem with directly casting the mathematical approach outlined above into electronics: Dirac impulses are difficult to generate.

Therefore, the basic scheme is somewhat more complicated and is depicted in Figure 5.1. Let's start on the left-hand side of the figure. The analog signal $v(t)$ is filtered to prepare it for

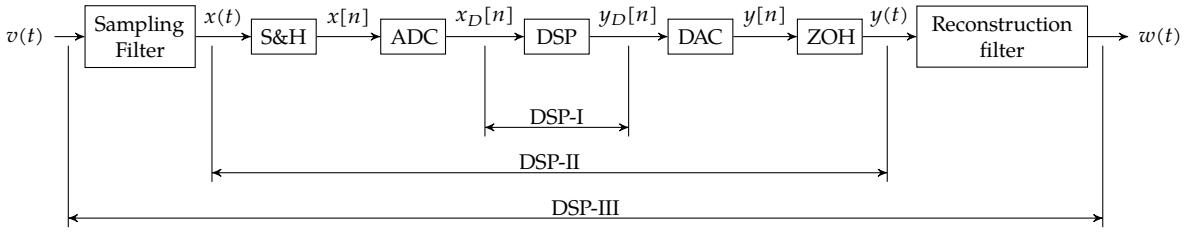


Figure 5.1: Overall block diagram of a DSP system from the viewpoint of analog to digital conversion and digital to analog conversion. The DSP processor may be merely a numerical processor (DSP-I) or also having the ADC and DAC on chip (DSP-II), or even be equipped with sampling and reconstruction filters (DSP-III).

sampling, resulting in $x(t)$. Then, $x(t)$ is sampled by a *sample and hold* block into $x[n]$, such that we have a full sampling period the time to digitize the signal. This is done by the *analog to digital converter* (*ADC*). It converts the discrete-time signal into a true (quantized) digital signal $x_D[n]$ that our *digital signal processor* (*DSP*) can process. After processing, the digital signal $y_D[n]$ is converted to an analog quantity $y[n]$ by the *digital to analog converter* (*DAC*), kept in between the sampling points by a *zeroth-order hold* block to become $y(t)$.[1] Please note that the true signal values calculated by your DSP to be converted back into an analog signal, will again be quantized by the DAC. Finally, $y(t)$ is further filtered to fully reconstruct the wanted signal $w(t)$. Pretty complicated, don't you think?

In this chapter we will highlight the significance and role of each of these blocks in more detail using the three basic topics of this chapter: sampling, reconstruction and quantization.

5.2 Sampling

Let's see how we get from $v(t)$ to $x[n]$. There are two possibilities:

- the information[2] in the signal $v(t)$ is contained in a frequency band: $[-\omega_B, \omega_B]$.
- the information in the signal $v(t)$ is contained in a small frequency band around a central frequency: $[-\omega_0 - \omega_B, -\omega_0 + \omega_B]$ and $[\omega_0 - \omega_B, \omega_0 + \omega_B]$ with $\omega_B < \omega_0$.

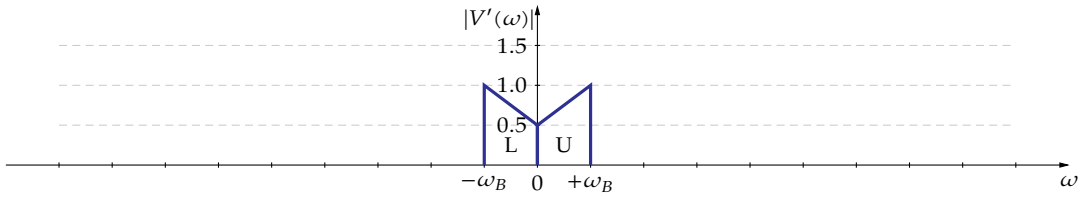
In the former case, we will meet Shannon again and perform *low-pass sampling*. In the latter case, we will see how to be smarter than Shannon and apply *band-pass* sampling.

5.2.1 Low-pass sampling

Consider the bandwidth-limited spectrum ($\omega < \omega_B$) of the signal $v(t)$.

[1] Depending on the type of DAC used, the zeroth-order hold may appear before the actual DAC block.

[2] With information we mean interesting information here. There might be other useful information in the signal (e.g., the television channel next to the channel you're watching), but we're only interested in a specific part.



According to Shannon's theorem (see 41), we need to pick a sampling frequency ω_s that's sufficiently high, i.e.

$$\omega_s \geq 2\omega_B.$$

If we sample the signal, the spectrum will be replicated with a pitch equaling ω_s. We are aware of that: it's not an issue anymore. What's more important is that the replication will also happen for frequency components larger than ω_B (e.g., coming from the television channel next to the one you are watching).
You might argue, that there aren't any such components, however...you're very wrong! There's always spurious components outside of the band of interest. They may be due to noise, distortion, electromagnetic coupling, etc. We indicated such a spurious component in Figure 5.2(a).

We indicated the sampling spectrum in Figure 5.2(b). If we sample the original signal (i.e. convolve it with the sampling spectrum), we obtain the polluted spectrum of Figure 5.2(c). The spurious out-of-band components have become in-band aliases!

Conclusion: we need to remove the out-of-band spurious components before sampling our signal. Removing these spurious components can be easily done using a low-pass filter, the so-called *sampling filter* or *anti-alias filter*. This has been indicated in Figure 5.2(d). The resulting spectrum of $x(t)$ has been indicated in Figure 5.2(e). This filtered spectrum is now ready for sampling.

Exercises
Some exercises on low-pass sampling

Exercise 5.2.1-1: A signal will be sampled at sample rate $\omega_s = 20\,\text{Mrad/s}$. What is the maximal cut-off angular frequency ω_c for an ideal anti-alias filter?

Exercise 5.2.1-2: An ideal anti-alias filter is low-pass filtering a signal with a cut-off frequency $f_c = 20\,\text{kHz}$. What is the lowest sample rate f_s I must achieve to avoid aliasing?

5.2.2 Band-pass sampling

Consider the bandwidth-limited spectrum of the signal $v(t)$. It has a bandwidth ω_B symmetrically around ω_0.

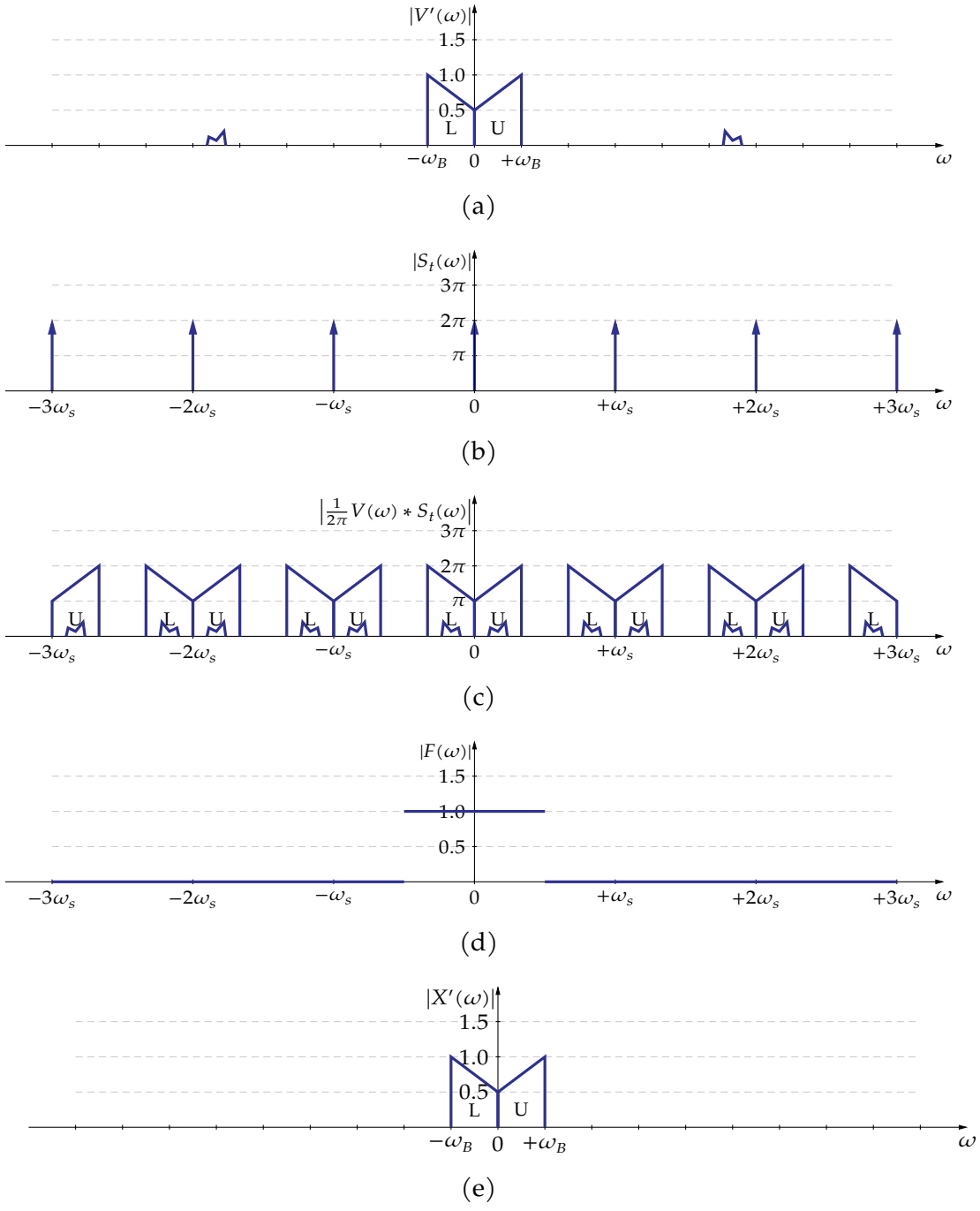
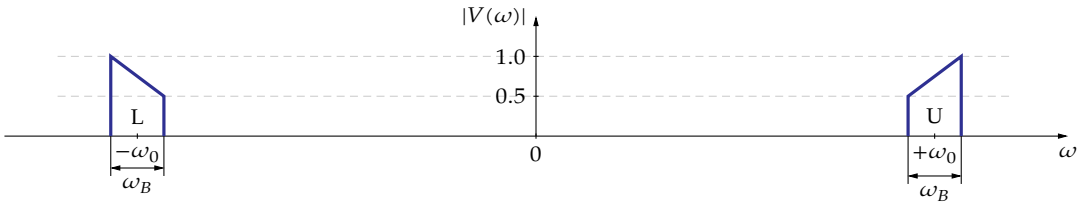


Figure 5.2: Illustration of the concept of low-pass sampling: (a) spectrum of the original signal $v(t)$ with some out-of-band spurious components, (b) sampling spectrum (needs be convolved with original spectrum to model sampling), (c) spectrum of the sampled signal $v_{bad}(t)$ that has been polluted by out-of-band spurious frequencies due to lack of anti-alias filter, (d) sampling filter (anti-alias filter), (e) spectrum of the filtered signal $x(t)$



According to Shannon's theorem (see 41), we need to pick a sampling frequency ω_s that's sufficiently high:

$$\omega_s \geq 2\omega_0 + \omega_B$$

That's the situation we treated in the previous section. The replication pattern after filtering and sampling is indicated in Figure 5.3(a). However, consider what happens if we try to lower the sampling frequency a little (see Figure 5.3(b)). In that case, the upper- and lower-sideband spectra start to overlap: the aliasing ruins our signal.

Now, if we lower the sampling frequency even a little more, we get the situation of Figure 5.3(c). Our original signal together with its replicas is intact and (though spectrally reversed) appears in our base-band sampling range!

And there's more. We can even further lower our sampling frequency until the spectra again start to overlap. This situation is depicted in Figure 5.3(d). This is the first band-pass sampling slot.

Subsequently, we again need to pass a dead-zone until the overlap disappears again and we enter the second band-pass sampling slot. This situation has een depicted in Figure 5.3(e). The slot ends with the situation of Figure 5.3(f). This procedure can be repeated for consecutive slots until the entire range $[-\omega_0, +\omega_0]$ is crowded with replicated spectra. The boundaries of slot 3 have been drawn in Figure 5.3(g) and -(h).

The boundary frequencies for slot n can be easily derived from the drawings:

$$\underbrace{\frac{2\omega_0 + \omega_B}{n+1}}_{\omega_{s,nL}} \leq \omega_s \leq \underbrace{\frac{2\omega_0 - \omega_B}{n}}_{\omega_{s,nU}} \tag{5.1}$$

For $n = 0$ this reduces to the original Shannon condition:

$$\omega_s \geq 2\omega_0 + \omega_b$$

The slot boundaries can more easily be graphically represented, if we normalize (5.1) with respect to ω_0:

$$\frac{2 + \frac{\omega_B}{\omega_0}}{n+1} \leq \frac{\omega_s}{\omega_0} \leq \frac{2 - \frac{\omega_B}{\omega_0}}{n}$$

This set of normalized equations has been graphically represented in Figure 5.4. The valid regions (shaded regions) form a kind of shark-tooth pattern. It is straightforward to show that the crossing points of the lower and upper boundary of a tooth occur at

$$\frac{\omega_B}{\omega_0} = \frac{2}{2n+1}$$
$$\frac{\omega_s}{\omega_0} = \frac{4}{2n+1}$$

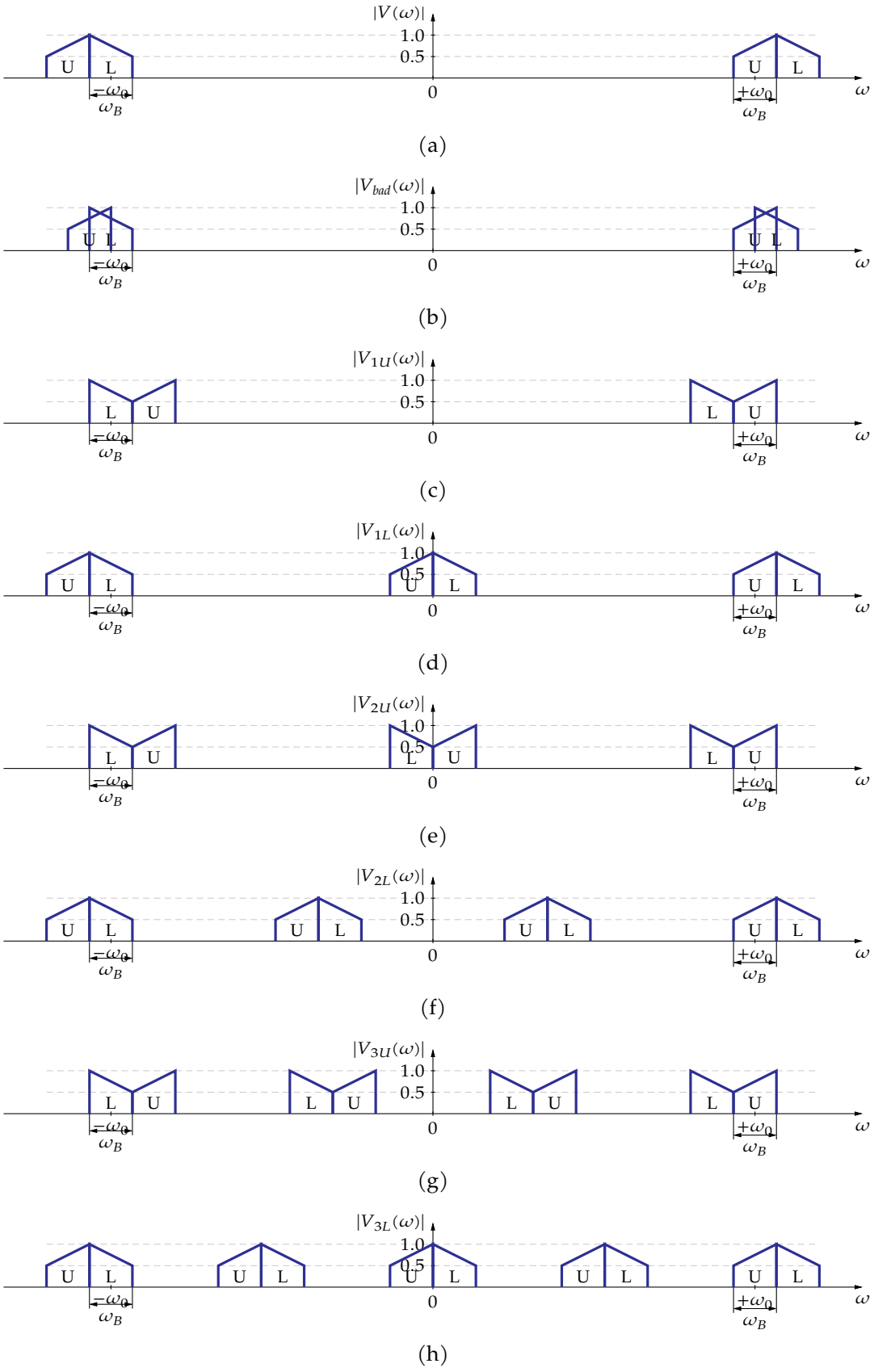


Figure 5.3: Illustration of the concept of band-pass sampling: (a) spectrum of the original signal after Shannon sampling, (b) with a slightly lower sampling frequency, (c) sampling at $\omega_{s,1U}$, (d) sampling at $\omega_{s,1L}$, (e) sampling at $\omega_{s,2U}$, (f) sampling at $\omega_{s,2L}$, (g) sampling at $\omega_{s,3U}$

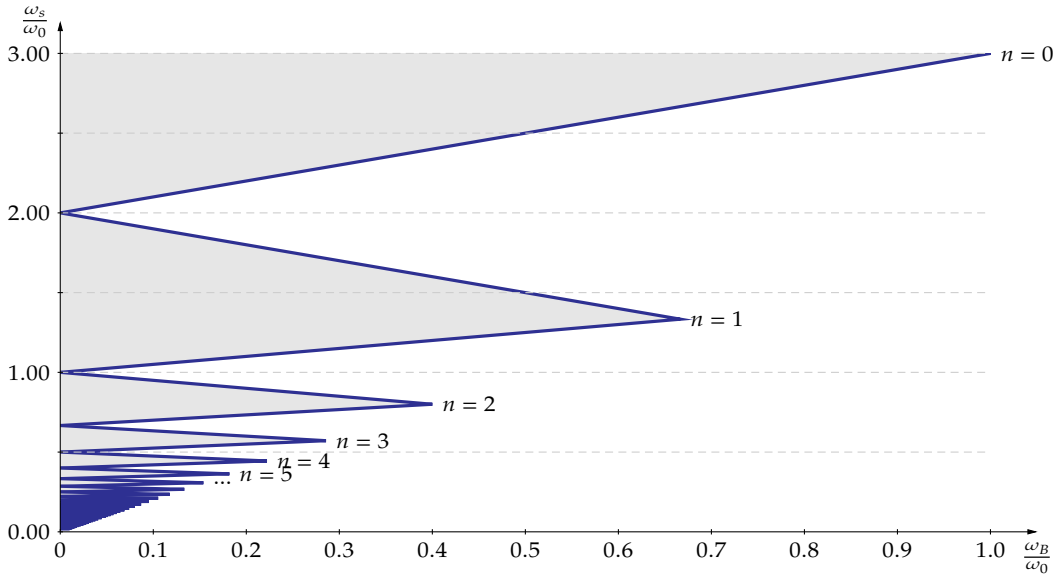


Figure 5.4: Valid band-pass sampling slots (shaded areas)

Remarks

- **Selection of a good sampling frequency**
  It is not very wise to select sampling frequencies at the boundaries of the valid regions, for this causes the lower- and upper-sidebands to abut (see Figure 5.3). In that case, any spurious frequencies around the edges of the sidebands will pollute the original signal. An infinitely steep roll-off anti-alias filter could cure this. Alas, as we will see in section 5.2.3, life is not that ideal.

  Therefore, it is better to select a sampling frequency that has some slack w.r.t. the region boundaries. This slack is often called a *guard band*. A particular good frequency is:

  $$\frac{\omega_s}{\omega_0} = \frac{4}{2n+1} \tag{5.2}$$

  i.e. at the points of the shark's teeth. This has the advantage that the center of the lower- and upper-sidebands are located at $\omega_s/4$. It can be shown that a digital frequency translation over this amount is much simpler than an arbitrary frequency translation.

  As we can see in Figure 5.4, the slack w.r.t. to the region boundaries decrease for increasing values of n. As decreasing the slack implies employing a sharper anti-alias filter, it is not hard to see that a careful trade-off has to be made in selecting a good n that allows a good (low) sampling rate in combination with an acceptable anti-alias filter.
- **Spectral reversal**
  In the odd band-pass sampling slots, the spectrum is reversed, i.e. the lower-sideband is appearing above the upper-sideband. This can easily be corrected by sign reversing our signal every other sample.

  How come? Let's apply our knowledge about frequency spectra and the Fourier transform. Sign reversing every other sample corresponds to multiplying with the infinite time-sequence

  $$s[n] = (-1)^n.$$

  This sequence can be written as:

  $$s[n] = cos(\pi n) = \frac{e^{j\pi n} + e^{-j\pi n}}{2}.$$

  You can find the spectrum of this signal in Figure 5.5(a). Multiplication in the time domain corresponds to convolution in the frequency domain. For an example signal $x[n]$ (resulting

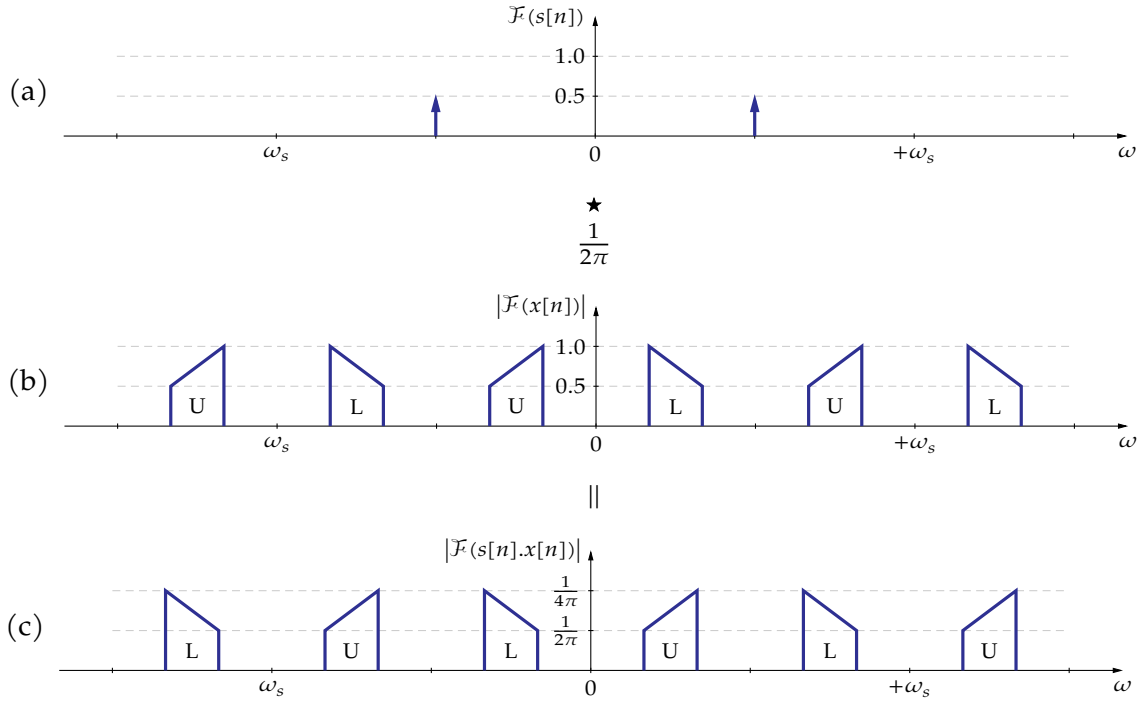


Figure 5.5: Illustration of spectral reversal applied on a band-pass sampled signal $x[n]$: spectrum of (a) $s[n] = (-1)^n$, (b) $x[n]$, (c) $s[n].x[n]$.

from a band-pass sampling according to (5.2) (see Figure 5.5(b)), you can find the convoluted spectrum in Figure 5.5(c). As one can see, the spectral reversal has been undone.

Exercises

Some exercises on band-pass sampling (Hint: make drawings of the spectrum to guide your answer)

Exercise 5.2.2-1: Consider a signal with nonzero spectral content for 100 kHz $\leq f \leq$ 105 kHz. Is a bandpass sampling frequency $f_s = 50$ kHz a good frequency? Does spectral inversion occur in this case?

Exercise 5.2.2-2: Consider a signal with nonzero spectral content for 50 kHz $\leq f \leq$ 58 kHz. Is a bandpass sampling frequency $f_s = 27$ kHz a good frequency? Does spectral inversion occur in this case?

Exercise 5.2.2-3: Consider a signal with nonzero spectral content for 26 MHz $\leq f \leq$ 31 MHz. What is the lowest bandpass sampling frequency I can use? Does spectral inversion occur?

Exercise 5.2.2-4: Consider a signal wih nonzero spectral content for 26 MHz $\leq f \leq$ 31 MHz. What is the lowest bandpass sampling frequency I can use avoiding spectral inversion?

5.2.3 Sampling (anti-alias) filters

A typical beginner's error is to think that you can let your DSP do the low-pass or band-pass anti-alias filtering: huge mistake. The filtering needs take place before the sampling, i.e. in the analog domain. Any spurious frequency present at the moment of sampling will translate itself into an unwanted alias that spoils the party.

Ideally, the anti-alias filter is a perfect low-pass filter, i.e.

1. it has a flat response in the passband (gain equaling 1);
2. it has an infinitely steep roll-of;
3. it has zero gain in the stopband;
4. it has a zero (or at least only linear) phase characteristic.

The first requirement is to minimize the gain distortion of the wanted passband signals. The second is to maximize the usable portion of the passband. The third is to avoid aliasing. The fourth is to maximally maintain the time-domain shape of the signal.

However, the analog world is not as perfect as we would have wanted it to be. We'll have to deal with the typical analog filters:

- Butterworth filters: maximally flat magnitude response filter
- Bessel filters: maximally linear phase response filter
- Chebyshev filters: steep roll-off filter (in exchange for limited passband or stopband ripple)
- Elliptical filters: even steeper roll-off in exchange for ripple in both passband and stopband

It is not our goal to explain how to design these analog filters, but some basic knowledge of their properties is usefull. So, let's start by taking a look at the frequency responses of some of these filters. The table below serves as a lookup table for the drawings.

Type	Logarithmic Bode plot	Linear Bode plot	Step response
Butterworth	Figure 5.6(a)	Figure 5.6(b)	Figure 5.10(a)
Bessel	Figure 5.7(a)	Figure 5.7(b)	Figure 5.10(b)
Chebyshev-I	Figure 5.8(a)	Figure 5.8(b)	Figure 5.10(c)
Chebyshev-II	Figure 5.9(a)	Figure 5.9(b)	Figure 5.10(d)

The filters presented have not been designed using an actual specification in mind. They only have been "designed" such that a comparison is not totally unfair. The goal is not to compose a 6-digit ranking, but get the flavor of each of these filter types. Just to give you a rule-of-the-thumb guide when you ever have to select a sampling filter.

The goal of our sampling filter is to block out unwanted frequencies that might give rise to aliases.

So, let's first take a look at how good the filters are at blocking out unwanted signals in the

stopband. The logarithmic Bode plot of the Chebyshev-II filter kind of pops out. The problem with this type of filter is that the gain of the filter does not go down for increasing frequencies. Sampling an input signal that has gone through this kind of filter, will result in an overload of foreign frequencies, not only neighbouring ones, but also very distant ones. Of course, one can tighten the specs to allow only 0.1% or 0.01% ripple in the stop band. However, it keeps rippling from the stopband edge frequency to infinity! This is a sure way to bury yourself in loads of spurious garbage. Consider Chebyshev-II to be out.[3]

Inspecting the logarithmic Bode plots further for good stopband blockers, brings us to two conclusions:

- the higher the order the better;
- Chebyshev I is doing better than Butterworth, in turn better than Bessel.

Nice...but logarithmic plots are good at hiding details that might be important. Indeed, taking a look at the passbands in the linear Bode plots[4] reveals that the passband of the Chebyshev-I is also suffering from ripple[5]. It will depend on your application whether you can tolerate such a ripple. Or maybe you have enough money to spend to digitally correct the ripple. It is not that easy, because you will need some careful calibration. Of course the Butterworth (as maximally flat gain filter) is thé champion in ensuring you a good passband.

However, if we take a look at the time-domain response of the filters (i.e. related to phase linearity), than there is only one conclusion: take a high-order Bessel filter! Bessel filters are the perfect filters for maintaining low and uniform delay. Chebyshev filters are really no good at maintaining time-domain shapes. Butterworth filters are also not that impressive in this respect. If linear phase is important to you (i.e. to your application), Bessel is the way to go.

As you might have guessed, there's no single answer to "what type of filter should I use as an anti-alias filter?" So be it. Though...there are two conclusions to carry from this discussion:

1. if you take a look at the linear Bode plots, you will be convinced that every filter has a significant roll-off band. Your sampling frequency needs to be high enough to also sample the transition band frequencies. Otherwise, they will pollute your real signal. So, in practice, the region from $0.4\omega_s$ to $0.5\omega_s$ will be of no use to you. The garbage in this "guard band" may be filtered away later...in the digital domain.[6]
2. you will never be able to recover from the havoc caused by a badly chosen or badly designed analog filter. Selecting and designing your anti-alias filter is also DSP!

5.3 Reconstruction

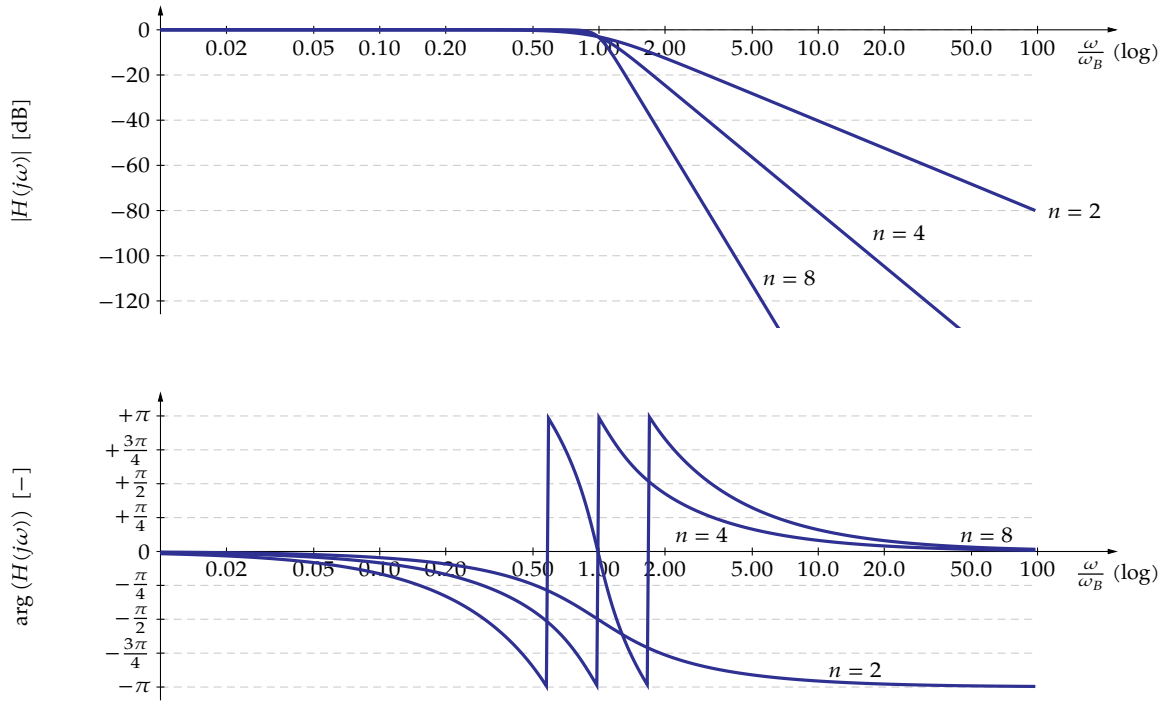
Let's see how to get from $y_D[k]$ to $w(t)$.

[3]This is also the reason why we did not bother to plot any elliptical (Cauer) filter characteristics: it suffers the same stopband ripple.

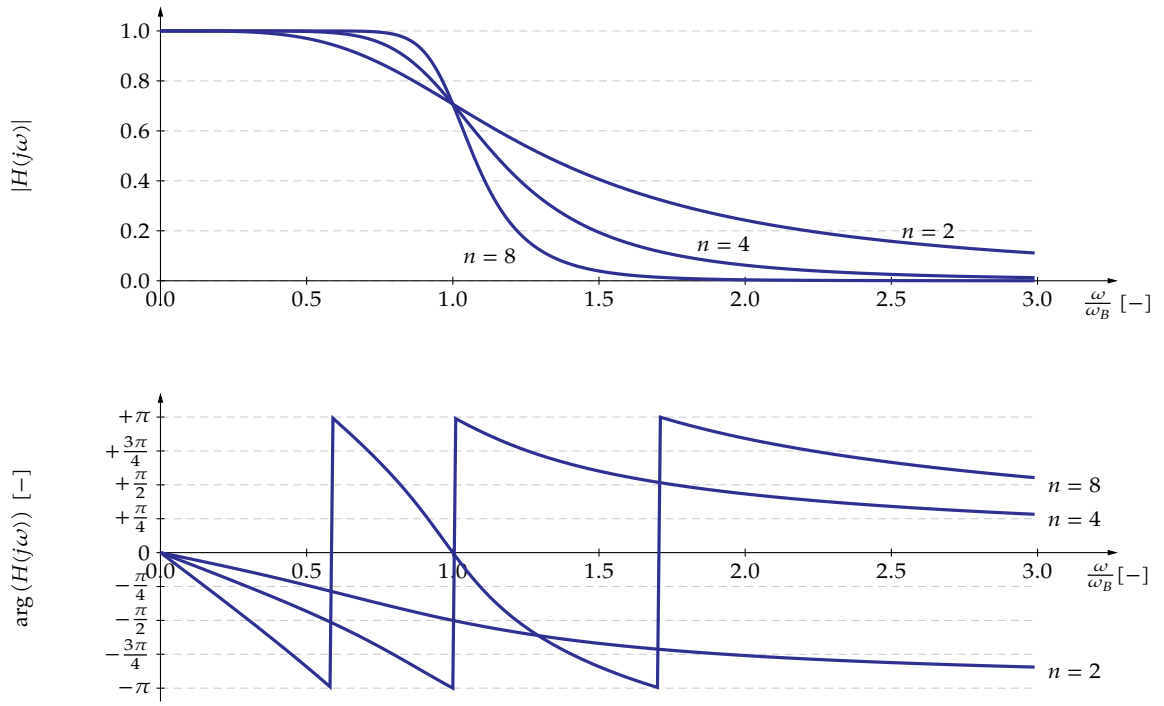
[4]Purists will claim that these are not *real* Bode plots, but don't let this fundamentalism distract you.

[5]As you might remember from your analog classes, Chebyshev filters are based on cylindrical cosine projections, so the ripple is not there by accident.

[6]Likewise, the gain distortion in the pass-band (if important) may be corrected in the digital domain.

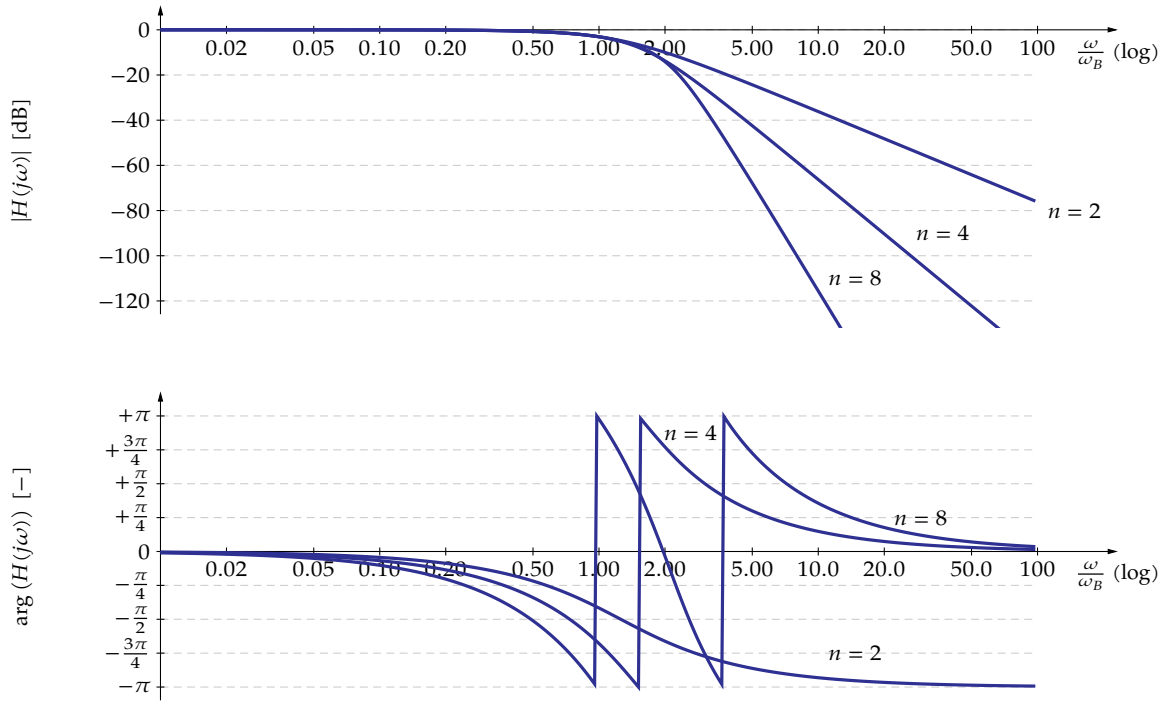


(a)

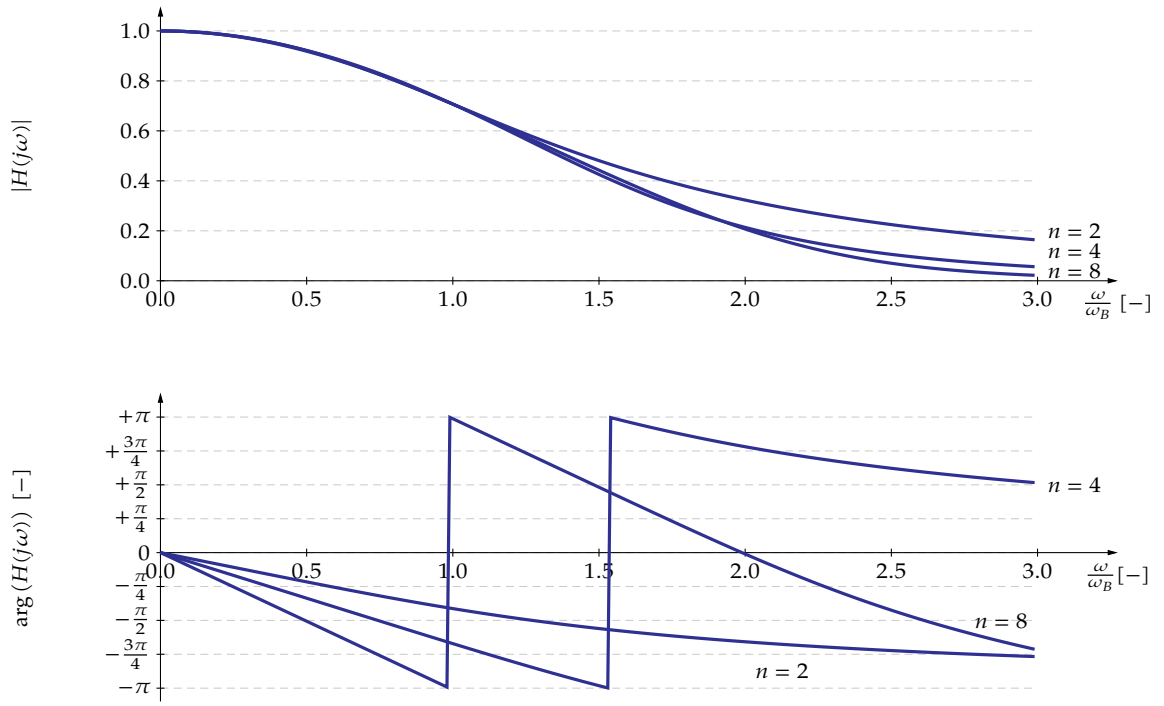


(b)

Figure 5.6: Bode plot of a low-pass analog Butterworth filter for orders 2, 4 and 8 (designed for $\omega_{-3\,\mathrm{dB}} = \omega_B$): (a) on a logarithmic scale, (b) on a linear scale.

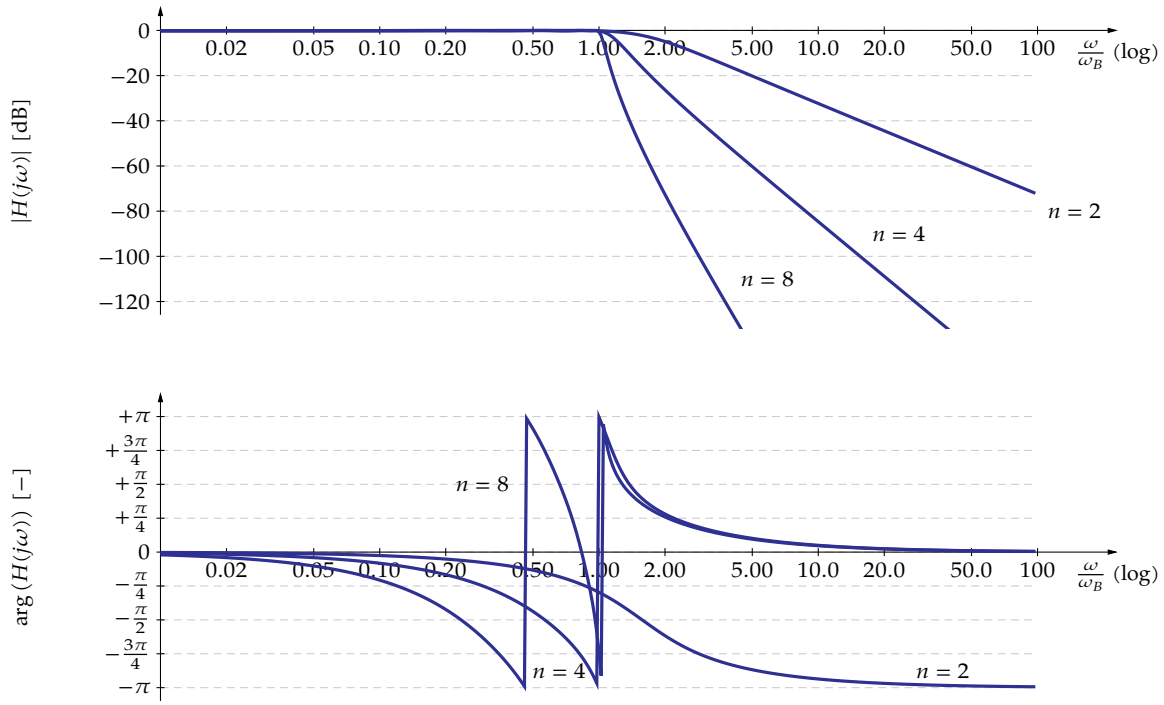


(a)

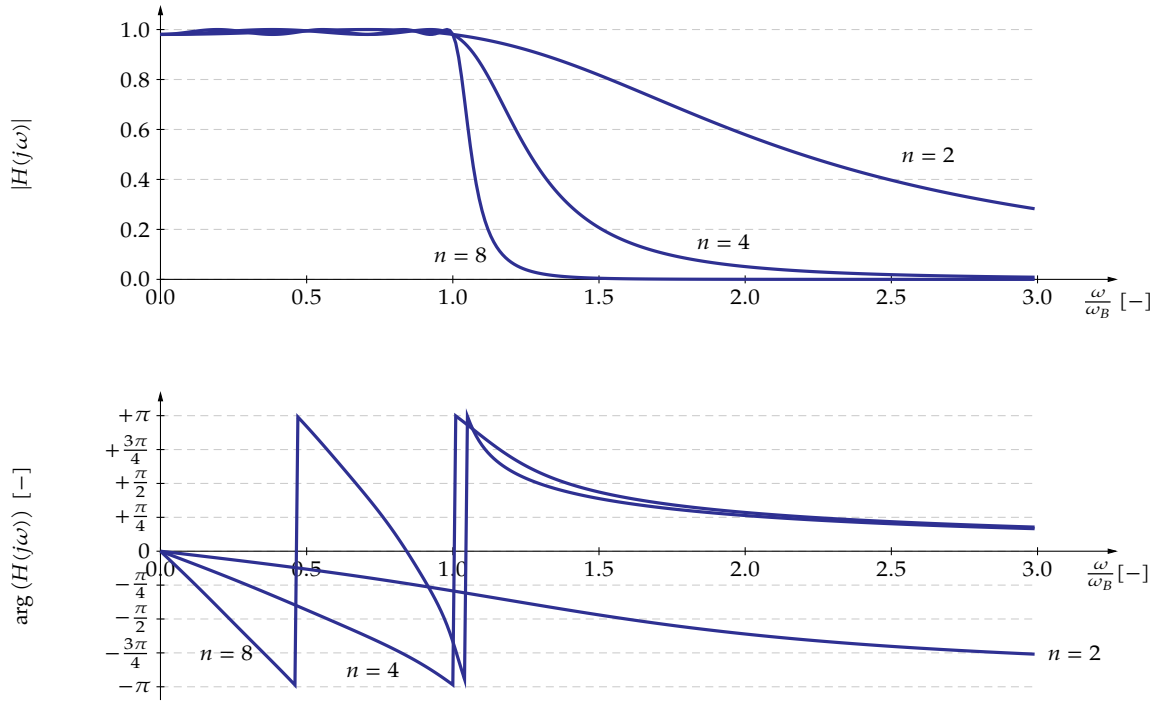


(b)

Figure 5.7: Bode plot of a low-pass analog Bessel filter for orders 2, 4 and 8 (designed for $\omega_{-3\,\mathrm{dB}} = \omega_B$): (a) on a logarithmic scale, (b) on a linear scale.

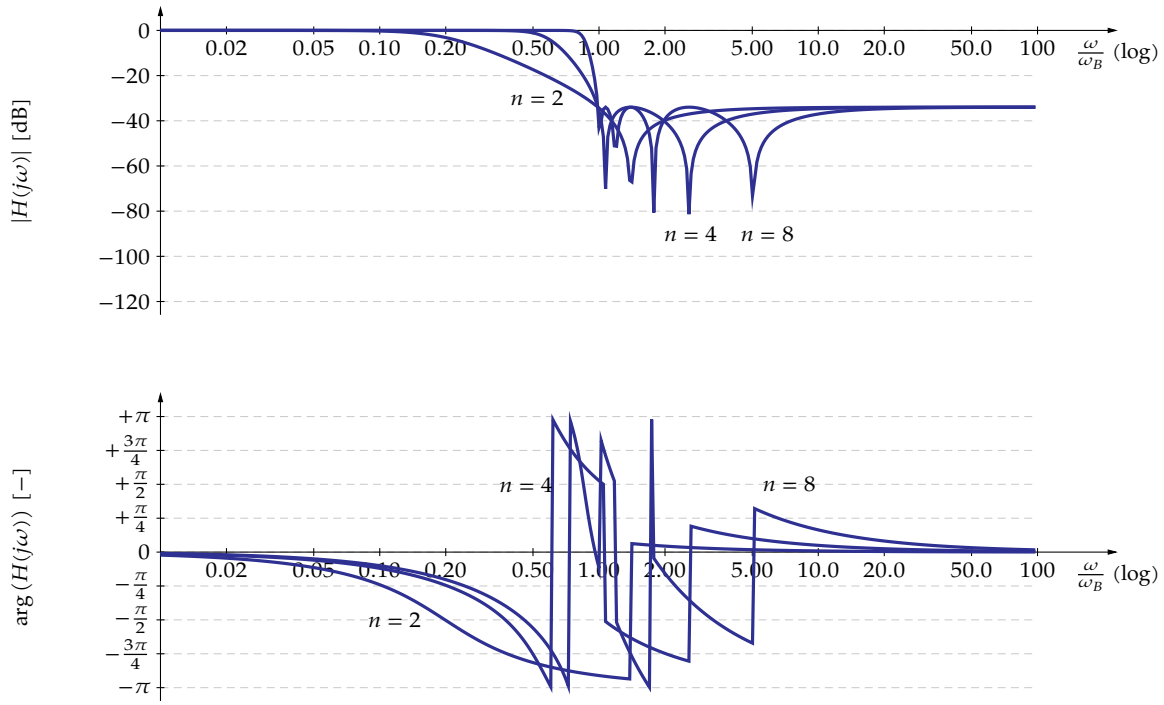


(a)

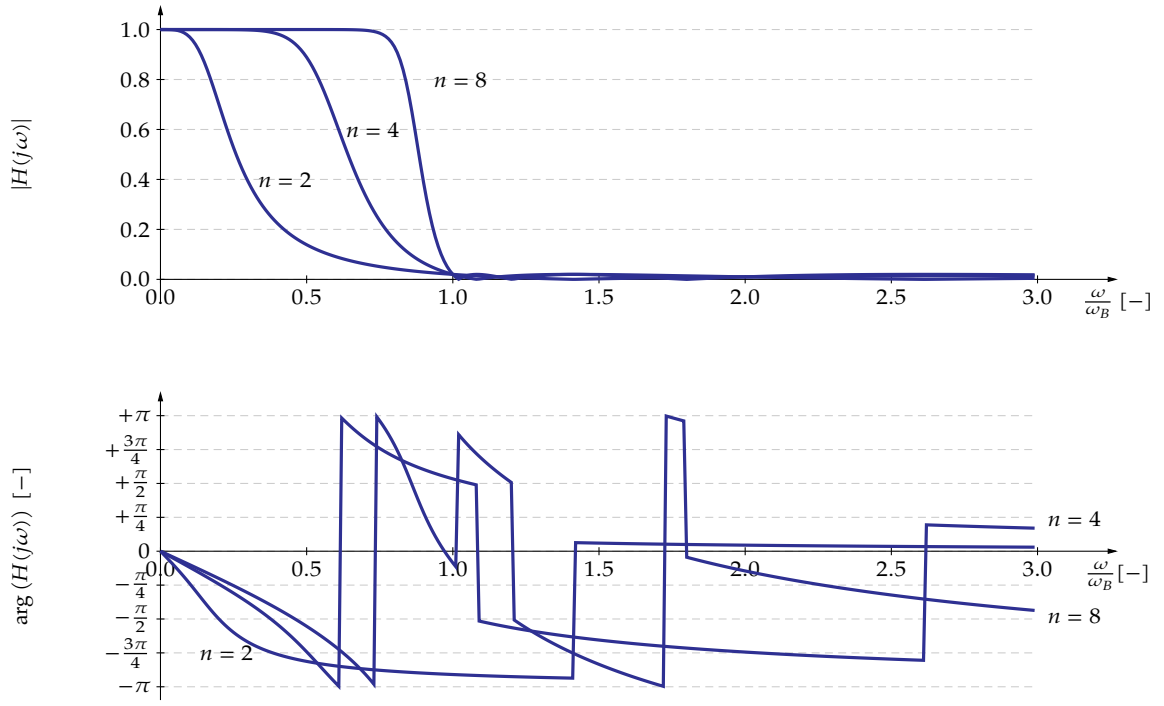


(b)

Figure 5.8: Bode plot of a low-pass analog Chebyshev (type I) filter for orders 2, 4 and 8 (designed for $\omega_{passband} = \omega_B$ and 2% passband ripple): (a) on a logarithmic scale, (b) on a linear scale.



(a)



(b)

Figure 5.9: Bode plot of a low-pass analog Chebyshev (type II) filter for orders 2, 4 and 8 (designed for $\omega_{stopband} = \omega_B$ and 2% stopband ripple): (a) on a logarithmic scale, (b) on a linear scale.

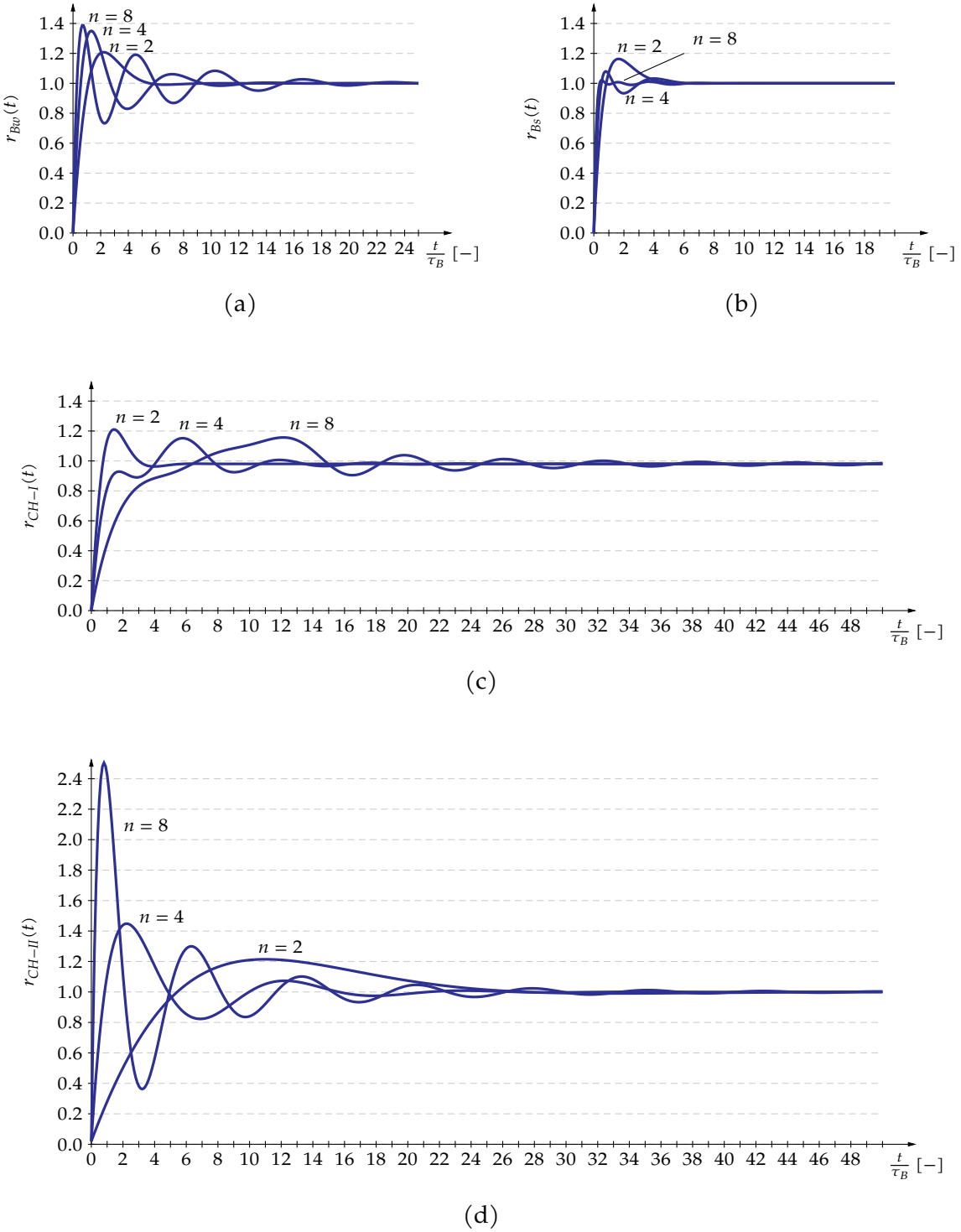


Figure 5.10: Step responses of analog filters (order 2, 4 and 8): (a) Butterworth filter, (b) Bessel filter, (c) Chebyshev (type I) filter, (d) Chebyshev (type II) filter.

5.3.1 Theory

In our theoretical view on reconstruction that we developed in chapter 3, reconstruction was as simple as applying a low-pass filter to the Dirac impulses that come out of our DSP core.

However, in our digital signal processor, signals are represented by numbers stored in memory: in no way close to Dirac impulses. In addition, as mentioned in the introduction, Dirac impulses are difficult to generate in the analog domain.

5.3.2 Practice

It is more easy to have the DAC generate the analog representation of our signal at the sampling points and have the DAC or a separate circuit maintain the values during the entire sample period. This is called a zeroth-order hold effect.

This effect corresponds in the time domain to convolution with a rectangular unit pulse:

$$zoh(t) = \begin{cases} 1/T_s & \text{if } t \in [0, T_s] \\ 0 & \text{if } t \notin [0, T_s] \end{cases}$$

Of course, such an operation cannot go without consequence in the frequency domain. Figure 5.11 illustrates the effect. The spectrum of the zeroth-order hold signal is:

$$ZOH(\omega) = \text{sinc}\left(\frac{\omega T_s}{2}\right) \mathrm{e}^{-\mathrm{j}\frac{\omega T_s}{2}}$$

Luckily, this effect exhibits a linear phase. Therefore, it introduces no extra time-domain distortion. However, in our signal band there is an extra attenuation caused by the zeroth-order hold effect. Luckily, we can compensate for that by using a reconstruction filter with a special shape. The perfect compensating reconstruction filter therefore is a linear phase filter with the following spectrum:

$$R(\omega) = \mathrm{e}^{-\mathrm{j}\omega\tau} \begin{cases} \frac{1}{\text{sinc}(\pi\frac{\omega}{\omega_s})} & |\omega| \leq \frac{\omega_S}{2} \\ 0 & |\omega| > \frac{\omega_S}{2} \end{cases}$$

The magnitude spectrum of this filter has been indicated in Figure 5.11.
The value of τ in the equation above represents phase portion of this filter and therefore represents the (extra) delay in the compensation filter. Indeed, the ZOH-operation causes a nonzero delay that cannot be undone. One would need a noncausal filter to do so. On the contrary, an extra delay will be added by the compensation filter. The value of τ will be equal to half the length of the impulse response of the linear-phase filter. If you want to keep τ low, then you must limit the length of the compensation filter.

The setup with the special compensation reconstruction filter can be found in Figure 5.1 with the only specific detail that the reconstruction filter is not just an ordinary low-pass filter, but has a special 'one over sinc'-shape. However, the problem with this type of filter is similar to the problems with the filters presented as sampling filters in section 5.2.3: infinitely sharp roll-off is not possible. In addition, the filter has a sharp, peaky edge and therefore is even more difficult to make.

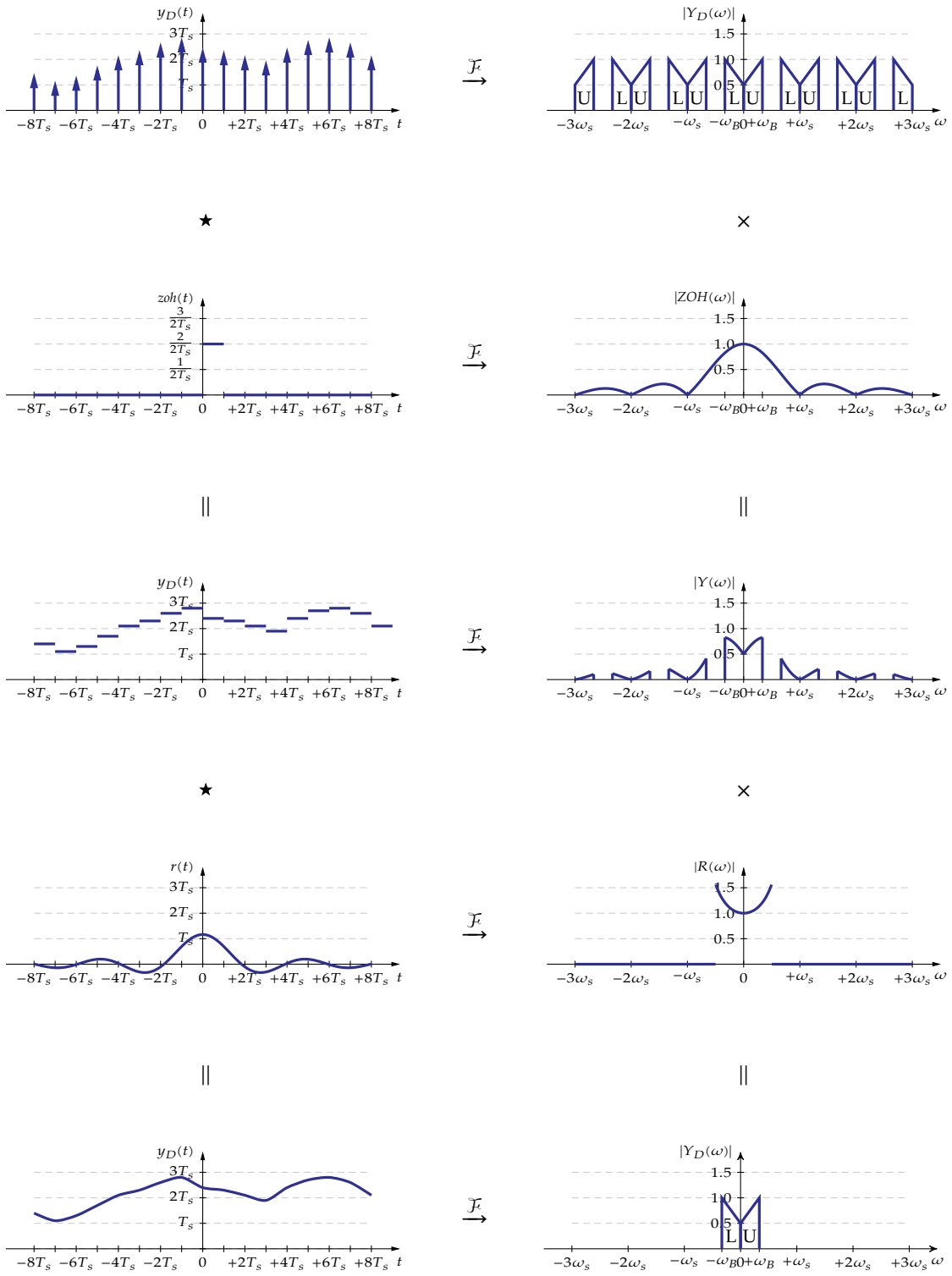
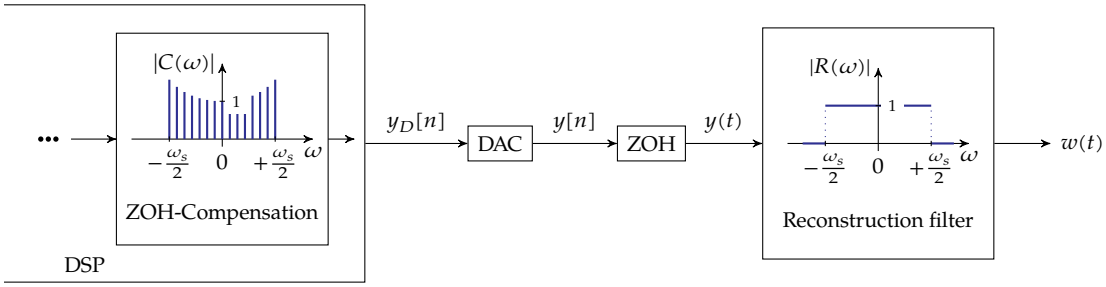


Figure 5.11: Signal reconstruction using a zeroth-order hold circuit and an appropriate reconstruction filter.

Therefore, most often, the correction of the ZOH-effect is done digitally before the reconstruction takes place (as a last filtering step in the DSP before sending the signal to the DAC). In that scenario the reconstruction filter still is an ordinary low-pass filter, and the compensation is taken care of digitally (where generating special filter shapes is a piece of cake). One can even compensate further for nonidealities in the ordinary low-pass reconstruction filter.

In this scenario the back end of our DSP system looks as follows:



5.4 Oversampling and multi-rate techniques

In this chapter, one bottleneck appears over and over: the analog filters needed to successfully complete sampling and reconstruction. The specs imposed on these filters are very tight.

There is one way out. The specifications of these filters can be loosened by trading in analog for digital. Some would even say: trade in expensive, unreliable analog for inexpensive, cheap, reliable digital. I agree to the cheap vs. expensive (in this case at least), but not to the reliable vs. unreliable. Sure, digital algorithms can be made much more precise than analog electronics, but there the story ends.

5.4.1 Oversampling

A very simple technique is to just *oversample* the signal. This means sampling it at a multiple of the required Shannon rate.

Pro: looser filter spec

E.g., consider a 20 kHz hifi audio signal, which is — in a recording environment in audio studios — frequently sampled at 48 kHz. Consider that we will be sampling this signal using a 16bit ADC. This means that we need a filter that rolls off in the band from 20 to 24 kHz, starting at a gain of −3 dB at the edge of the pass-band and ideally reaching a gain of $\frac{2\sqrt{3}}{2^{16}}$ at the start of the stop-band (at 24 kHz). Assuming we're a top-notch audio-freak, our goal might be to have the gain dropped by 98 dB at the start of the stop-band. This means 98 dB per 0.08 decade roll-off. If we'd use a Butterworth filter, our filter order will end up somewhere around 62: go figure!

Now, let's assume we can oversample the signal four times, i.e. sampling it at 192 kHz. In this

way, the roll-off region increases from 0.08 decade to 0.68 decade. The filter order will go down to a reasonable 7 to 8. That should work.

Con: faster ADC

Con: extra digital filter required

As a consequence of relaxing our filter specs we're faced with a wasteland of spurious material starting at the edge of our signal-band going all the way up to $f_s/2$. We need to filter this garbage away. Luckily, we can now do this in the digital domain. As we will see later, creating digital filters with very sharp transitions is no problem at all. Still you pay a price for this extra filter. You will have less processing time available for your real application.

Con: more data

The good thing about the increased data rate is that it also helps you loosening the specs for the reconstruction filter. The bad thing is that your DSP is now receiving n times as much data per second to process and therefore needs to be n times as performant. If you can afford this, no problem. However, it is a waste of money, as there is only signal content up to your signal bandwidth. The rest has been filtered away by the extra digital filter mentioned above. Using multi-rate techniques is the solution to this.

5.4.2 Multi-rate conversion

Oversampling with decimation in time

A big disadvantage of oversampling was the increased datarate. A very simple solution to this is decimation. Why the term 'decimation'? Roman officers punished their soldiers by killing one out of every ten soldiers. We're going to do the same with our samples. An important difference is that we will kill nine out of ten soldiers, keeping only one!

We define decimation in time by a factor of m as consecutively keeping 1 sample, and removing the next $m-1$ samples, keeping 1, removing $m-1$, etc.

Mathematically, this corresponds to multiplying the original continuous-time signal corresponding to our discrete-time signal $y[n]$ by a Dirac impulse train $\text{Ш}_m[n]$:

$$\text{Ш}_m[n] = \sum_{k=-\infty}^{+\infty} \delta[n - km]$$

This is like sampling at a reduced rate:

$$\omega_{s,m} = \frac{\omega_s}{m}$$

Hence the spectrum becomes periodic with period ω_s/m. Of course, if our original signal has a bandwidth ω_B, we must ensure that:

$$\omega_{s,m} \geq 2\omega_B$$

If this condition is not met, we need a sampling (anti-alias) filter to remove the frequencies that are subject to become in-band aliases. This is a so called *decimation filter*. The extra filter needed to realize oversampling (one of the con's listed there) can be considered to be such a decimation filter.

Except for this decimation filter, decimation in time is one of the cheapest operations from a computational perspective.

Interpolated reconstruction

A similar technique may be used to ease the specifications of the reconstruction filter. The technique is called *time interpolation* and is the direct opposite of decimation in time.

We define time interpolation with a factor of m as increasing the sample rate with a factor of m. This corresponds to inserting $m-1$ new samples in between any two samples. The key question is: what will be the values of these samples? Many strategies could be considered:

- repeating the most recent sample $m-1$ times (corresponds to zeroth-order interpolation, or zeroth-order hold)
- linear interpolation in between the adjacent existing samples
- higher-order interpolation
- zero stuffing (i.e., inserting zeros)
- ...

For signals where the information is encoded in the time-domain, the time-domain interpolation techniques may be just appropriate. However these simple interpolation techniques greatly affect the spectrum of the resulting signal. However, in our application (reconstruction of a spectrum in the band $[-\omega_B, \omega_B]$) the authenticity of this spectrum is not to be affected. Strangely, setting the new samples to zero in between the existing old samples (zero stuffing), seems to do the perfect job. The term *interpolation in time* is often used to refer to this particular subclass of time interpolation techniques.

Let's consider the Discrete-time Fourier Transform of our original signal:

$$X_p(\omega) = \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}\omega n T_s}$$

Now, if we interpolate the signal by a factor of m using zero stuffing, i.e.,

$$x'[n'] = \begin{cases} x[n] & \forall n' = mn \\ 0 & \forall n' \neq mn \end{cases}$$

then, the Discrete-time Fourier Transform of the newly composed signal becomes:

$$X'_p(\omega) = \sum_{n'=-\infty}^{+\infty} x'[n']\, \mathrm{e}^{-\mathrm{j}\omega n' \frac{T_s}{m}} \tag{5.3}$$

As we know that $x[n'] = 0$ for $n' \neq mn$, we only need to consider the values of $n' = mn$. For

those values we know that: $x'[n'] = x[n]$. Therefore, (5.3) reduces to:

$$\begin{aligned} X'_p(\omega) &= \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}\omega m n \frac{T_s}{m}} \\ &= \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}\omega n T_s} \\ &\overset{!!}{=} \end{aligned}$$

This means that zero stuffing doesn't affect the spectrum!

Low-pass filtering this zero-stuffed signal to only keep the band $[-\omega_s, \omega_s]$ generates the intermediate samples of the signal exactly as if we sampled the original analog signal at a rate $m\omega_s$ instead of ω_s. This is not really magic: no new 'information' has been generated. This can be easily understood, by realizing that decimating an originally oversampled signal, does not destroy information either. As long as the decimation respects Shannon's theorem, the amount of information remains. Quite a conclusion.

The key point is that we can do this low-pass filtering in the digital domain, and therefore can make a pretty accurate low-pass filter that realizes the interpolation.

Concluding: the price we pay is an extra digital interpolation filter, an m times faster DAC. The benefit we get is a relaxed analog reconstruction filter.

5.5 Quantization

So far, we did not mention quantization at all. Quantization is a necessary step in making signals tractable for a digital computer (i.e. in our case a DSP). This is basically for two reasons:

- Computers cannot represent a continuous spectrum of real numbers. So, be aware that even a floating point representation of a real number is a mapping of that number onto the discrete set of numbers that can be represented.
- The precision of the available converters (ADCs and DACs) is limited. In addition, precision and speed are on a trade-off curve. One can be traded for the other. Therefore limiting the precision very often is unavoidable.

The limited precision of the converters results in a *conversion error*, which in many cases, we can model as a noise source at the input.
The limited internal DSP number precision (which is often much higher than the converter precision) continuously causes *truncation* or *round-off* errors that propagate in your algorithm. In a bad scenario, they accumulate and will be one of the problems you face during your DSP design.

In this chapter we will focus on the quantization itself.

5.5.1 Quantizing the signal range

Quantizing an analog signal using a limited number of values, assumes that the range of the analog signal is bounded.

To ease the discussion, let's assume our signal has a symmetrical range around zero:

$$-x_{\max} \leq x(t) \leq x_{\max}$$

The essence of discretization is to subdivide this range into a number of intervals, each interval mapping onto a specific quantized value. Actually, the ADC will map the interval onto a binary code L_i. After processing, our DSP will have to map this code to a quantized value. In this way, every value in the range of the original signal $x(t)$ is mapped onto its discretized level $\hat{x}(t)$:

$$x(t) \mapsto L_i \mapsto \hat{x}(t)$$

In practice, the range of $\hat{x}(t)$ does not need to correspond to the range of $x(t)$. However, for our purposes, this assumption makes the treatment easier without restricting the validity of our conclusions below.

We can distinguish two major ways of defining such a mapping:

- fixed intervals, i.e. the intervals are independent of the signal;
- variable intervals, i.e. the intervals are signal dependent.

5.5.1.1 Fixed quantization

If the entire range is subdivided in intervals of equal size, we perform a so-called *uniform quantization*. If the intervals are unequal in size, we perform a so-called *nonuniform quantization*.

Uniform quantization

Two flavors can be distinguished:

- N-bit mid-rise quantization

$$\hat{x} = \frac{0.5 + \left\lfloor \frac{x}{\Delta} \right\rfloor}{2^{N-1}} x_{\max} \qquad \text{with} \qquad \Delta = \frac{2x_{\max}}{2^N} \tag{5.4}$$

- N-bit mid-tread quantization

$$\hat{x} = \frac{\left\lfloor \frac{x}{\Delta} + 0.5 \right\rfloor}{2^{N-1} - 0.5} x_{\max} \qquad \text{with} \qquad \Delta = \frac{2x_{\max}}{2^N - 1}$$

The former is very common for ADCs. The latter is very commonly used in compression techniques. The equations are rather complicated and not really suited to be memorized. Graphically, the two versions are more easily comprehended.

The graphical representation in Figure 5.12, also illustrates the origin of the names of both flavors. We also indicated the theoretical ideal quantization line (the dashed diagonal line). Deriving the equations from the graphs is not that difficult. To prepare ourselves to treat converter nonidealities in a later section we also indicated the *quantization levels* t_k on the graph.

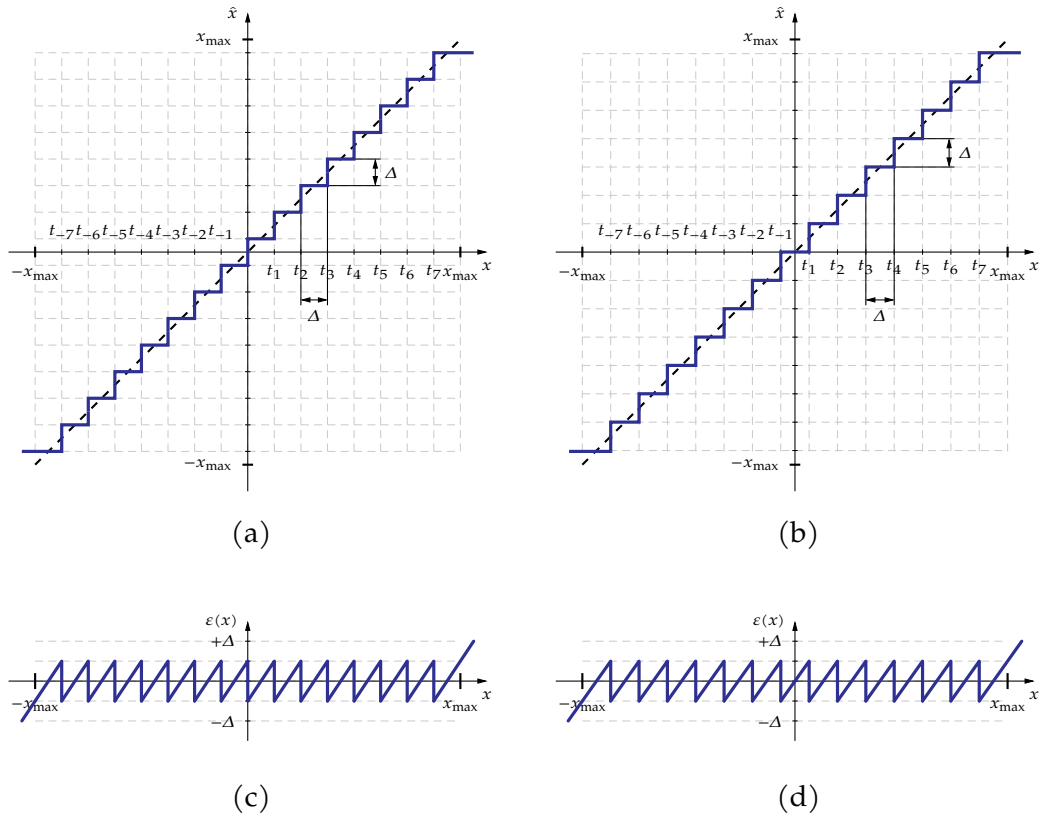


Figure 5.12: Illustration of mid-rise (a) quantization and (c) quantization error, and mid-tread (b) quantization and (d) quantization error, for a 4bit quantizer.

Nonuniform quantization

The goal of nonuniform quantization is *companding*. Companding is a combination of *com*pression and ex*pan*sion. If the information is coded exponentially in the amplitude (or frequency magnitude) of a signal then compressing the signal in a logarithmic way during sampling is an efficient way to reduce the number of bits needed to adequately represent the information. Speech is a such a signal.

This reduction in number of bits is in general not needed for DSP applications. However, in telecommunication systems, where the signal needs to be digitally transmitted, the reduction ís important.

Sampling using unequal intervals can be accomplished by using a higher-resolution ADC followed by a look-up table, or by using a ADC with nonlinear quantization intervals.

A good example of such nonuniform quantization is the quantization of speech signals in telephony applications. In order to adequately uniformly quantize speech for telephony applications, approximately 12bits of resolution is needed.

By applying a nonuniform quantization scheme, the required number of bits can go down to 8. Two nearly identical standards that are used for companding telephony speech signals are:

- A-law: used in Europe
- μ-law: used in North America

The nonlinearity in the quantization is defined by the following relationships.

$$\text{A-law: } y = \begin{cases} \frac{Ax}{1+\ln(A)} & \text{for } 0 \leq x \leq 1/A \\ \frac{1+\ln(Ax)}{1+\ln(A)} & \text{for } 1/A < x \leq 1 \end{cases}$$

$$\mu\text{-law: } y = \frac{\ln(1+\mu x)}{\ln(1+\mu)}$$

with

$$A = 87.6 \text{ or } 87.7$$
$$\mu = 255$$

These relationships only describe the positive half of the signal range. The negative half is symmetrical (see Figure 5.13). Subsequently these transformed variables are quantized uniformly using 8 bits.

Mathematically, these relationships seem very simple. However, implementing them in electronics is everything but simple. Dedicated ADCs and DACs that have these companding features on board do exist.

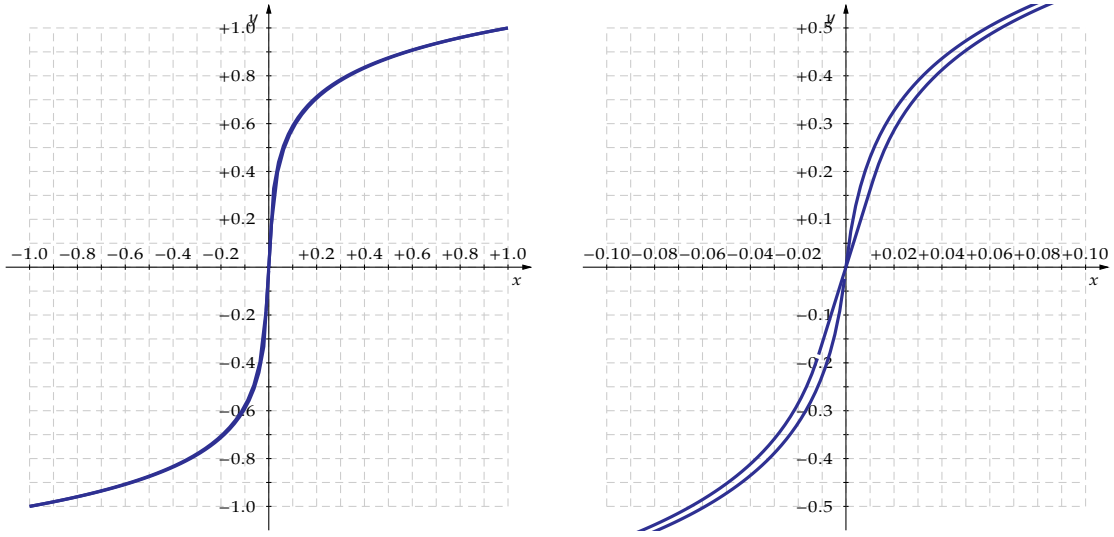


Figure 5.13: Nonuniform quantization of speech signals according to A-law and μ-law: (a) full-scale view, (b) detailed view.

Exercises
Some exercises on quantization

Exercise 5.5.1.1-1: Load a 10 seconds 16-bit music sample at 44.1 kHz using Octave (e.g. grab it from one of your favorite CDs). Make sure you can load and play the sample. The commands you need for this are:

- `sound`
- `auload`

They are part of the audio package.

The samples are saved as values between -1 and 1. Requantize them with a resolution of 8-bit and compare the quality of the result with the orginal 16-bit speech sample.

Exercise 5.5.1.1-2: Now requantize the 16-bit music sample of exercise 5.5.1.1-1 nonlinearly to 8-bit using mu-law (using Octave's `lin2mu`). Then relinearize it (using `mu2lin`) and sound it. Compare the quality of the result with the 8-bit linearly quantized version you made in the previous exercise.

Exercise 5.5.1.1-3: First, reduce the loudness of the 16-bit music sample of exercise 5.5.1.1-1 by a factor of 10, and then compare the linear 8-bit quantization with the nonlinear 8-bit mu-law quantization (using Octave's `lin2mu` and `mu2lin`).

5.5.1.2 Variable quantization

In a variable quantization approach the intervals are calculated based on the signal as it was sampled so far. The idea is to increase the interval size for fast varying signals and to decrease the interval for slowly varying signals. An example of this is the Continuously Variable Slope Delta (CVSD) modulator.

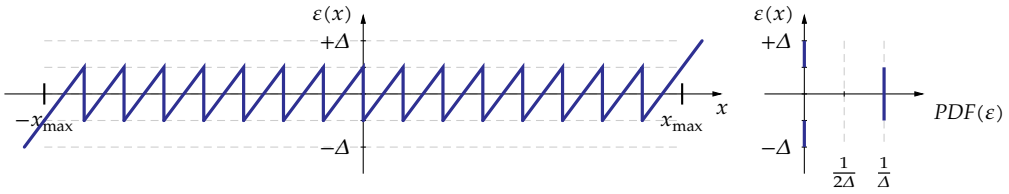
5.5.2 Quantization error

Modeling the quantization error in general is a difficult task. However, under certain circumstances a simple model may be sufficient.

5.5.2.1 Uniform quantization error model

For a fixed uniform quantization approach, the error is a shark-tooth shaped. For the mid-rise scheme that is used for ADCs, this pattern can be found in Figure 5.12(c) (and is replicated below for your convenience).

If we assume that the signal that is being quantized is bounded to the quantization range and has a rich spectral content, we may assume that the quantization error has a uniform probability density function on the range from $-\Delta/2$ to $\Delta/2$.



The standard deviation of the quantization error can be easily calculated:

$$\begin{aligned}\sigma(\varepsilon) &= \sqrt{\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \varepsilon^2 \frac{1}{\Delta} d\varepsilon} \\ &= \sqrt{\frac{1}{\Delta}\left[\frac{\varepsilon^3}{3}\right]_{\varepsilon=-\Delta/2}^{\varepsilon=+\Delta/2}} \\ &= \frac{\Delta}{\sqrt{12}}\end{aligned}$$

In addition, we may assume that the quantization error behaves like a white-noise signal. This means the noise energy is equally spread over our primary frequency range. The *noise power spectral density* $\mathrm{PSD}_n(\omega)$ on this range therefore equals:

$$\mathrm{PSD}_n(\omega) = \frac{\Delta^2}{12\omega_s} \tag{5.5}$$

In case we only consider a limited number of frequencies (using a Fourier Series or a DFT), we can model the noise using a uniform *noise power spectral mass* function, equaling:

$$\mathrm{PSM}_n(\omega_k) = \frac{\Delta^2}{12N} \tag{5.6}$$

with N the number of spectral components that are being considered.

Having a model for the quantization noise allows studying the effect of this noise on our DSP algorithm. But, keep in mind that the model does not hold in general.

5.5.2.2 Dynamic range

Being the noise floor, the quantization noise puts an effective limit to the dynamic range of any converter. The dynamic range is the ratio between the RMS value of the largest signal one can convert and the RMS value of the noise floor.

The peak-to-peak value of any periodic signal we want to convert is limited by the converter's range. The RMS value of such a signal is often expressed in terms of half the peak-to-peak

value. The relationship between both of them is expressed using the so-called *form* or *loading* factor F:

$$F = \frac{V_{\text{RMS}}}{V_{PTP}/2}$$

E.g., a sinusoid signal $A\sin(\omega_0 t)$ has a half peak-to-peak value equaling A. It's RMS value is $A/\sqrt{2}$. Hence, the form factor equals $1/\sqrt{2}$.

If we consider an N-bit converter, the maximal RMS value of any periodic signal is given by:

$$V_{\text{RMS,max}} = 2^{N-1}\Delta F \tag{5.7}$$

while the RMS value of the quantization noise equals:

$$V_{\text{RMS},q-noise} = \frac{\Delta}{\sqrt{12}} \tag{5.8}$$

Dividing (5.7) by (5.8), leads to an expression for the dynamic range:

$$DR = 2^N\sqrt{3}F$$

Very often, this dynamic range is expressed in decibel:

$$DR_{\text{dB}} = 6,0206N + 4.7712 + 20\log(F)$$

This equation indicates that we gain about 6 dB per extra converter bit.

For a sinusoidal signal this reduces to:

$$DR_{\text{sin,dB}} = 1.7609 + 6,0206N \approx 6N$$

Hence the popular quote: "a 16-bit converter has a dynamic range of about 96 dB".

Remarks

- Don't make the mistake of reversing the story. A dynamic range of 96 dB doesn't imply the noise floor to be 96 dB below the signal.
- The "noise" floor of your converter is also influenced by other mechanisms (e.g., aperture jitter error and nonlinearities in general). Therefore, expect the real dynamic range to be at least 6 dB lower than the theoretical value.
- In practice, you will never want to excite your ADC with the maximum signal. The reason is straightforward: any deviation beyond the range of the converter will cause clipping, causing significant distortion. Therefore, you will rarely be able to exploit the full dynamic range of the converter.

5.5.2.3 Selecting the converter resolution

Quantization (and quantization error) puts a lower bound on the smallest signal that can be converted. We modeled the quantization error as an equivalent noise source. This noise source is adding noise to your signal. However, other noise sources exist in the system:

- noise may be present in your input signal,
- the anti-alias filter adds noise to your input signal,
- your DSP may add digital noise (round-off errors),
- the reconstruction filter adds noise to your input signal.

It does make sense to increase the converter's accuracy (increasing the number of bits) to make the quantization noise smaller. However if the other noise contributions become comparable to the quantization noise, increasing the accuracy further, will only result in a more accurate sampling of the noise, not the signal.

So, awareness of the noise-levels in your DSP system will help you in selecting the correct converter accuracy.

E.g., suppose we need to convert a signal with a range of 750 mV in the band from $0 - 20$ kHz as good as we can, knowing that the double-sided input-referred noise power density (including sampling filters) equals 35 aV^2/Hz. We proceed by making the quantization noise power equal to the in-band noise:

$$\frac{\Delta^2}{12} \approx 2\omega_B \,\mathrm{PSD}$$

Knowing that $\Delta = (x_{\max} - x_{\min})/2^N$ with N the number of conversion bits, we can determine N to be:

$$N \approx \log_2\left(\frac{x_{\max} - x_{\min}}{\sqrt{24\omega_B \,\mathrm{PSD}}}\right)$$

leading to $N \approx 12$.

5.5.2.4 Dithering

Problem

One particular case in which the quantization noise error model does not hold, is when the input is almost constant. Assuming the input is constant, the error will also be constant, and can be as large as $\Delta/2$. The core of the problem is not that our simple quantization noise model is no longer valid, we have a model that is as good: a constant error. However, the problem is that we don't know the value of the constant error. If we would be integrating this signal, the error could highly impact our result.

Another particular case in which the quantization noise error model does not hold, is when the amplitude of a periodic signal is small compared to the resolution of the ADC.

Solution

There is a way to improve this situation: by — believe it or not — adding noise! How come? The problem in both cases is the same: the quantization noise is highly correlated to the original signal. The trick is to bury this correlation with noise. Obviously, the noise needs to be added before quantization, and hence needs to be generated in the analog domain. You might want to rely on analog components to generate the noise (maybe your input amplifier can do the job), or even using a full blown DAC to convert digital pseudo-random noise into an analog signal.

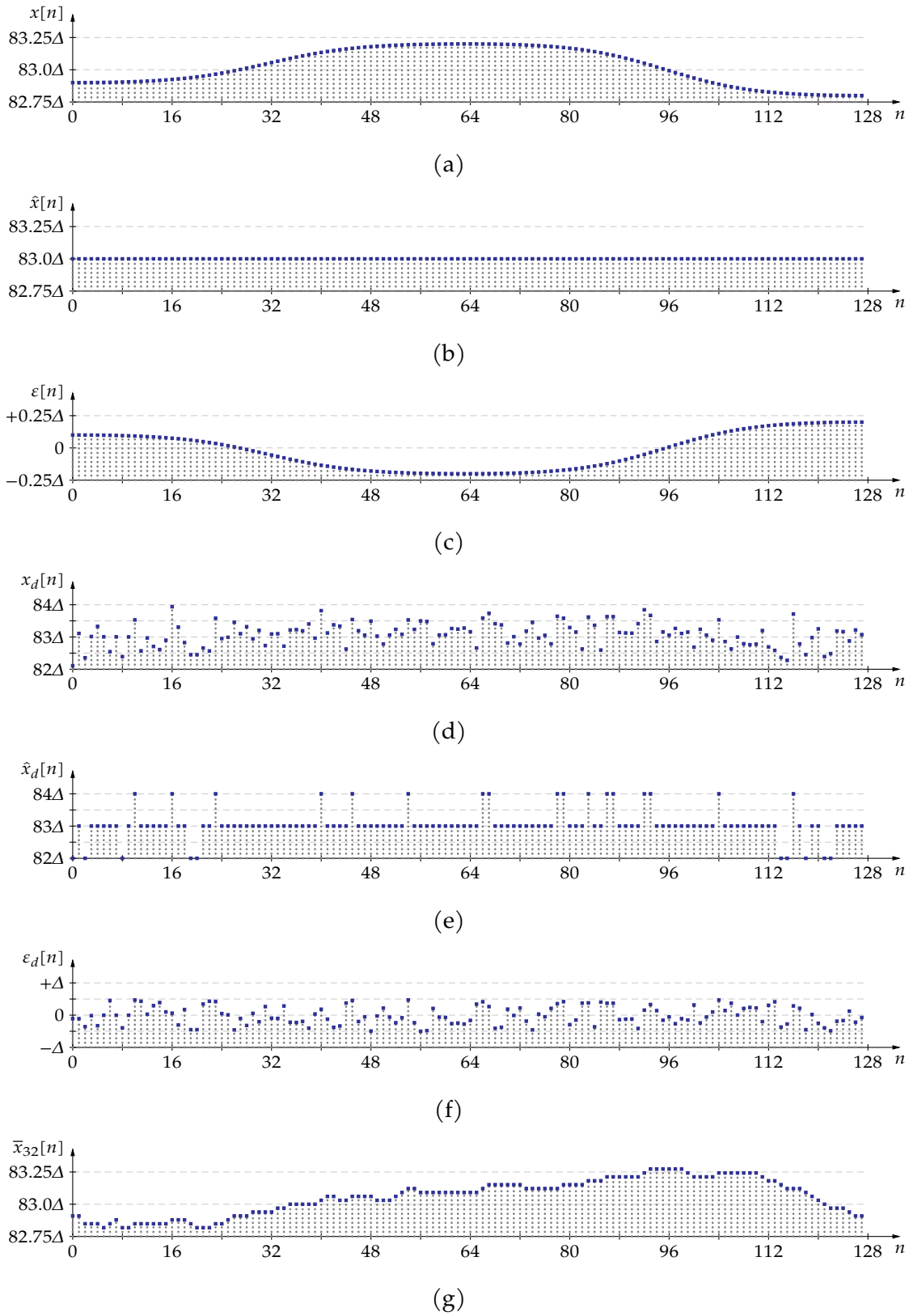


Figure 5.14: Illustration of the appearance of the beneficial effect of dithering noise on the average resolution for slowly varying signals: (a) the original signal, (b) the signal after quantization, (c) the quantization error, (d) the dithered signal, (e) the dithered signal after quantization, (f) the quantization error after dithering, (g) the final result after averaging (m=32).

Our goal is adding enough noise to destroy the correlation while not adding too much noise in order not to completely ruin our signal-to-noise ratio (see section 5.5.3 for a definition of signal-to-noise ratio). Typically, white Gaussian noise with a standard deviation of about 1/3 or more of an LSB is used. The optimal value depends on the application. However, consider 1/3 a good value to start from.

Examples

- **A slowly varying signal**
  Consider the slowly varying signal of Figure 5.14(a). When sampling this signal, all information except for an approximate DC value is lost (Figure 5.14(b)). The error is correlated to the original signal with a correlation factor of -1 (see Figure 5.14(c)).

  Adding Gaussian noise with a standard deviation of 1/3 LSB, results in Figure 5.14(d)). The sampled signal (see Figure 5.14(e)) doesn't really look much better than the original in subfigure (b). In fact, the absolute sample error has increased to a full LSB (see Figure 5.14(f))!

  However, when considering a moving average filtered version of the sampled signal (see Figure 5.14(g)), the meaning of it all becomes clear. The moving average filter (in this case with length 64) filters away the noise, and reconstructs a signal that is closer to the original than before. The averaged signal is delayed because of the moving average filter with length 64.
- **A small cyclostationary signal**
  Consider the sinusoidal signal of Figure 5.15(a). The signal has a rather low amplitude, resulting in a significant quantization error. This can been seen in Figure 5.15(c) and (e). In addition, the quantization error is highly correlated with the original signal, resulting in a number of significant spectral components that fall on the grid of the fundamental frequency of the sinusoid (see Figure 5.15(d) and (f) for the DFTs of these signals). These are so-called *spurious frequencies*.

  We can suppress these spurious frequencies to a large extent by adding dithering noise. The same signal with Gaussian noise added (with standard-deviation of 1/3 LSB) can be found in Figure 5.16(a). The quantized dithered signal and the corresponding error can be found in parts (c) and (e). If you inspect their spectra (DFTs in parts (d) and (f)) you can see that the spurious frequencies are gone at the expense of an increased noise floor. There's no free lunch!

Remarks

- Dithering is also used in pictures. Using a color palette that is too limited may result in edgy pictures. A dithering process (adding noise) and equalizing the noise over a larger region removes the edginess. A typical strategy used for pictures is *Floyd Steinberg dithering*.
- Dithering is based on two principles: *adding noise* and *averaging or integration*. In pictures this is the equalization step. In the examples given above, the DFT is the averaging factor in the process. This also means that if the length of your signals is not sufficient (or the amount of averaging is too small) the result may not be satisfactory.
- Dithering is also a technique that is often used when requantizing a signal with a lower resolution. In this case, dithering may also remove the edginess of the end result.

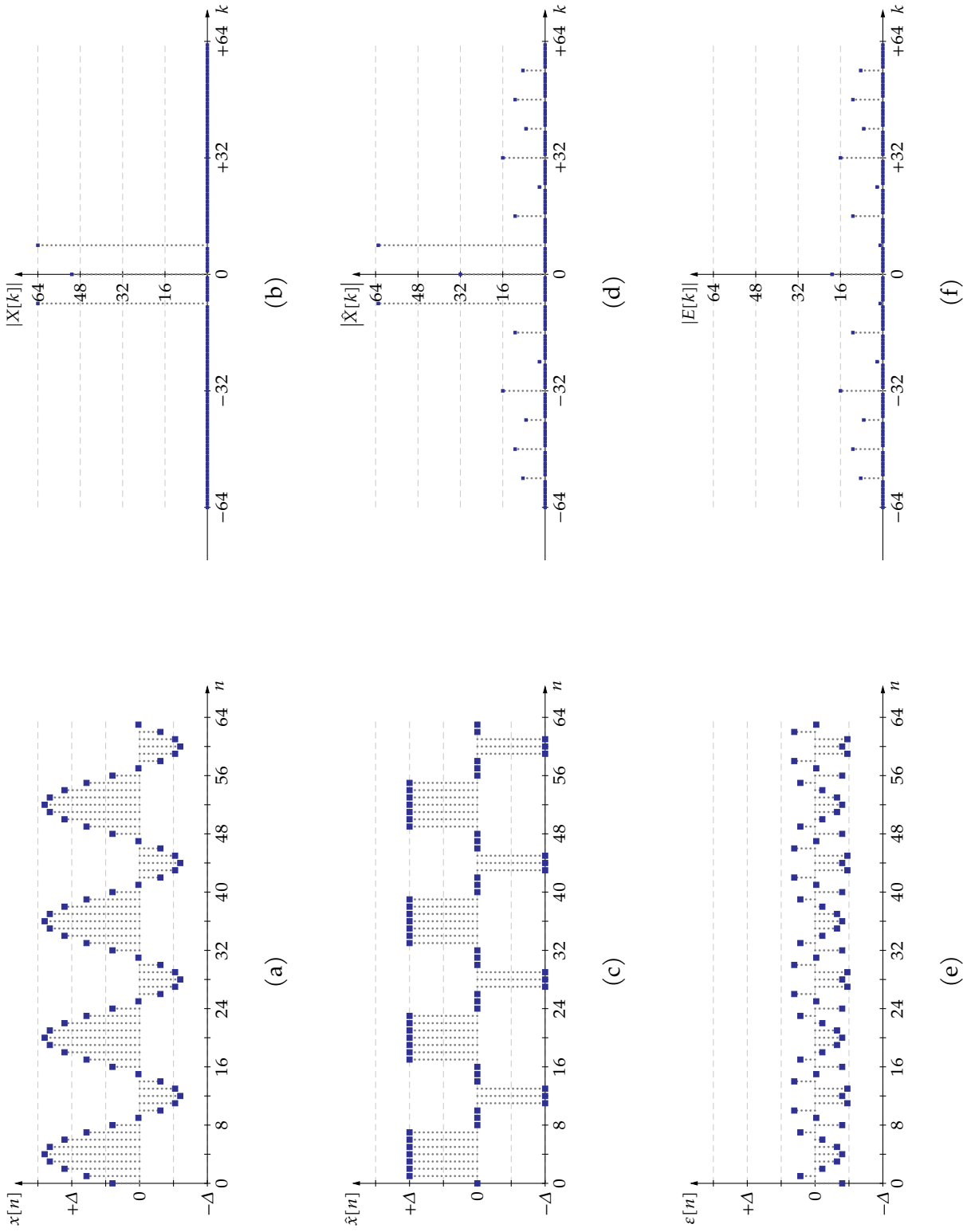


Figure 5.15: Illustration of the appearance of spurious components due to correlated quantization noise: time-domain representation (only 64 points shown) of (a) the original signal, (c) the signal after quantization, (e) the quantization error, and (b,d,f) their respective DFTs.

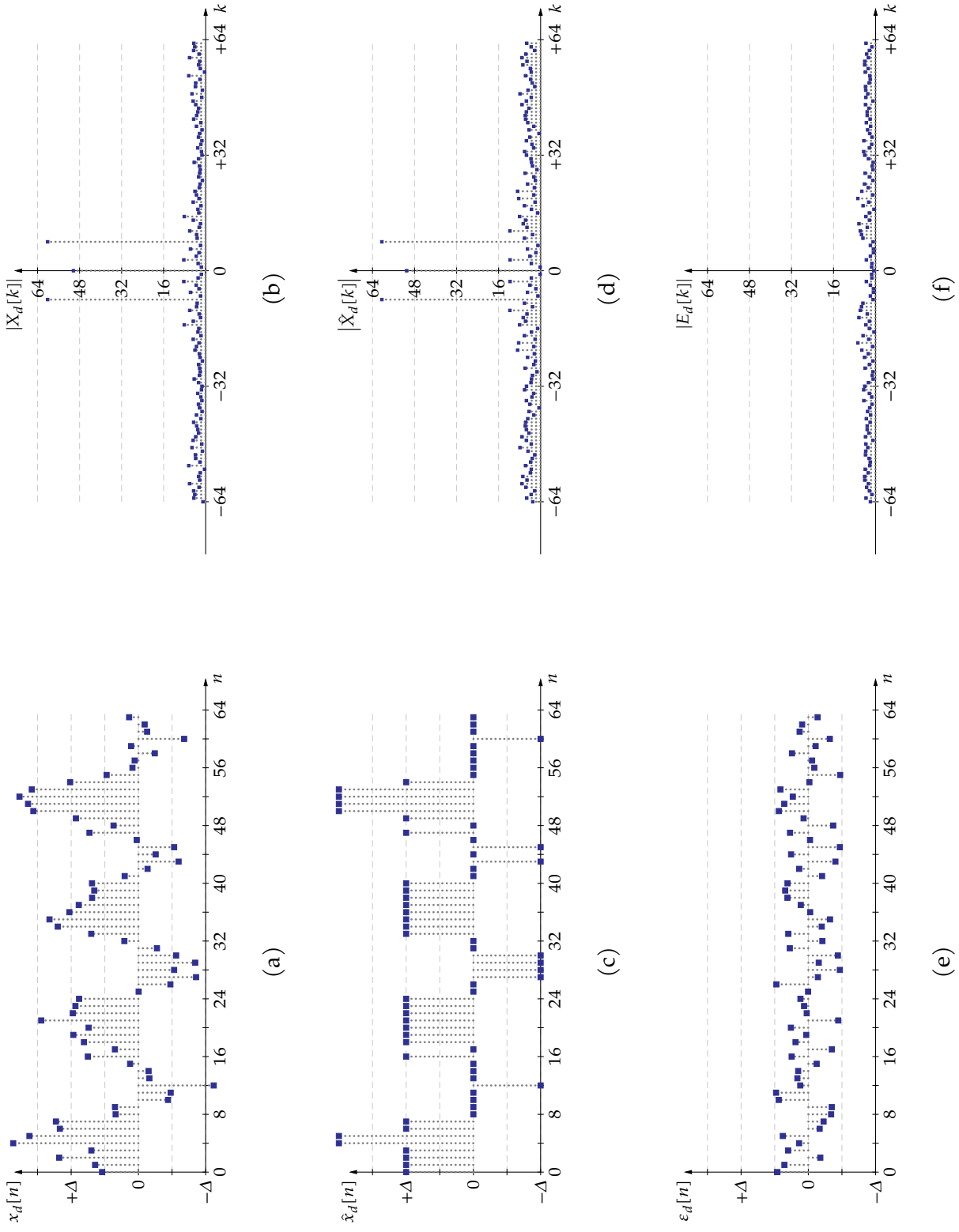


Figure 5.16: Illustration of the influence of dithering (using a Gaussian noise signal with $\sigma = 1/3\text{LSB}$) on the appearance of spurious components due to correlated quantization noise: time-domain representation (only 64 points shown) of (a) the signal with additive dithering noise, (c) the dithered signal after quantization, (e) the quantization error, and (b,d,f) their respective DFTs.

5.5.2.5 Oversampling

Oversampling may also be used as a measure to reduce the impact of quantization. Our uniform quantization error model stated we could model the error as a white-noise source (with a constant noise power density function).

If we oversample the signal by a factor of M, this means according to (5.5) (and equally (5.6)) that we just spread the same quantization energy over an M-times wider primary signal range.

Using a sharp digital filter removing all but the in-band frequencies thus allows reducing the in-band noise by a factor of M.

5.5.3 Dequantizing the signal range

For a moment, take a look back at section 5.5.1 on page 143. In that section, we discussed mapping the continuous range of the input signal $x(t)$ to a quantized value $\hat{x}(t)$, ready for treatment by the DSP. When the DSP finishes processing the signal, it needs to reconstruct the internal quantized outputvalue $\hat{y}(t)$ to the continuous range of the output signal $y(t)$. This again happens in a two-step process. The internal DSP output signal is mapped onto a binary code L_o, that on its turn is converted to a continuous value.

$$\hat{y}(t) \mapsto L_o \mapsto y(t)$$

In practice, the range of $\hat{y}(t)$ does not need to correspond to the range of $y(t)$. However, for our purposes, this assumption makes the treatment easier without restricting the validity of our conclusions below.

Graphically, we can represent this process again with a graph relating $y(t)$ and $\hat{y}(t)$. To allow combining the quantization and the dequantization graph in a single graph, we put $y(t)$ on the abcissa and $\hat{y}(t)$ on the ordinate. You can find the result for a 4-bit converter in Figure 5.17 on the facing page. Again, To prepare ourselves to treat converter nonidealities in a later section we also indicated the *reconstruction levels* r_k on the graph.

Now, the same issues, regarding fixed vs. variable quantization, uniform vs. non-uniform quantization, quantization error, dithering and oversampling can be repeated here. For brevity, we won't.

5.6 ADC and DAC nonidealities related to discretization

Though sampling and quantization introduce important artefacts like frequency replication and quantization noise, our view on ADCs en DACs so far is based on two assumptions that may not hold perfectly:

- the quantization step Δ is constant over the entire converter range
- the sample-and-hold circuit and the zero-th order hold circuit are clocked using a jitterless clock.

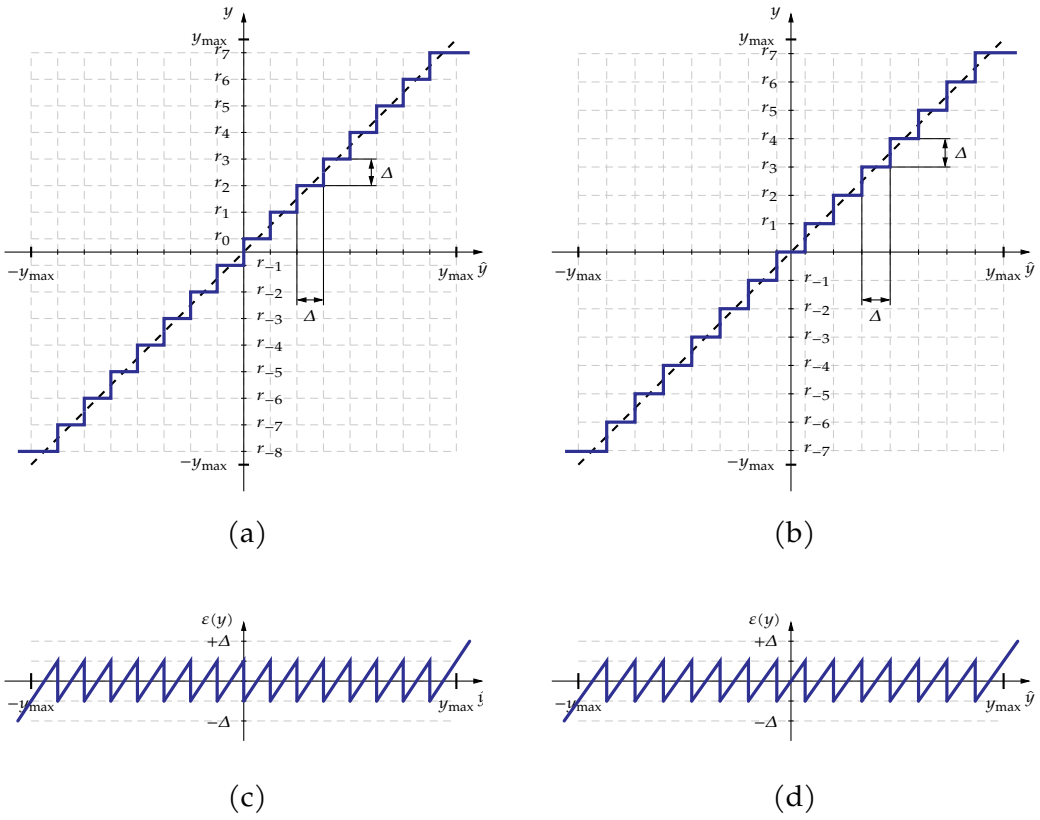


Figure 5.17: Illustration of mid-rise (a) dequantization and (c) dequantization error, and mid-tread (b) dequantization and (d) dequantization error, for a 4bit dequantizer.

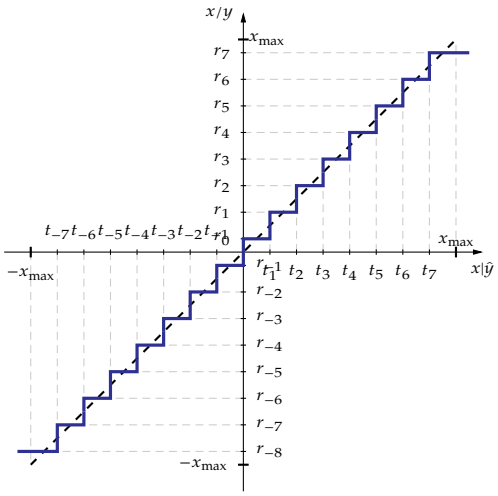


Figure 5.18: Combined quantization and dequantization graph

Several parameters are used to characterize these nonidealities. The values of these parameters will typically appear in the datasheet of a converter.

In the parameter definitions given below, the index boundaries of minimum, maximum and summation operators are not always explicitly given. We assume the indices to be valid. Deriving the correct boundaries yourself should pose no problem to you.

5.6.1 Static performance

Static nonidealities are (by definition) not frequency dependent, and can be measured using a (quasi-) static measurement setup. We'll treat these nonidealities from the perspective of a uniform 2^N mid-rise quantization and dequantization as depicted (for an ideal convertor) in Figure 5.18 for $N = 4$. Notice how we combined the quantization graph (abcissa) and the dequantization graph (ordinate) into a single graph. The ideal quantization and dequantization line has been indicated using a black dashed line. The quantization levels t_k and reconstruction levels r_k have been indicated. The main problem we are treating here, is that these levels in reality are not as ideal as their theoretical values (for the quantizationlevels, see (5.4)).

We can distinguish several static performance parameters. We will define them for the case of ADC (using the t_k levels. To obtain the performance parameters for a DAC, just replace the t_k levels in the definitions with the corresponding r_k.

Offset

The offset is the average deviation of the real quantization line w.r.t. to the ideal (theoretical) one.

$$E_{offset,\text{ADC}} = \frac{\sum_k (t_k - k\Delta)}{2^N - 1}$$
$$E_{offset,\text{DAC}} = \frac{\sum_k (r_k - (k + 0.5)\Delta)}{2^N}$$

Determining this offset can be useful to apply offset-correction (either electronically by tuning

the ADC control/reference voltages, or numerically by taking it into account in your DSP):

$$t_{k,o} = t_k - E_{offset,\text{ADC}}$$
$$r_{k,o} = r_k - E_{offset,\text{DAC}}$$

Full-scale gain error

The full-scale gain error is the deviation of the real converter range w.r.t. to the ideal (theoretical) one.

$$E_{FSG,\text{ADC}} = t_{2^{N-1}-1} - t_{-2^{N-1}+1} - (2^N - 2)\Delta$$
$$E_{FSG,\text{DAC}} = r_{2^{N-1}-1} - r_{-2^{N-1}} - (2^N - 1)\Delta$$

Determining the full-scale gain error can be useful to do some full-scale gain correction (either electronically by tuning the DAC control/reference voltages, or numerically by taking it into account in your DSP):

$$t_{k,g} = t_k - k\frac{E_{FSG,\text{ADC}}}{2^N - 2}$$
$$r_{k,g} = r_k - (k + 0.5)\frac{E_{FSG,\text{DAC}}}{2^N - 1}$$

Usually, one performs both offset and gain correction simultaneously. Please note that offset correction does not influence the gain error. It is therefore safe to determine both the offset and the full-scale gain error on the original data and then perform the correction based on these values[7] according to:

$$t_{k,g\&o} = t_k - E_{offset,\text{ADC}} - k\frac{E_{FSG,\text{ADC}}}{2^N - 2}$$
$$r_{k,g\&o} = r_k - E_{offset,\text{DAC}} - (k + 0.5)\frac{E_{FSG,\text{DAC}}}{2^N - 1}$$

Differential nonlinearity (DNL)

The DNL of a converter (expressed in (a fractional) number of bits) measures the maximum deviation of any quantization step w.r.t. to the ideal step.

$$\text{DNL}_{\text{ADC}} = \frac{\max_k |t_k - t_{k-1} - \Delta|}{\Delta}$$
$$\text{DNL}_{\text{DAC}} = \frac{\max_k |r_k - r_{k-1} - \Delta|}{\Delta}$$

In most cases the DNL is calculated after offset and gain correction. The reason for this is that gain correction can be done by "simple" calibration/tuning (in software or in hardware).

Integral nonlinearity (INL)

The INL of a converter (expressed in (a fractional) number of bits) measures the maximum cumulative deviation of the quantization steps w.r.t. to the ideal quantization line.

$$\text{INL}_{\text{ADC}} = \frac{\max_k(t_k - k\Delta) - \min_k(t_k - k\Delta)}{\Delta}$$
$$\text{INL}_{\text{DAC}} = \frac{\max_k(r_k - (k + 0.5)\Delta) - \min_k(r_k - (k + 0.5)\Delta)}{\Delta}$$

[7] However, please note that gain correction does influence the offset calculation. Therefore, one must apply offset correction before gain correction. If not, the offset must be calculated after gain correction.

In most cases the INL is calculated after offset and gain correction. The reason for this is that this correction can be done by "simple" calibration/tuning (in software or in hardware).

Exercises

Some exercises on static ADC and DAC performance paramters

Exercise 5.6.1-1: Consider a 3-bit mid-rise ADC quantizing the range $[-1\,\text{V}, 1\,\text{V}]$, with the following quantization levels:

Level	Value
t_{-3}	-0.80
t_{-2}	-0.45
t_{-1}	-0.25
t_0	-0.01
t_{+1}	$+0.23$
t_{+2}	$+0.49$
t_{+3}	$+0.80$

Calculate the offset and the full-scale gain error.
Hint: determine the ideal levels, first.

Exercise 5.6.1-2: Calculate the the DNL and INL of the ADC of the previous exercise (after gain correction).

Exercise 5.6.1-3: Consider a 3-bit mid-rise DAC dequantizing the range $[-1\,\text{V}, 1\,\text{V}]$, with the following reconstruction levels:

Level	Value
r_{-4}	-0.865
r_{-3}	-0.611
r_{-2}	-0.372
r_{-1}	-0.131
r_0	0.128
r_{+1}	0.384
r_{+2}	0.622
r_{+3}	0.886

Calculate the offset and the full-scale gain error.
Hint: determine the ideal levels, first.

Exercise 5.6.1-4: Calculate the the DNL and INL of the ADC of the previous exercise (after gain correction).

Exercise 5.6.1-5: (*) Compose an excel spreadsheet to analyze a 4-bit mid-rise ADC w.r.t. offset, full-scale gain error, and DNL and INL (after gain correction).
Hint: make the exercises above, first, and then try to organize your calculations in tabular format.

Exercise 5.6.1-6: (*) Compose an excel spreadsheet to analyze a 4-bit mid-rise DAC w.r.t. offset, full-scale gain error, and DNL and INL (after gain correction).
Hint: make the exercises above, first, and then try to organize your calculations in tabular format.

Exercise 5.6.1-7: (**) Compose an Octave script to analyze an N-bit (with N arbitrary) ADC or DAC w.r.t. offset, full-scale gain error, and DNL and INL (after gain correction).

5.6.2 Dynamic performance

Dynamic nonidealities are frequency-related. They require a periodic signal source in the measurement setup. The frequency often impacts the result, so the test-frequency is an essential piece of the specification.

We can distinguish a number of dynamic performance parameters. For all these parameters we opted to use 'power'-definitions, hence the squared voltages. In some textbooks and/or datasheets you will find values that are voltage based. Check which definition is used, before you blindly use the numbers.

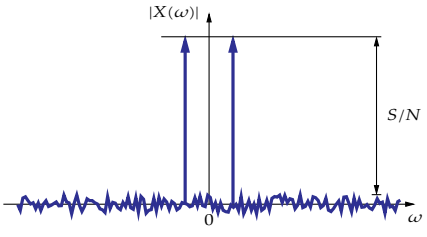
Applying an arbitrary periodic waveform to the converter, the *signal-to-noise ratio* (*SNR*) is the ratio of the RMS value of the input signal to the RMS value of the noise floor (not considering harmonic distortion).

$$\mathrm{S/N} = \frac{V^2_{\mathrm{RMS},signal}}{V^2_{\mathrm{RMS},noise}}$$

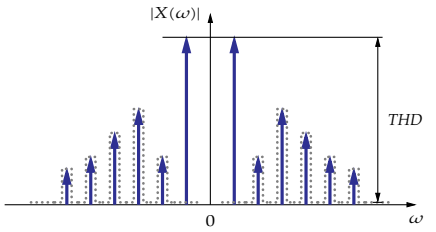
Very often this value is expressed in decibels:

$$\mathrm{S/N_{dB}} = 20\log\left(\frac{V_{\mathrm{RMS},signal}}{V_{\mathrm{RMS},noise}}\right)$$

Don't be mistaken by the "artistic" picture on the right. The RMS values need to be considered, and not the spectral magnitudes.



When we apply a sinusoidal input signal to the converter, the *total harmonic distortion* is defined as the ratio of the RMS value of all harmonic components to the RMS value of the input signal.

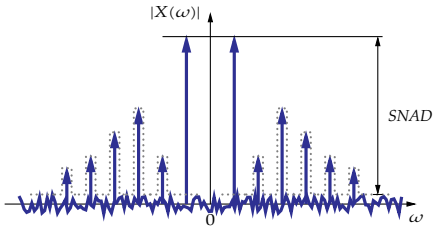


$$\text{THD} = \frac{\sum_{i=2}^{+\infty} V_{\text{RMS},i}^2}{V_{\text{RMS},signal}^2}$$

This value is typically expressed as a percentage. Because all signals appearing in the equation are sinusoids (and therefore all of them have the same load-factor), we can express this parameter also using spectral magnitudes:

$$\text{THD} = \frac{\sum_{i=2}^{+\infty} V_i^2}{V_{signal}^2}$$

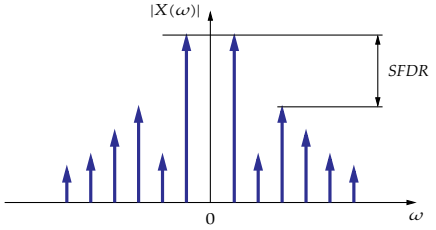
Applying a sinusoidal input signal to the converter and considering both distortion and noise leads to the *signal-to-noise-and-distortion ratio*, relating the original signal to all unwanted components:



$$\text{SNAD} = \frac{V_{\text{RMS},signal}^2}{V_{\text{RMS},noise}^2 + \sum_{i=2}^{+\infty} V_{\text{RMS},i}^2}$$

Again: don't be mistaken by the "artistic" picture on the right. The RMS values need to be considered, and not the spectral magnitudes.

If we apply a sinusoidal input signal to the converter, we can measure the largest spectral harmonic. The *spurious free dynamic range* is defined as the ratio spectral magnitude of the input signal to the largest harmonic:



$$\text{SFDR} = \frac{V_{signal}^2}{\max_{i\geq 2} V_i^2}$$

This parameter is often expressed in decibels:

$$\text{SFDR} = 20\log\left(\frac{V_{signal}}{\max_{i\geq 2} V_i}\right)$$

5.6.3 Timing performance

Aperture delay

The sampling of the input signal does not occur instantaneous on the triggering clock-edge:

there is a little delay. This is the so-called *aperture delay*.[8]

Clockjitter / Aperture error

So far, we assumed our clock signal to be perfectly periodical. However, a typical clock signal has a little jitter on the clock edges (i.e. a variation on the timing). In addition, the apperture delay of a converter is not perfectly stable, and may vary a little bit.

The combination of both causes an estimation error due to early and late clock edges as indicated in the table below:

	early clock	late clock
rising signal	underestimation	overestimation
falling signal	overestimation	underestimation

Consider a signal $x(t)$ sampled using a clock $kT_c + \varepsilon_{T_c}[k]$. We can approximate the error $\varepsilon_x[k]$ using a first-order model:

$$\varepsilon_x[k] \approx \left.\frac{dx(t)}{dt}\right|_{t=kT_c} \varepsilon_{T_c}[k]$$

This error can be considered to be an additional noise source. However the error magnitude is dependent on the signal's frequency content and the amount of clock jitter and aperture delay. This equation also allows to calculate an estimate for the maximum allowable clock jitter given the maximum tolerable aperture error and the maximum slope of the signal.

E.g., let's calculate how much clock jitter we can allow if we want to have a maximum of 1 bit aperture error for a 16 bit hifi audio converter sampling a full-range 20 kHz sinus at 44.1 kHz. The signal can be written as:

$$x(t) = 2^{15}\sin\left(2\pi 20 \times 10^3 t\right)$$

The maximum slope equals

$$\max\left(\frac{dx(t)}{dt}\right) = 2^{16}\pi 20 \times 10^3$$

The maximum tolerable clock jitter, therefore can be estimated to be:

$$\varepsilon_{T_c}[k] \approx \frac{1}{2^{16}\pi 20 \times 10^3} = 0.243\,\text{ns}$$

This is quite a jitter spec on a clock with a period of 227 µs. Luckily, in practice, the tolerable error is related to the magnitude of the signal. Besides, we will rarely meet an audio signal with a full range sinusoid at 20 kHz. Therefore the spec will not be that stringent in practice. However, the key point to remember is that clock jitter and aperture error limit the accuracy of our AD and DA conversion processes.

[8] The sampling of a signal can be compared to taking pictures using a camera. The shutter needs to open and close on every sampling instance. This opening and closing is also referred to as the "aperture".

Chapter 6

The Z-transform

In this chapter, you will learn about:

- the Z-transform, and how it is relelated to the Discrete-time Fourier Transform,
- why we need this transform,
- its properties,
- its stability, and finally
- the extended Fourier family diagram.

After having read this chapter, some questions will still be left unanswered:

- how do I describe systems using these transforms?
- how do I design such systems?

After having read/studied this chapter, you are expected to be able to

- calculate any (forward or inverse) Z-transform given a signal or a spectrum,
- apply the transform properties,
- understand and compose pole-zero diagrams,
- assess the stability of systems based on a pole-zero diagram,
- solve finite difference equations using the Z-transform,
- reproduce (and find your way) in the extended Fourier family diagram.

6.1 From Discrete-time Fourier transform to Z-transform...

The Discrete-time Fourier transform was defined in 3.1 on page 38 as:[1]

$$X_p(\omega) = \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}\omega n T_s}$$
$$x[n] = \frac{1}{\omega_s} \int_{-\frac{\omega_s}{2}}^{+\frac{\omega_s}{2}} X_p(\omega)\, \mathrm{e}^{\mathrm{j}\omega n T_s}\, \mathrm{d}\omega$$

[1] Remember, the subscript p has been added to X_p to emphasize that it is a periodic spectrum.

The discrete-time Fourier transform of a discrete-time signal is a decomposition of the signal in terms of a set of orthonormal base functions. The analysis equation (3.3) can be seen as the projection equation which projects the signal "vector" onto the set of base "vectors", or stated differently, as the equation that computes the correlation between the signal "vector" and the base "vectors".

The entire theory, of course, only holds if the integral and the sum in these equations converge. For a whole lot of cases (e.g., time-limited signals) they do converge. However, in a whole lot of other cases these integral and sum do *not* converge: e.g., for sinusoidal signals (or any other periodic function) or exponential signals.

Similarly as we did for the (continuous-time) Fourier transform, an obvious idea is to give convergence a hand by adding a convergence factor to $x[n]$. A very good choice, to this end, is adding an exponential convergence factor $\mathrm{e}^{-\sigma n T_s}$ (with $\sigma \in \mathbb{R}$), defining a new $x_c[n]$ as

$$x_c[n] = x[n]\,\mathrm{e}^{-\sigma n T_s}$$

In this way, the discrete-time Fourier transform relating $x_c[n] \overset{\mathcal{F}}{\longrightarrow} X_{c,p}(\omega)$ can be written as:

$$x[n]\,\mathrm{e}^{-\sigma n T_s} = \frac{1}{\omega_s}\int_{-\frac{\omega_s}{2}}^{+\frac{\omega_s}{2}} X_{c,p}(\omega)\,\mathrm{e}^{\mathrm{j}\omega n T_s}\,\mathrm{d}\omega \tag{6.3}$$

$$X_{c,p}(\omega) = \sum_{n=-\infty}^{+\infty} x[n]\,\mathrm{e}^{-\sigma n T_s}\,\mathrm{e}^{-\mathrm{j}\omega n T_s}$$

Now,

1. considering a fixed σ,
2. substituting $z = \mathrm{e}^{(\sigma+\mathrm{j}\omega)T_s}$,
3. realizing that $X_{c,p}(\omega)$ also is a function of σ, so we'd better denote is as $X(z)$, and
4. rearranging (6.3) a little bit,

we obtain the well known

Z-transform

Given a time-domain signal $x[n]$, we can calculate a frequency domain representation $X(z)$, the Z-transform of the signal, from which the original time-domain signal can be recovered by inverse transformation:

$$x[n] = \frac{1}{2\pi\mathrm{j}}\int_{\mathrm{e}^{\sigma T_s - \mathrm{j}\pi}}^{\mathrm{e}^{\sigma T_s + \mathrm{j}\pi}} X(z) z^{n-1} dz$$

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

The integral appearing in this definition is a contour integral integrating over a counterclockwise circle C with radius $\mathrm{e}^{\sigma T_s}$ in the Z-plane:

$$x[n] = \frac{1}{2\pi\mathrm{j}}\oint_C X(z) z^{n-1} dz$$

In order to ensure convergence σ must be chosen such that the circle C encloses all poles of $X(z)$.

In this way, the Z-transform checks for every value of σ the correlation between the exponentially modulated signal $x[n]\,\mathrm{e}^{-\sigma n T_s}$ and complex exponentials $\mathrm{e}^{-\mathrm{j}\omega n T_s}$ in the same way as the discrete-time Fourier transform does. However, it is no longer an orthogonal signal decomposition.

Remarks

- The Z-transform is often symbolized as an operator $\mathcal{Z}$.

$$X(z) = \mathcal{Z}(x(t)) \qquad\qquad x(t) = \mathcal{Z}^{-1}(X(z))$$

- The use of lowercase symbols for time-domain signals and uppercase symbols for their Z-transform is quite common. The pair is also often written as follows:

$$x(t) \overset{\mathcal{Z}}{\rightarrow} X(z)$$

- Note how simple the Z-transform is. Transforming a time sequence just requires multiplying the samples a_k with an appropriate power of z, i.e. z^{-k}.
- Though the inverse transform seems a whole lot more complicated, a simple strategy is just to try recovering the form $\sum_k a_k z^{-k}$ from the Z-transformed signal and just read the sample values a_k.

6.2 ...and back

In case the integrals above converge for $\sigma = 0$, these equations reduce to the original discrete-time Fourier transform equations. It is therefore (under these circumstances) very simple to derive the discrete-time Fourier transform (and hence the frequency spectrum it represents) from the Z-transform: just set $\sigma = 0$ in the Z-transform, i.e. substitute $z = \mathrm{e}^{\mathrm{j}\omega T_s}$.

Example We can illustrate this by considering as example the transform pair:

$$x[n] = \begin{cases} 1 & \text{if } |n| \leq 1 \\ 0 & \text{if } |n| > 1 \end{cases} \overset{\mathcal{Z}}{\rightarrow} X(z) = z^1 + 1 + z^{-1} = \frac{z^2 + z + 1}{z} \tag{6.4}$$

For $\sigma = 0$ the Z-transform summation still converges. Therefore the Fourier transform can easily be derived by replacing z by $\mathrm{e}^{\mathrm{j}\omega T_s}$

$$\begin{aligned} X_p(\omega) &= 1 + \mathrm{e}^{\mathrm{j}\omega T_s} + \mathrm{e}^{-\mathrm{j}\omega T_s} \\ &= 1 + 2\cos(\omega T_s) = 1 + 2\cos\left(2\pi\frac{\omega}{\omega_s}\right) \end{aligned}$$

In view of the substitution of z by $\mathrm{e}^{\mathrm{j}\omega T_s}$, and in view of the fact that often T_s is arbitrarily set to 1, very often, the discrete-time Fourier transform is denoted as $X(\mathrm{e}^{\mathrm{j}\omega})$ instead of $X_p(\omega)$.

6.3 The one-sided Z-transform

Very often, we only consider causal signals (i.e. zero for negative times). In that case we obtain the

One-sided Z-transform
Given a causal time-domain signal $x[n]$, we can calculate a frequency domain representation $X(z)$, the Z-transform of the signal, from which the original time-domain signal can be recovered by inverse transformation:

$$x[n] = \frac{1}{2\pi \mathrm{j}} \int_{\mathrm{e}^{\sigma T_s - \mathrm{j}\pi}}^{\mathrm{e}^{\sigma T_s + \mathrm{j}\pi}} X(z) z^{n-1} dz \tag{6.5}$$

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

In most cases the one-sided Z-transform is simply labeled as *the* Z-transform, while the two-sided version is explicitly described as the *two-sided* or *bilateral* Z-transform.

6.4 Region of convergence

Our definition of the Z-transform has the form of a series. The question rises whether this series converges or not. For a generic causal signal $x[n]$ that is exponentially bounded, we can use the following lemma:

Lemma: convergence of the Z-transform
For any signal $x[n]$ that is

1. causal,
2. bounded for $n \in [0, N]$, and
3. exponentially bounded otherwise, i.e.

$$\exists M \in \mathbb{R}, \exists N \in \mathbb{N}, \forall n > N : |x[n]| < M^n,$$

the corresponding Z-transform $X(z) = \mathcal{Z}(x[n])$ converges for $|z| > M$.

Proof
We take a start with the definition of the Z-transform:

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

We can split the summation in two parts:

$$X(z) = \underbrace{\sum_{n=0}^{N} x[n] z^{-n}}_{X_1(z)} + \underbrace{\sum_{n=N+1}^{+\infty} x[n] z^{-n}}_{X_2(z)}$$

The first part $(X_1(z))$ is a finite sum of finite (complex) numbers, and therefore is finite.

The second part

$$X_2(z) = \sum_{n=N+1}^{+\infty} x[n] z^{-n}$$

is elaborated below.

Taking the absolute value of both sides of the equation yields:

$$\begin{aligned} |X_2(z)| &= \left| \sum_{n=N+1}^{+\infty} x[n] z^{-n} \right| \\ &\leq \sum_{n=N+1}^{+\infty} |x[n]|\, |z^{-n}| \\ &< \sum_{n=N+1}^{+\infty} M^n\, |z^{-n}| \end{aligned} \tag{6.6}$$

This is a geometric series with ratio $q = \frac{M}{|z|}$. Convergence is guaranteed if:

$$q = \frac{M}{|z|} < 1 \iff |z| > M$$

If (6.6) converges, then also $|X(z)|$ is bounded and therefore $X(z)$ converges. ■

6.5 Properties

From the definition of the Z-transform a number of properties can easily be derived.

Let's start by making the following assumptions:

$$\begin{gathered} x[n] \xrightarrow{Z} X(z) \\ y[n] \xrightarrow{Z} Y(z) \\ a, b \in \mathbb{R} \\ n_0 \in \mathbb{N}_0 \end{gathered}$$

Linearity

$$ax[n] + by[n] \xrightarrow{Z} aX(z) + bY(z)$$

Z-scaling

$$a^n x[n] \xrightarrow{Z} X\left(\frac{z}{a}\right)$$

Time shifting

$$\begin{aligned} x[n - n_0] &\xrightarrow{Z} z^{-n_0} X(z) \\ x[n + n_0] &\xrightarrow{Z} z^{n_0} \left(X(z) - \sum_{i=0}^{n_0 - 1} x[i] z^{-i} \right) \end{aligned}$$

Time reversal (for two-sided Z-transform)

$$x[-n] \xrightarrow{Z} X(z^{-1})$$

Time difference

$$x[n] - x[n-n_0] \xrightarrow{Z} \frac{z^{n_0}-1}{z^{n_0}} X(z)$$

Z-Differentiation

$$nx[n] \xrightarrow{Z} -z\frac{dX(z)}{dz}$$

Summation

$$\sum_{i=0}^{n_0} x[n-i] \xrightarrow{Z} \frac{z-z^{-n_0}}{z-1} X(z)$$
$$\sum_{i=0}^{+\infty} x[n-i] \xrightarrow{Z} \frac{z}{z-1} X(z)$$

Convolution

$$x[n] \star y[n] \xrightarrow{Z} X(z)Y(z)$$

Initial/Final value theorem

$$x[0] = \lim_{z\to\infty} X(z)$$
$$\lim_{n\to+\infty} x[n] = \lim_{z\to 1} (z-1)\, X\,(z)$$

Arbitrary value theorem

$$x[n] = \lim_{z\to\infty} z^n \left(X(z) - \sum_{i=0}^{n-1} x[i] z^{-i} \right)$$

Exercises

Exercise 6.5-1: Prove the property of Z-scaling.

Exercise 6.5-2: Prove the property of Time shifting.

Exercise 6.5-3: Prove the property of Summation.

Exercise 6.5-4: Prove the property of Time difference.

Exercise 6.5-5: Derive a property to calculate $x[n+n_0]-x[n]$, given $x[n] \xrightarrow{Z} X(z)$.

6.6 Practical ways to calculate the (inverse) Z-transform

6.6.1 Forward

Writing down the Z-transform of a causal signal $x[n]$ is most easy. Just multiply the value of the signal at every time point by the appropriate factor z^{-n} and add them up:

$$X(z) = x[0] + x[1]z^{-1} + x[2]z^{-2} + x[3]z^{-3} \ldots$$

The disadvantage of this procedure is that the end result is an open expression (i.e. an expression that doesn't terminate).

Alas, there is no procedure that guarantees a closed form expression. Being creative with general knowledge about series, and the properties of Z-transform is the only way to go.

Example 1 Compose the Z-transform of $x[n] = \mathrm{e}^{an}$.

Obviously:

$$\mathrm{e}^{an} \xrightarrow{Z} X(z) = 1 + \mathrm{e}^{a}z^{-1} + \mathrm{e}^{2a}z^{-2} + \mathrm{e}^{3a}z^{-3} + \mathrm{e}^{4a}z^{-4} + \ldots$$

This is a geometric series with ratio e^{a}/z. If its absolute value is smaller than 1, the series will converge.

$$\left|\frac{\mathrm{e}^{a}}{z}\right| < 1 \Leftrightarrow |\mathrm{e}^{a}| < |z| \Leftrightarrow \mathrm{e}^{<a} < |z|$$

Therefore, in the complex Z-plane outside the circular area with radius $\mathrm{e}^{<a}$, the following Z-transform pair holds:

$$\mathrm{e}^{an} \xrightarrow{Z} X(z) = \frac{1}{1-\mathrm{e}^{a}z^{-1}} = \frac{z}{z-\mathrm{e}^{a}}$$

Example 2 Compose the Z-transform of $x[n] = 1$.

Obviously:

$$1 \overset{\mathcal{Z}}{\rightarrow} X(z) = 1 + z^{-1} + z^{-2} + z^{-3} + z^{-4} + \ldots \tag{6.7}$$

Actually, this is a special case of example 1 (set a to 0). Therefore, if $|z| > 1$, the following Z-transform pair holds:

$$1 \overset{\mathcal{Z}}{\rightarrow} X(z) = \frac{z}{z-1}$$

Example 3 Compose the Z-transform of $x[n] = n$.

Again, obviously:

$$n \overset{\mathcal{Z}}{\rightarrow} X(z) = 0 + 1z^{-1} + 2z^{-2} + 3z^{-3} + 4z^{-4} + \ldots$$

Creative as we are, we propose to rewrite the above as:

$$\begin{aligned} n \overset{\mathcal{Z}}{\rightarrow} X(z) = {} & z^{-1} + z^{-2} + z^{-3} + z^{-4} + \ldots \\ & + z^{-2} + z^{-3} + \ldots \\ & + z^{-3} + z^{-4} + \ldots \\ & + z^{-4} + \ldots \\ & + \ldots \\ = {} & z^{-1}\left(1 + z^{-1} + z^{-2} + z^{-3} + \ldots\right) \\ & + z^{-2}\left(1 + z^{-1} + z^{-2} + z^{-3} + \ldots\right) \\ & + z^{-3}\left(1 + z^{-1} + z^{-2} + z^{-3} + \ldots\right) \\ & + \ldots \end{aligned}$$

Probably you've noticed that the series of example 2 (see (6.7)) appears in the last step, i.e.

$$1 + z^{-1} + z^{-2} + z^{-3} + \ldots = \frac{z}{z-1}$$

Therefore:

$$\begin{aligned} n \overset{\mathcal{Z}}{\rightarrow} X(z) &= z^{-1}\frac{z}{z-1} + z^{-2}\frac{z}{z-1} + z^{-3}\frac{z}{z-1} + \ldots \\ &= \frac{z}{z-1}\left(z^{-1} + z^{-2} + z^{-3} + \ldots\right) \\ &= \frac{z}{z-1}\left(-1 + \underbrace{1 + z^{-1} + z^{-2} + z^{-3} + \ldots}_{\frac{z}{z-1}}\right) \\ &= \frac{z}{z-1}\left(-1 + \frac{z}{z-1}\right) \\ &= \frac{z}{(z-1)^2} \end{aligned}$$

You will appreciate that finding closed form expressions for the Z-transform requires decent mathematical skills and a significant bit of *Fingerspitzengefühl*.

Of course, it makes sense to keep common transform pairs handy in a table, such that we can avoid doing the same calculations over and over again (see section 6.6.2). It turns out that the table gets us quite a way.

Exercises

Exercise 6.6.1-1: Find the Z-transform for the following time-domain signals:

a) $x[n] = (-1)^n$ b) $x[n] = 2^n$ c) $x[n] = 2^{-n}$

You might want to consider $Z(e^{na})$ as a starting point.

Exercise 6.6.1-2: Find the Z-transform for the following time-domain signals:

a) $x[n] = n^2$
b) $x[n] = n^3$
c) $x[n] = n^4$
d) $x[n] = n\,\mathrm{e}^{na}$

Use the property of z-differentation.

6.6.2 Inverse

Where the forward transform ranged from very easy (for non-closed form expressions) to 'not for the faint of heart' (for closed-form expressions), the inverse transform starts out to be almost impossible. Indeed, solving (6.5) is a nightmare[2].

To get around this, in general, we have several options:

1. Substitute $1/z$ by u and compose a McLaurin series around $u = 0$. Then substitute u again by $1/z$, obtaining a Laurent series with only negative exponents. The inversely transformed signal reveals itself as coefficients of the Laurent series. Factors z^p for p integer, can be left out before composing the McLaurin series and reconsidered at the end.
2. Use the list of properties and the table of common transform pairs to make our way
3. Use the arbitrary value property to calculate the successive values of $x[n]$. However, it turns out that the limit calculations are often quite cumbersome.

However, if $X(z)$ is a ratio of two polynomials in z (which in engineering covers over 90% of all cases), there is a simple procedure that we can employ.

Given:

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

It is easy to see that when $x[n] \overset{Z}{\rightarrow} X(z)$ is causal, the following must hold:

$$\text{degree}\,(N(z)) \leq \text{degree}\,(D(z))$$

Procedure:

[2]This statement only holds for engineers; I know a good deal of mathematicians that consider solving this equation to be mouth-watering.

1. Find the highest power p of z that appears in $X(z)$ and multiply numerator $N(z)$ and denominator $D(z)$ by z^{-p}.
2. Perform a long division of $N(z)$ by $D(z)$, for increasing powers of z^i.
3. If possible try to find a closed form expression for the coefficients of the series of the resulting quotient.
4. The successive time-domain values are given by the aforementioned coefficients.

Example Compose the inverse Z-transform of $X(z) = \frac{z^2-3}{z^2+1}$.

Step 1: the highest power is z^2, therefore:

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

Step 2: perform long division:

$$\begin{array}{ccccc|cccccc}
1 & -3z^{-2} & & & & 1 & +z^{-2} & & & & \\
\cline{6-11}
1 & +z^{-2} & & & & 1 & -4z^{-2} & +4z^{-4} & -4z^{-6} & + & \cdots \\
\cline{1-2}
0 & -4z^{-2} & & & & & & & & & \\
 & -4z^{-2} & -4z^{-4} & & & & & & & & \\
\cline{2-3}
 & 0 & +4z^{-4} & & & & & & & & \\
 & & +4z^{-4} & +4z^{-6} & & & & & & & \\
\cline{3-4}
 & & 0 & -4z^{-6} & & & & & & & \\
 & & & -4z^{-6} & -4z^{-8} & & & & & & \\
\cline{4-5}
 & & & 0 & \vdots & & & & & &
\end{array}$$

Step 3: It can be seen from the long division that a pattern starts to emerge:

$$X(z) = \frac{1-3z^{-2}}{1+z^{-2}} = 1 - 4z^{-2} + 4z^{-4} - 4z^{-6} + 4z^{-8} + \ldots$$

Step 4: The coefficients form the successive time-domain values:

$$X(z) \xrightarrow{Z^{-1}} x[n] = [1, 0, -4, 0, 4, 0, -4, 0, 4, 0, -4, \ldots]$$

with more in general, $\forall n \in \mathbb{N}_0$:

$$\begin{aligned}
x[0] &= 1 \\
x[2n-1] &= 0 \\
x[2n] &= (-1)^n 4 \ \text{ for } n \neq 0
\end{aligned}$$

Exercises

Exercise 6.6.2-1: Determine the inverse Z-transform of:

a) $\frac{z}{z^2+1}$

b) $\frac{z^2}{z^2+1}$

c) $\frac{z^3}{z^4-5}$

d) $\frac{2z}{3z-4}$

Exercise 6.6.2-2: (*) Use the method of Laurent-series composition by composing a McLaurin-series around $u = 0$, after subsitution $u = 1/z$, to determine the inverse Z-transformed signals corresponding to:

a) $X(z) = \mathrm{e}^{1/z}$

b) $X(z) = \frac{1}{z}\left(1 + \frac{1}{z}\right)$

c) $X(z) = z \sin\left(\frac{1}{z}\right)$

d) $X(z) = \cosh\frac{1}{z} - 1$

e) $X(z) = \ln\frac{z-1}{z+1}$

Exercise 6.6.2-3: (*) Use the *arbitrary value property* to solve the problems of the previous exercise.

6.7 Common transform pairs

Table 6.1 on page 177 displays the most common transform pairs.[3]

6.8 Why do we need the Z-transform?

It must be said that even though we originally stated having added the convergence factor $\mathrm{e}^{-\sigma n T_s}$ to help convergence, the factor may well be added even if the original discrete-time Fourier transform equations *did* converge.

Convergence or not, the net result is that now we have a transform that maps the original signal to a two-dimensional space. Why would we want this complication if we find ourselves served equally well by the discrete-time Fourier transform?

Example Consider the graphical plots of the Z-transform of the example of (6.4) in Figure 6.1 on the following page.

There are two things to note on these plots:

1. Not so easily seen as in the Laplace case, the discrete-time Fourier transform shows itself on the unit circle $|z| = 1$;
2. apart from this, the plots are not really more insightful than the ones of the "original" discrete-time Fourier transform.

[3] The use of $u[k]$ makes these formula's also valid for the (generic) two-sided transform.

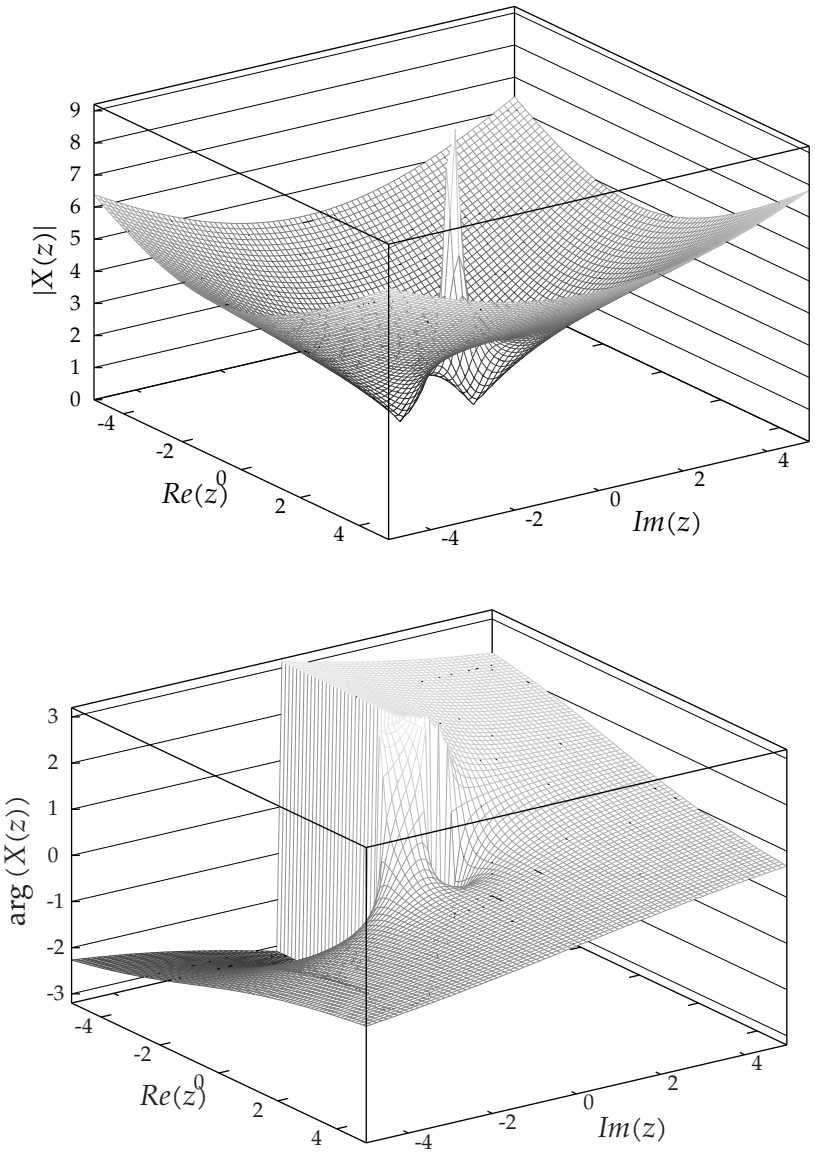


Figure 6.1: Graphical representation of the Z-transform of the example of (6.4): (a) magnitude plot, and (b) phase plot

$x[n]$	$\xrightarrow{\mathcal{Z}}$	$X(z)$
$\delta[n]$	$\xrightarrow{\mathcal{Z}}$	1
$u[n]$	$\xrightarrow{\mathcal{Z}}$	$\frac{z}{z-1}$
$nu[n]$	$\xrightarrow{\mathcal{Z}}$	$\frac{z}{(z-1)^2}$
$n^2u[n]$	$\xrightarrow{\mathcal{Z}}$	$\frac{z(z+1)}{(z-1)^3}$
$n^3u[n]$	$\xrightarrow{\mathcal{Z}}$	$\frac{z(z^2+4z+1)}{(z-1)^4}$
na^n	$\xrightarrow{\mathcal{Z}}$	$\frac{az}{(z-a)^2}$
$a^nu[n]$	$\xrightarrow{\mathcal{Z}}$	$\frac{z}{z-a}$
$\frac{a^n}{n!}$	$\xrightarrow{\mathcal{Z}}$	$\mathrm{e}^{\frac{a}{z}}$
$\sin(an)u[n]$	$\xrightarrow{\mathcal{Z}}$	$\frac{z\sin a}{z^2-2z\cos a+1}$
$\cos(an)u[n]$	$\xrightarrow{\mathcal{Z}}$	$\frac{z(z-\cos a)}{z^2-2z\cos a+1}$
$\sinh(an)u[n]$	$\xrightarrow{\mathcal{Z}}$	$\frac{z\sinh a}{z^2-2z\cosh a+1}$
$\cosh(an)u[n]$	$\xrightarrow{\mathcal{Z}}$	$\frac{z(z-\cosh a)}{z^2-2z\cosh a+1}$

Table 6.1: Common Z-transform pairs

Therefore, once more: why would we put up with this extra complication if there's no benefit for it?

Well, many physical systems can be described using systems of (linear) difference equations. The solutions to these equations can be written in terms of sinusoidal and exponential functions. A particular fact is that the Z-transforms of these types of functions become very simple expressions. This leads to an easy solution of the original system in the Z-domain.

These simple expressions typically are ratios of polynomials in z. The roots of the numerator and the denominator are the well-known zeros and poles of the system.

The poles and zeros of (6.4) can be seen on Figure 6.1 on the preceding page. However, very often, these plots are displayed in a simplified form, as a 2-D pole-zero diagram, as in Figure 6.2.

Typical examples of these are linear discrete systems. In this case the transform is carried out implicitly by describing the digital elements by a Z-transformed function. The following table

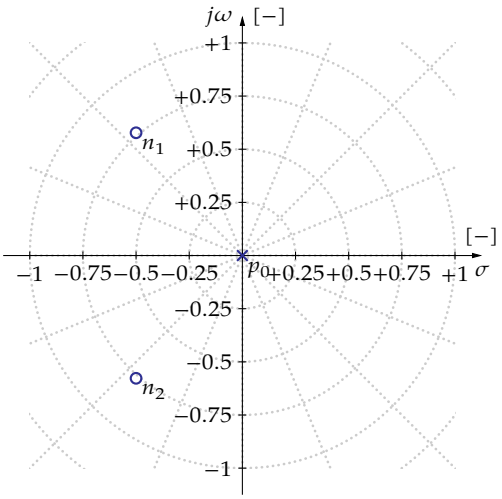


Figure 6.2: Pole-zero diagram of the example of (6.4)

summarizes the mapping for your convenience:

Element	Z-transform
Unit delay	z^{-1}
Amplifier	A

In case the one-sided Z-transform is considered, the initial conditions should also be taken into account.

We won't give an example on this technique right now, as we will see plenty of examples in the next chapters.

Also note that the Z-transform is an analytical transform, not very suited for *numerical* computer treatment.

6.9 Application - solving finite difference equations

One of the major applications for the Z-transform is to solve finite difference equations.

Finite difference equation
A finite difference equation is a relationship of the following type:

$$F(y[n+m], \ldots, y[n+2], y[n+1], y[n], y[n-1], y[n-2], \ldots, y[n-k]) = 0$$

with $k, m \in \mathbb{Z}$, such that $m > -k$.

In this equation $y[n]$ represents a time-discrete signal. Any signal $y[n]$ that results in a true expression is a so-called *solution* of the finite difference equation.

The value of $m + k$ is referred to as the *order* of the finite difference equation.

Example Consider the finite difference equation:

$$y[n] - y[n-1] = 1$$

Obviously $y[n] = n$ is a solution to this equation. However, this is not the only possible solution. Consider for example another possible solution: $y[n] = n + 1$. In fact, any signal of the form $y[n] = n + k$ with $k \in \mathbb{C}$ is a solution.

Finding solutions for a finite difference equation is referred to as *solving* it. A major question is how to solve such a finite difference equations? In fact there is a huge correspondence between differential equations and finite difference equations. We successfully used the Laplace transform to solve (sets of) linear differential equations subject to boundary conditions. Likewise, we will be able to use the Z-transform to solve (sets of) finite difference equations subject to boundary conditions. You will see that the solution strategy is identical to the solution strategy for solving linear differential equations using the Laplace-transform.

Solving finite difference equations using the Z-transform

Consider the finite difference equation:

$$y[n+2] - 2y[n+1] + y[n] = 2$$

subject to $y[0] = -1$ and $y[1] = 0$.

We will attempt to solve this equation by transforming both sides of the equation, using the Z-transform. Keep in mind that by using the Z-transform, we implicitly assume $y[n]$ to be causal!

$$\mathcal{Z}(y[n+2] - 2y[n+1] + y[n]) = \mathcal{Z}(2)$$

$$\downarrow \text{linearity of the Z-transform}$$

$$\mathcal{Z}(y[n+2]) - 2\mathcal{Z}(y[n+1]) + \mathcal{Z}(y[n]) = \mathcal{Z}(2)$$

Let's set $\mathcal{Z}(y[n]) = Y(z)$ and calculate some of the terms involved:

$$\begin{aligned}
\mathcal{Z}(y[n+1]) &= z\,(Y(z) - y[0]) \\
\mathcal{Z}(y[n+2]) &= z\,(z\,(Y(z) - y[0]) - y[1]) = z^2Y(z) - z^2y[0] - zy[1] \\
\mathcal{Z}(2) &= 2\frac{z}{z-1}
\end{aligned}$$

Therefore:

$$z^2Y(z) - z^2y[0] - zy[1] - 2(z\,(Y(z) - y[0])) + Y(z) = \frac{2z}{z-1}$$

$$\downarrow \text{fill in boundary conditions}$$

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

Solving for $Y(z)$ yields:

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

$$Y(z) = \frac{-z^3 + 3z^2}{(z-1)^3}$$

Now, we can determine $y[n]$ by inverse Z-transform. To this end, we use the procedure for determining the inverse Z-transform for ratios of polynomials in z.

$$\begin{aligned} Y(z) &= \frac{z^{-3}}{z^{-3}} \frac{-z^3 + 3z^2}{(z-1)^3} \\ &= \frac{-1 + 3z^{-1}}{1 - 3z^{-1} + 3z^{-2} - z^{-3}} \\ &\downarrow \text{ long division} \\ &= -1 + 3z^{-2} + 8z^{-3} + 15z^{-4} + 24z^{-5} + 35z^{-6} + \ldots + (n^2-1)z^{-n} + \ldots \\ y[n] &= n^2 - 1 \end{aligned}$$

Note that the general expression for the n-th coefficient was determined based on gut feeling, and therefore needs confirmation. We can easily get this confirmation by calculating the Z-transform of $y[n] = n^2 - 1$.

$$\begin{aligned} \mathcal{Z}(n^2 - 1) &= \mathcal{Z}(n^2) - \mathcal{Z}(1) \\ &= \frac{z(z+1)}{(z-1)^3} - \frac{z}{z-1} \\ &= \frac{-z^3 + 3z^2}{(z-1)^3} \end{aligned}$$

This confirms our gut feeling rock solid.

Exercises

Exercise 6.9-1: Solve the following finite difference equations, subject to the listed boundary conditions:

a)	$y[n+2] + y[n] = 1$	with $y[0] = 0$	and $y[1] = 1$	
b)	$y[n+2] - y[n+1] = 2n + 3$	with $y[0] = 1$	and $y[1] = 2$	
c)	$2y[n+2] - y[n+1] - y[n] = 9$	with $y[0] = 1$	and $y[1] = 4$	
d)	$4y[n+2] + 4y[n+1] = 3y[n]$	with $y[0] = 12$	and $y[1] = 6$	
e)	$(n+3)y[n+3] = 3y[n+2]$	with $y[0] = 12$	and $y[1] = 6$	and $y[2] = 9$
f)	$ny[n+1] = (n+1)y[n]$	with $y[0] = 0$	and $y[1] = 3$	
g)	$(n+3)y[n+1] - y[n] = 0$	with $y[0] = 3$		
h)	$18y[n+2] - 3y[n+1] = y[n]$	with $y[0] = 1$	and $y[1] = 1/3$	
i)	$y[n+2] - 4y[n+1] + 3y[n] = 2^n$	with $y[0] = 0$	and $y[1] = -1$	
j)	$y[n+2] = 2ay[n+1] - y[n]$	with $y[0] = 1$	and $y[1] = a \in [-1, 1]$	

6.10 Stability

A function $x[n]$ is called stable if its "end-value" is finite, i.e. iff

$$\lim_{n \to +\infty} x[n] \in \mathbb{R}$$

For signals whose Z-transform can be described by a ratio of polynomials in z this translates into the requirement that *all* (*observable*[4]) *poles need to be located within the unit circle*.

This requirement is easy to derive:

$$\begin{aligned}
X(z) &= \frac{N(z)}{D(z)} \\
&\downarrow \quad \text{Polynomial long division} \\
&= T(z) + \frac{N_1(z)}{D(z)} \\
&= T(z) + z\frac{N_1(z)}{zD(z)} \\
&= T(z) + z\frac{\prod_{k=0}^{M}(z-n_k)}{z\prod_{l=0}^{N}(z-p_l)} \\
&\downarrow \quad \text{Partial fraction decomposition} \\
&= T(z) + z\left(\frac{K}{z} + \sum_{l=0}^{N}\frac{A_l}{z-p_l}\right) \\
&= T(z) + K + \sum_{l=0}^{N}\frac{A_l z}{z-p_l}
\end{aligned}$$

with $T(z)$ a polynomial in z. For physical (causal) signals $T(z)$ will be zero and therefore $N_1(z) = N(z)$.

A typical term corresponding to pole p_l will give rise to a term in the time-domain:

$$\frac{A_l z}{z-p_l} \quad \xrightarrow{Z^{-1}} \quad A_l p_l^n = A_l\,|p_l|^n\,\mathrm{e}^{\mathrm{j}n\phi_{p_l}}$$

As poles are either real or appear in complex conjugate pairs, we can again distinguish two cases:

- For real poles, the term reduces to $A_l p_l^n$. The power p_l^n will only be stable if $|p_l| \leq 1$, i.e. if the pole is located inside the unit circle.
- For complex conjugate pole pairs, one can prove that

  $$\overline{p_u} = p_v \Longrightarrow \overline{A_u} = A_v$$

  If we put $A_u = B_u + \mathrm{j}C_u$ with $B_u, C_u \in \mathbb{R}$ and $p_u = |p_u|\,\mathrm{e}^{\mathrm{j}\phi_{p_u}}$, the combination of the two complex conjugate pole terms can be rewritten:

  $$\begin{aligned}
  \frac{z(B_u+\mathrm{j}C_u)}{z-|p_u|\,\mathrm{e}^{\mathrm{j}\phi_{p_u}}} + \frac{z(B_u-\mathrm{j}C_u)}{z-|p_u|\,\mathrm{e}^{-\mathrm{j}\phi_{p_u}}} \quad &\xrightarrow{Z^{-1}} \quad (B_u+\mathrm{j}C_u)\,|p_u|^n\,\mathrm{e}^{\mathrm{j}n\phi_{p_u}} + (B_u-\mathrm{j}C_u)\,|p_u|^n\,\mathrm{e}^{-\mathrm{j}n\phi_{p_u}} \\
  &\xrightarrow{Z^{-1}} \quad B_u\,|p_u|^n\left(\mathrm{e}^{\mathrm{j}n\phi_{p_u}} + \mathrm{e}^{-\mathrm{j}n\phi_{p_u}}\right) \\
  &\qquad\qquad + \mathrm{j}C_u\,|p_u|^n\left(\mathrm{e}^{\mathrm{j}n\phi_{p_u}} - \mathrm{e}^{-\mathrm{j}n\phi_{p_u}}\right) \\
  &\xrightarrow{Z^{-1}} \quad 2B_u\,|p_u|^n\cos\left(n\phi_{p_u}\right) - 2C_u\,|p_u|^n\sin\left(n\phi_{p_u}\right)
  \end{aligned}$$

  These exponentially modulated sines and cosines are only stable if $|p_u| \leq 1$, i.e. if the conjugate pole pair is located inside the unit circle.

[4] A pole is observable if it is not canceled by a zero.

6.11 The extended Fourier family diagram

In this and the previous chapter, we saw how the Laplace and the Z-transform can be derived from the Fourier and the discrete-time Fourier transforms. Because of these relationships, it is logical to add the Laplace and the Z-transform to the Fourier family diagram. Figure 6.3 shows the entire family picture.

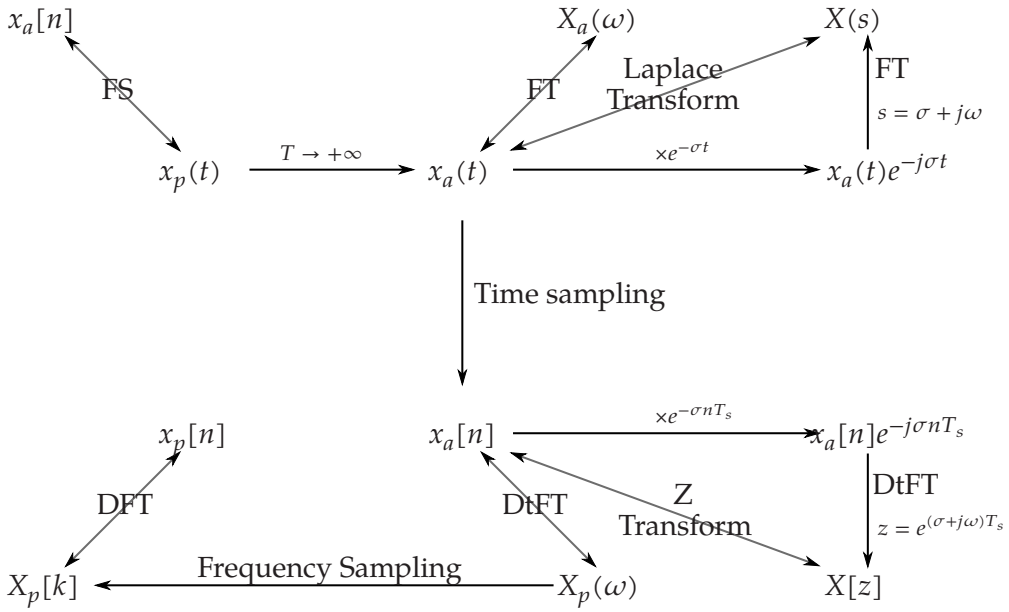


Figure 6.3: Extended Fourier family diagram

In a similar way, we can complete the Fourier family transition diagram to show how to make transitions from one transform to another: see Figure 6.4 on the next page.

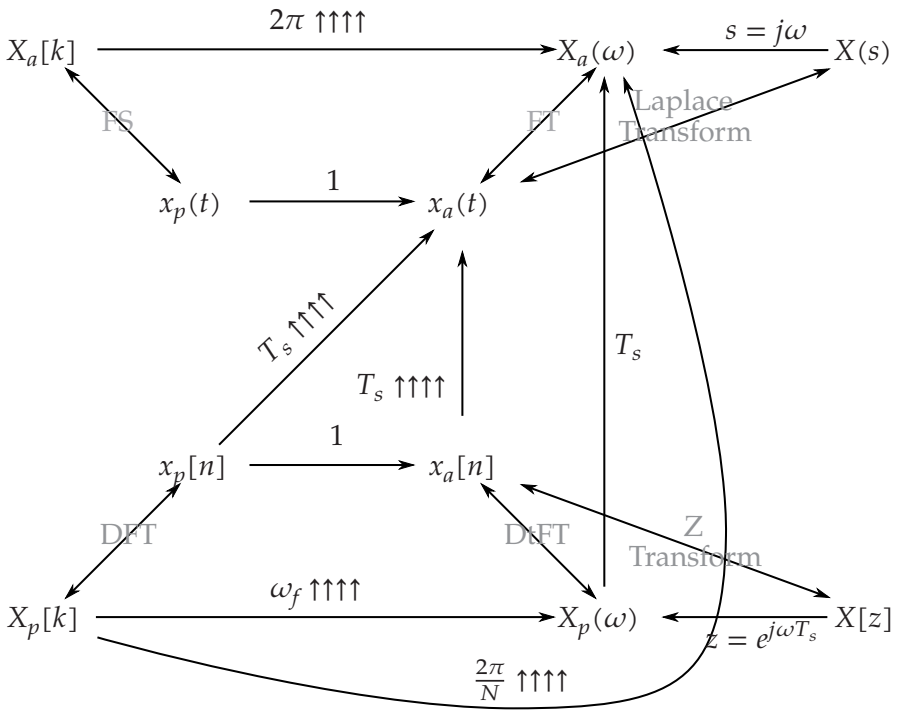


Figure 6.4: Extended Fourier family transition diagram

Appendix A

Mathematical Bits and Pieces

A.1 Taylor's theorem

Preconditions Consider a function $f : \mathbb{R} \to \mathbb{R}$ on a closed interval $[a, b]$ with $a, b \in \mathbb{R}$. Let's assume that the function f is continuous and has continuous derivatives $f^{(i)}$ on the interval $[a, b]$ up to the $(n+1)$-th derivative.

Theorem Under these circumstances, we can find a $c \in [a, b]$ for which the following holds:

$$\begin{aligned} f(b) &= f(a) + f^{(1)}(a) \cdot (b-a) + \frac{f^{(2)}(a)}{2!}(b-a)^2 + \frac{f^{(3)}(a)}{3!}(b-a)^3 + \cdots + \frac{f^{(n)}(a)}{n!}(b-a)^n \\ &\quad + \frac{f^{(n+1)}(c)}{(n+1)!}(b-a)^{(n+1)} \\ &= \sum_{i=0}^{n} \frac{f^{(i)}(a)}{i!}(b-a)^n + \frac{f^{(n+1)}(c)}{(n+1)!}(b-a)^{(n+1)} \end{aligned}$$

This is generally known as *Taylor's theorem* for approximating smooth continuous functions. It is also referred to as approximating functions by *Taylor expansion*. In case $a = 0$, the theorem is also referred to as *McLaurin's theorem*.

Note that the continuity conditions are a precondition to be able to apply this theorem!

Example As an example, consider approximating $\sin x$ for x in the interval $[0, 1]$. The sin-function is infinitely smooth (all its derivatives are continuous) and therefore Taylor's theorem can be applied. Applying the theorem for $n = 2$ yields:

$$\begin{aligned} \sin(x) &= \sin(0) + \cos(0) \cdot x + \frac{-\sin(0)}{2!}x^2 + \frac{\cos(c)}{3!}x^3 \\ &= x + \frac{\cos(c)}{3!}x^3 \end{aligned}$$

with $c \in [0, 1]$.

A.2 The 'Big-O' notation

When calculating limits of a function $f(x)$, it is often not so interesting what the actual limit value is, but rather how quickly it approaches the limit value. This holds especially when the

limiting value is infinity. Consider for example $f(x) = x$ and $f(x) = x^2$. The limit value for $x \to +\infty$ is in both cases $+\infty$. However, the speed at which they approach infinity is different. The function $f(x) = e^x$ even surpasses any x^n for arbitrary n in approaching infinity for $x \to \infty$.

To allow for a rough estimate of the approach speed, the Big-O concept was 'invented'.

A.2.1 Formal definition

Limiting case $x \to a$

We call $r(x) = O(f(x))$ for $x \to a$ if and only if we can find constants ϵ and C for which the following holds:

$$|f(x)| < C\,|r(x)|\,, \forall x \in [a, a+\varepsilon]$$

Limiting case $x \to \infty$

We call $r(x) = O(f(x))$ for $x \to \infty$ if and only if we can find constants x_0 and C for which the following holds:

$$|f(x)| < C\,|r(x)|\,, \forall x > x_0$$

A.2.2 Practical use

In asymptotic behavior of functions

If $r(x) = O(f(x))$ for $x \to a$ or $x \to \infty$ this means that both functions have the same *rate of attack* w.r.t. the asymptote. They may differ up to a multiplicative constant, but if you get close enough to the limiting value, they essentially approach their asymptotes with the same 'speed'. In words, we say 'r of x is big O of f of x when x approaches a'. Another frequently heard phrase is that $r(x)$ is of the same order of magnitude as $f(x)$ when x approaches a (or infinity).

In function approximation

In function approximation the Big-O notation is very used to identify an approximation error that is of a certain order of magnitude. Consider for example the Taylor series expansion of the previous section. The expression

$$f(b) = f(a) + f^{(1)}(a)\cdot(b-a) + \frac{f^{(2)}(a)}{2!}(b-a)^2 + \frac{f^{(3)}(a)}{3!}(b-a)^3 + \cdots + \frac{f^{(n)}(a)}{n!}(b-a)^n$$

is often written as:

$$f(b) = f(a) + f^{(1)}(a)\cdot(b-a) + \frac{f^{(2)}(a)}{2!}(b-a)^2 + \frac{f^{(3)}(a)}{3!}(b-a)^3 + \cdots + O\left((b-a)^n\right)$$

When $b \to a$, the value $(b-a)$ becomes infinitesimally small. in that case $(b-a)^n$ becomes very small. You probably have heard the phrase in case $n = 2$: "These are effects of second order, and therefore, we can neglect them". Of course, the phrase only holds if you think a linear approximation is good enough.

In algorithm development

The behavior of an algorithm w.r.t. the required computation time is usually dependent on the size of the problem. The relationship between size of the problem and the algorithm is called the *computational complexity* of the algorithm.

E.g., when writing a matrix multiplication algorithm to multiply $N \times N$-matrices, a naive implementation would lead you to the conclusion that we need about N^3 multiplications and additions. Of course, there are loop counters to be incremented and there is memory access to be taken care of. This may lead to the conclusion that one needs something like $1.05N^3 + 2N^2 - N + 5$ multiplications and additions.

In Big-O terms, we phrase this as 'matrix multiplication is $O\left(N^3\right)$'.[1]

The practical meaning is obvious. If a double the problem size N for an algorithm that is $O\left(N^3\right)$, the computation time will roughly increase by a factor of 2^3, i.e. 8. If a increase the problem size by a factor of 10, the computation time will roughly increase by a factor of 10^3, i.e. 1000!

Considering the computational complexity of an algorithm when attacking large problems is something you'd better do before writing any program code.

[1] Note that at this moment algorithms exist for matrix multiplication that are $O\left(N^{2.373}\right)$. This explains the adjective 'naive' that is used in the previous paragraph.

Appendix B

Engineering Bits and Pieces

B.1 The Decibel

The *decibel* ([dB]) is a quite common (pseudo) unit. However, it is frequently misunderstood. This is partly due to some fine nuance in the defining equations and partly due to the fact that the concept is heavily (ab)used in multiple engineering domains.

Let's attempt to set things straight.

B.1.1 Definition

B.1.1.1 Power ratio

The starting point for the decibel is the concept of *power ratio*, i.e. the ratio of a power value P_2 with respect to a value P_1.[1] Formally:

$$R_p = \frac{P_2}{P_1}$$

For reasons that will become clear below, it makes sense to take the logarithm of this ratio, leading to:

$$L_p = \log R_p = \log \frac{P_2}{P_1} \qquad [\mathrm{B}]$$

with log being Briggs' logarithm (i.e. base 10). The quantity L_p is assigned the unit Bel ([B]) in honour of Graham Bell. It is a dimensionless and only indicates that the value expresses a logarithmic power ratio.

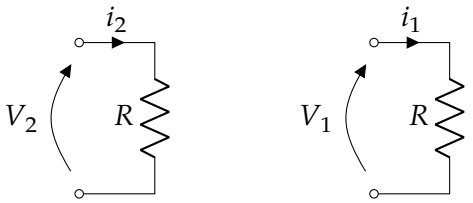
For some reason (probably to be able to use integer numbers in most common cases), it was then decided to use the deciBel ([dB]) instead of the unit Bel. In physical equations we normally don't introduce constants for unit conversions[2], but in this case, it has become quite common to write:

$$L_p = 10 \log \frac{P_2}{P_1} \qquad [\mathrm{dB}]$$

Even more, from now on, we will stop writing explicitly the unit dB after the equation, but silently assume that everyone knows this.

[1] Instead of power quantities, also intensity quantities qualify, e.g. $\mathrm{W/m^2}$ or lm/sr.
[2] We assume SI units are always used.

In theory, that's all there is to it. However, in practice, people got used to calculating L_p starting from field quantities (e.g., V, V/m,Pa)instead of power quantities. Instead of making the intermediate step, converting the field quanties to power (or intensity) quantities, a formula was developed that avoids that intermediate step.
Consider as an example a voltage V_2 applied to a resistor R, in comparison to a voltage V_1 applied to the same resistor R.



The power consumed by the resistors is V_2^2/R and V_1^2/R respectively. Therefore, the power ratio L_p can be written as:

$$L_p = 10\log\frac{P_2}{P_1} = 10\log\frac{V_2^2/R}{V_1^2/R} = 10\log\left(\frac{V_2}{V_1}\right)^2$$
$$= 20\log\frac{V_2}{V_1}$$

The last step can be made knowing that $\log a^b = b\log a$.

This convenience equation, though simple, has led to a lot of misunderstanding, ranging from doubt when to use the factor 10 versus the factor 20, to speaking about dB-power and dB-voltage.

The following is the only correct statement: the value of L_p represents *always* a power ratio, but it can be calculated starting from a ratio of powers (using the factor of 10), or starting from a ratio of field quantities (using the factor of 20).

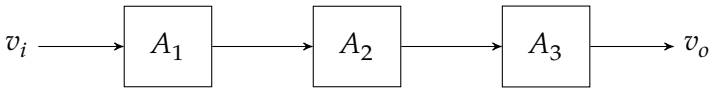
B.1.1.2 Absolute power quantity

The unit decibel [dB] is also often used to express absolute quantities by comparing them to a fictive reference value.

B.1.2 Use

Now, what's the point of using this logarithmic value instead of just a simple ratio?

Simplification of calculations A convincing use case is when one considers a chain of signal amplification blocks amplifying (or attenuating) an input voltage v_i to an output voltage v_o:



This could be a communications channel, with a transmit amplifier, a channel loss and a reception amplifier.
Calculating v_o as a function of v_i is simple:

$$v_o = A_1 \cdot A_2 \cdot A_3 \cdot v_i$$

Referring the voltages to a common reference voltage v_r (i.e. dividing both sides of the equation above by v_r) leads to:

$$\frac{v_o}{v_r} = A_1 \cdot A_2 \cdot A_3 \cdot \frac{v_i}{v_r}$$

Then, let's take the 20-fold logarithm of both sides and elaborate:

$$\begin{aligned} 20\log\left(\frac{v_o}{v_r}\right) &= 20\log\left(A_1 \cdot A_2 \cdot A_3 \cdot \frac{v_i}{v_r}\right) \\ &\downarrow \quad \log(a \cdot b) = \log a + \log b \\ L_{P_o} &= 20\log A_1 + 20\log A_2 + 20\log A_3 + L_{P_i} \end{aligned}$$

Or in a sloppy, but very engineeringish notation:

$$v_{i,\mathrm{dB}} = A_{1,\mathrm{dB}} + A_{2,\mathrm{dB}} + A_{3,\mathrm{dB}} + v_{o,\mathrm{dB}}$$

This means that the *complex* multiplication has been replaced by a more simple *addition*.

Logarithmic quantities A second compelling reason to use logarithmic power ratios is that one of our human senses, hearing, is logarithmic in nature. It requires doubling the sound power to increase the perceived loudness by a fixed amount.

B.1.3 Examples

The decibel is used in many physics/engineering domains, each domain having its own reference power. The table below, lists some typical cases.

Unit	Reference	Use
dBV	1 V	voltages regardless of impedance.
dBmV	1 mV	voltages regardless of impedance.
dBµV	1 µV	voltages regardless of impedance.
dBm	1 mW	signal power w.r.t.
		- 600 Ω in audio electronics
		- 50 Ω in RF electronics
dBSPL[3]	20 µPa	sound pressure level in air
	1 µPa	sound pressure level in liquids
dBSIL	1×10^{-12} W/m^2	sound intensity level
dBSWL	1×10^{-12} W	sound power level
dBc	carrier power	noise or sideband power relative to the carrier

The goal of the table above was not to give an exhaustive overview, but to show you how heavily it is used in different domains. So, you'd better check the exact reference of the specific unit at hand, before you trust any readings in it.

B.1.4 Rules of thumb for calculating with dBs

The following estimation table shows you howto quickly determine the ratio that corresponds to a certain dB-level.

L_p	Power ratio	Field ratio
3 dB	2	$\sqrt{2}$
6 dB	4	2
10 dB	10	π
20 dB	100	10

Of course, the value of π is not correct, but it comes darned close.

As an example, let's try to estimate to what power ratio and what field ratio a decibel-level $L_p = 36\,\text{dB}$ corresponds.

$$\begin{aligned} L_p = 36\,\text{dB} &= (20 + 10 + 6)\text{dB} \\ &\Downarrow \\ \frac{P_2}{P_1} &= 100 \cdot 10 \cdot 4 = 4000 \\ \frac{V_2}{V_1} &= 10 \cdot \pi \cdot 2 = 62.8 \end{aligned}$$

If you need more accurate figures (e.g., with an accuracy of 1 dB or less), then you'd better use a calculator.

B.1.5 Conclusion

Don't be afraid when meeting decibels as an engineer, but make sure to exactly know what the reference value is.

B.2 Representing impulse trains: the Sha-function

Impulse trains are such a fundamental concept in digital signal processing that Ronald Bracewell, a former professor of electrical engineering of Stanford University, proposed a new notation for them.

The notation is very compact and makes use of a letter of the Cyrillic script: the Sha, written as ш and mostly used as a capital Ш.

The definition Bracewell proposed, combines the letter Sha (the peaks representing the individual pulses) with a subscript, indicating the spacing in between the peaks. In the continuous domain, with t the domain variable, this leads to:

$$\text{Ш}_T(t) = \sum_{i=-\infty}^{+\infty} \delta(t - iT)$$

It may also be used in the discrete domain (e.g. as a *resampling function*). With n the domain variable, this leads to:

$$\text{Ш}_N[n] = \sum_{i=-\infty}^{+\infty} \delta[n - iN]$$

The advantage of the Sha-notation is that it keeps focus on the important details (the spacing of the Dirac impulses), rather than on the cumbersome details of the summation and its indexing.

B.3 Orthogonal signal decompositions

Decomposing signals w.r.t. an orthogonal basis, is based on the observation that we can consider discrete-time signals to be vectors in a vector space.

We assume that the reader is familiar with the basics of vector spaces. The overview is only intended to refresh this knowledge and demonstrate how it can be applied to discrete-time signals. We start by investigating the elements of the set of signals.

B.3.1 Elements

The set under consideration is the set of all discrete-time signals S. These signals are sequences of numbers w.r.t. a time reference (e.g., $n = 0$).

Our signal f could be equidistantly sampled versions of a continuous-time function g:

$$f[n] = g(n\Delta), \qquad \forall n \in \mathbb{Z}$$

with Δ the *sampling period*.

However, f could as well be an abstract ordered set of numbers:

$$f[n] = [\dots, f_r, f_s, f_t, f_u, \underbrace{f_v}_{n=0}, f_w, f_x, f_y, \dots]$$

To make writing abstract signals easier, we will use integers as subscripts, with the subscript 0 referring to the time point $n = 0$. The abstract row above then becomes:

$$f[n] = \left[\dots, f_{-4}, \; f_{-3}, \; f_{-2}, \; f_{-1}, \; f_0, \; f_1, \; f_2, \; f_3, \; \dots\right]$$

We will concisely refer to the above equation as: $f[n] = [\dots, f_i, \dots]$

If a signal starts at $n = 0$ (i.e. it is zero for negative times, a so-called *causal signal*), we will omit the underbrace indicating the time point $n = 0$ and simply write

$$f[n] = \left[f_a, \; f_b, \; f_c, \; f_d, \; \dots\right]$$

silently assuming that f_a is the signal value for $n = 0$.

The signals we consider can be real or complex-valued ($f_i \in \mathbb{R}$ or $f_i \in \mathbb{C}$). Since $\mathbb{R} \subset \mathbb{C}$ we will only look at the more general case of complex-valued signals. In most occasions, the signals will have a limited time span (a.k.a. *limited support*), i.e., they are only nonzero in a limited range of time-indices. If these signals are only active for N time points, we will denote the set by S_N.

Finally, we also refer to the signal $f[n]$ in short as $\vec{f}$. When using $f[n]$ or $\vec{f}$ in the context of matrices, we will treat them as column vectors.

Remarks

- The arrow vector notation $\vec{f}$ is not so common in the wavelet and signal processing literature.

- However, an ambiguity arises from the fact that f_3 might denote the value of f for $n = 3$, but might as well denote a third version of f. In the former case it denotes a scalar, in the second case, it denotes a vector. In many (but not all cases) the correct meaning can be deduced from the context. However, in an introductory text, like this one, this puts an unacceptable burden on the novice reader. Therefore, we've chosen to adopt the vector notation, but in a limited fashion. Whenever a signal is not written as a function, we will use the arrow notation[4]. Whenever a signal is written as a function, we don't use the arrow symbol.

B.3.2 Operations

Addition Addition of two signals $\vec{f}$ and $\vec{g}$ is denoted by $\vec{f} + \vec{g}$ and obtained by adding the values for corresponding time points. If $f[n] = [\ldots, f_i, \ldots]$ and $g[n] = [\ldots, g_i, \ldots]$ then $\vec{h} = \vec{f} + \vec{g}$ is defined by:

$$h[n] = (f + g)[n] = [\ldots, f_i + g_i, \ldots]$$

Scaling Scaling a signal $\vec{f}$ with a scalar c out of a field F is denoted by $c\vec{f}$ and obtained by scaling all the individual values. If $f[n] = [\ldots, f_i, \ldots]$, then

$$cf[n] = [\ldots, cf_i, \ldots]$$

Most often, F is the set of real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$. In the former case, this leads to a *real vector space*; in the latter case to a *complex vector space*.

Scaling The scalar product of two signals $\vec{f}$ and $\vec{g}$ is denoted by $\langle \vec{f}, \vec{g} \rangle$. It is the basis for many new geometric notions like orthogonality and also length and distance.

If $f[n] = [\ldots, f_i, \ldots]$, and $g[n] = [\ldots, g_i, \ldots]$, then

$$\langle \vec{f}, \vec{g} \rangle = \sum_{i=-\infty}^{+\infty} f_i \overline{g_i} \tag{B.1}$$

where $\overline{g_i}$ denotes the complex conjugate of g_i.
For real signals $u[n] = [\ldots, u_i, \ldots]$ and $v[n] = [\ldots, v_i, \ldots]$, this reduces to:

$$\langle \vec{u}, \vec{v} \rangle = \sum_{i=-\infty}^{+\infty} u_i v_i$$

Note that for signals that are only active on time points ranging from r to t (i.e., they are of limited support), (B.1) reduces to:

$$\langle \vec{f}, \vec{g} \rangle = \sum_{i=r}^{t} f_i \overline{g_i}$$

B.3.3 Geometric notions

Two signals are orthogonal if their scalar product is zero. In symbols:

$$\vec{f} \perp \vec{g} \quad \Leftrightarrow \quad \langle \vec{f}, \vec{g} \rangle = 0$$

[4] A similar convention is to use a boldface font and is more common in signal processing literature. However, this is not compatible with hand writing. We therefore have chosen to use the arrow notation.

We can define the norm $\|\vec{f}\|$ of a signal $\vec{f}$ as:

$$\|\vec{f}\| = \sqrt{\langle \vec{f}, \vec{f} \rangle}$$

The norm of a signal $\vec{f}$ is a measure of its size. Very often the square of the norm is used, as it denotes the energy of the signal.

$$E = \|\vec{f}\|^2 = \langle \vec{f}, \vec{f} \rangle$$

We can define the distance $d(\vec{f}, \vec{g})$ between two signals $\vec{f}$ and $\vec{g}$ as:

$$d(\vec{f}, \vec{g}) = \|\vec{f} - \vec{g}\|$$

The distance is a measure of the similarity of the two signals. The smaller the distance, the more the signals are alike.

B.3.4 Vector space

The set of all complex-valued discrete-time signals S forms a vector space when it is equipped with the addition, scaling and scalar product as defined above.

Specifically, this means that the following properties hold. Let $\vec{x}$, $\vec{y}$ en $\vec{z}$ be arbitrary vectors in S, and a and b scalars.

1. the addition is commutative: $\vec{x} + \vec{y} = \vec{y} + \vec{x}$
2. the addition is associative: $(\vec{x} + \vec{y}) + \vec{z} = \vec{x} + (\vec{y} + \vec{z})$
3. the addition has an identity element $\vec{0}$: $\vec{x} + \vec{0} = \vec{x}$
4. every vector has an inverse element w.r.t. the addition $\forall \vec{x} \in S, \exists -\vec{x} \in S : \vec{x} + (-\vec{x}) = \vec{0}$
5. the scaling has an identity element, the scalar '1': $1\vec{x} = \vec{x}$
6. the scaling is distributive with respect to the vector addition: $a(\vec{x} + \vec{y}) = a\vec{x} + a\vec{y}$
7. the scaling is distributive with respect to the scalar addition: $(a + b)\vec{x} = a\vec{x} + b\vec{x}$

B.3.5 Set notions

Orthogonal set A (finite or infinite but countable) set of signals $V = \{\vec{e}_1, \vec{e}_2, \vec{e}_3, \ldots\}$ is a so-called *orthogonal set* of signals if and only if for every i and j holds:

$$\langle \vec{e}_i, \vec{e}_j \rangle = \begin{cases} 0 & \text{if} \quad i \neq j \\ \|e_i\|^2 \neq 0 & \text{if} \quad i = j \end{cases}$$

Independent set A (finite or infinite but countable) set of signals $V = \{\vec{e}_1, \vec{e}_2, \vec{e}_3, ...\}$ is an *independent set* of signals, if and only if the equation with scalars c_i

$$c_1\vec{e}_1 + c_2\vec{e}_2 + c_3\vec{e}_3 + \cdots = 0$$

only holds for $(c_1, c_2, c_3, ...) = (0, 0, 0, ...)$

Span The span S of a (finite or infinite but countable) set $V = \{\vec{e}_1, \vec{e}_2, \vec{e}_3, ...\}$ is defined by

$$S = \{\vec{g} \mid \exists (c_1, c_2, c_3, ...) \text{ with } c_i \in \mathbb{C} : \vec{g} = c_1\vec{e}_1 + c_2\vec{e}_2 + c_3\vec{e}_3 + \cdots\}$$

A commonly used notation is $S = \text{span}\,\{V\}$

B.3.6 Base and dimension of a vector space

Base A subset of vectors $B = \{\vec{e}_1, \vec{e}_2, \vec{e}_3, ...\}$ with $\vec{e}_i \in S$ is called a base of S if and only if

- B is an independent set
- B is complete, i.e. $S = \text{span}\,\{B\}$

If in addition B is an orthogonal set, then B is an *orthogonal base* of S.

If in addition $\|\vec{e}_i\| = 1, \forall i$, then B is an *orthonormal base* of S.[5]

Dimension The dimension of a vector space equals the size of the base, i.e. the number of base vectors.

For signals of unlimited support, the dimension of the vector space is infinite (but countable). When we restrict ourselves to signals of a specific limited support, the dimension of the vector space equals the support of the signals.

B.3.7 Decomposition of vectors in terms of the base vectors

One can prove that for a signal $\vec{f}$ and an orthogonal base $B = \{\vec{e}_1, \vec{e}_2, \vec{e}_3, ...\}$ and the decomposition:

$$\vec{f} = c_1\vec{e}_1 + c_2\vec{e}_2 + c_3\vec{e}_3 + ...$$

the coefficients c_i can be determined as:

$$c_i = \frac{1}{\|\vec{e}_i\|^2}\left\langle \vec{f}, \vec{e}_i \right\rangle = \frac{\left\langle \vec{f}, \vec{e}_i \right\rangle}{\left\langle \vec{e}_i, \vec{e}_i \right\rangle}$$

This is the so-called *projection equation*. It calculates the projection of $\vec{f}$ onto a specific base vector.

[5] In most literature, an orthonormal set of vectors is referred to as an *orthogonal set*. This is most confusing and for this reason, I chose not to use this very common misnomer.

If the base is orthonormal, we can simplify the above to:

$$c_i = \frac{1}{\underbrace{\left\|\vec{e}_i\right\|^2}_{=1}} \left\langle \vec{f}, \vec{e}_i \right\rangle = \left\langle \vec{f}, \vec{e}_i \right\rangle$$

B.3.8 Parseval's identity

Given a signal $\vec{f}$ and an orthogonal base $B = \{\vec{e}_1, \vec{e}_2, \vec{e}_3, ...\}$, such that $\vec{f} = \sum_i c_i \vec{e}_i$, then

$$\left\|\vec{f}\right\|^2 = |c_1|^2 \left\|\vec{e}_1\right\|^2 + |c_2|^2 \left\|\vec{e}_2\right\|^2 + |c_3|^2 \left\|\vec{e}_3\right\|^2 + \cdots \tag{B.2}$$

In fact, Parseval's identity is a generalization of the Pythagorean theorem.

If the base B is orthonormal, (B.2) reduces to:

$$\left\|\vec{f}\right\|^2 = |c_1|^2 + |c_2|^2 + |c_3|^2 + \cdots \tag{B.3}$$

Very often this simplified equation is called the *energy preservation theorem* in relation to signal transforms, because of the following. When considering the sequence $[c_1, c_2, c_3, ...]$ to be a transformed signal $\vec{c}$ of the original signal $\vec{f}$, we can rewrite (B.3) as

$$\left\|\vec{f}\right\|^2 = \left\|\vec{c}\right\|^2$$

If we restrict ourselves to real signals, then (B.2) reduces to:

$$\left\|\vec{f}\right\|^2 = c_1^2\left\|\vec{e}_1\right\|^2 + c_2^2\left\|\vec{e}_2\right\|^2 + c_3^2\left\|\vec{e}_3\right\|^2 + \cdots$$

When, in addition the base is normalized, we can go one step further and write:

$$\left\|\vec{f}\right\|^2 = c_1^2 + c_2^2 + c_3^2 + \cdots$$

and again when $\vec{c} = [c_1, c_2, c_3, ...]$:

$$\left\|\vec{f}\right\|^2 = \left\|\vec{c}\right\|^2$$

Bibliography

[Boz94] S.M. Bozic. *Digital and Kalman Filtering*. Edward Arnold, 2nd edition, 1994.

[BtMvdBJvdV93] R.J. Beerends, H.G. ter Morsche, van den Berg J.C., and E.M. van de Vrie. *Fourier & Laplace transformaties (in Dutch)*. Educaboek, 1993.

[DLW06] Gustaaf Deen, Paul Levrie, and Vanneste Wilfried. *Aanvullingen van Wiskunde (in Dutch)*. Karel de Grote-Hogeschool, Katholieke Hogeschool Antwerpen, 2006.

[Eat07] John W. Eaton. *GNU Octave — A high-level interactive language for numerical computations*. Free Software Foundation, Inc., 51, Franklin Street, Boston, MA 02110-1301, USA, 3rd edition, 2007. (online available on http://www.gnu.org/software/octave).

[FJ08] Matteo Frigo and Steven G. Johnson. FFTW — fastest fourier transform in the west. (online available on http://www.fftw.org), 2008.

[GR80] Robert A. Gabel and Richard A. Roberts. *Signals and Linear Systems*. John Wiley & Sons, 2nd edition, 1980.

[Lyo04] Richard G. Lyons. *Understanding Digital Signal Processing, 2nd Edition*. Prentice Hall, 2004.

[MAT15] MATLAB. *R2015a*. Natick, Massachusetts, 2015.

[MO94] H. Mannaert and A. Oosterlinck. *Stochastische Signaalanalyse en Filterontwerp (in Dutch)*. Katholieke Universiteit Leuven, 1994.

[OSB99] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time Signal Processing, 2nd Edition*. Prentice Hall, 1999.

[Sij04] Jan Sijbers. *Inleiding tot Digitale Signaalverwerking (in Dutch)*. Universiteit Antwerpen, 2004.

[Smi03] Stephen W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing, 2nd Edition*. California Technical Publishing, 2003.

[Smi08a] Julius O. Smith. *Mathematics of the Discrete Fourier Transform (DFT)*. http://ccrma.stanford.edu/~jos/mdft, accessed 2008. online book.

[Smi08b] Julius O. Smith. *Spectral Audio Signal Processing, March 2007 Draft*. http://ccrma.stanford.edu/~jos/sasp, accessed 2008. online book.

[vdEV87] A.W.M. van den Enden and N.A.M. Verhoeckx. *Digitale signaalbewerking (in Dutch)*. Delta Press, 1987.

[vdW92] P.E.M van den Wyngaert. *Analoge en digitale ketens (in Dutch)*. Katholieke Industriële Hogeschool Antwerpen, 1992.

Index