CHAPTER 2

FUNDAMENTALS

This chapter re-introduces and summarizes some of the important characteristics of digital audio signals and tools used in signal processing and ACA. Furthermore, it will introduce some definitions used in the following chapters. In order to keep this chapter short, some of the definitions may lack derivation since it is not possible to give an extensive introduction to digital audio signal processing in this context. The interested reader may, for instance, refer to the books by Oppenheim and Schafer [6], Ohm and Lüke [7], and Smith [8] for more complete introductions.

2.1 Audio Signals

An *audio signal* as humans perceive it can be described as a function of time-variant sound pressure level. A microphone can be used to convert the sound pressure level into voltage. The signal is defined for all times t and is therefore a (time-) continuous signal $x(t)$.

2.1.1 Periodic Signals

A *periodic signal* repeats itself in constant time intervals. Its periodicity can be formulated by

$$x(t) = x(t + T_0) \tag{2.1}$$

with T_0 being the periodicity interval, or, in other words, the *period length* of the first of the harmonics. The *fundamental frequency* of this periodic signal is then

$$f_0 = \frac{1}{T_0}. \tag{2.2}$$

The frequency of other tonal components, the remaining harmonics, is always an integer multiple of the frequency of the first harmonic.

Every periodic signal $x(t)$ can be represented by a superposition of weighted sinusoidal signals, the *Fourier series*

$$x(t) = \sum_{k=-\infty}^{\infty} a(k)e^{\mathrm{j}\omega_0 kt}. \tag{2.3}$$

The periodicity, or the fundamental frequency, of the signal is represented as angular frequency $\omega_0 = 2\pi f_0$. The real part of $e^{\mathrm{j}\omega_0 kt}$ represents the cosine components (even) and its imaginary part the sine components (odd) of the signal: $e^{\mathrm{j}\omega t} = \cos(\omega t) + \mathrm{j}\sin(\omega t)$. The coefficient $a(k)$ is the weight of the kth harmonic and can be calculated by

$$a(k) = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x(t)e^{-\mathrm{j}\omega_0 kt}\, dt. \tag{2.4}$$

If the number of frequencies used to represent the signal is limited

$$\hat{x}(t) = \sum_{k=-O}^{O} a(k)e^{\mathrm{j}\omega_0 kt}, \tag{2.5}$$

then some periodic signals may be constructed only imperfectly. The larger the order O, the higher is the highest frequency included and the better is the representation of the signal.

Periodic signals featuring "sudden changes" such as discontinuities of the waveform require higher order Fourier coefficients to model the waveform sufficiently well. The modeling of discontinuities and so-called transients requires high frequencies. More systematically we can say that the higher the order of derivations of the signal that are monotone is the lower the required order will be to represent the signal sufficiently well. The most extreme example is a sinusoidal signal which has an infinite number of monotone derivatives and for which only one coefficient $a(1)$ is required to represent the signal perfectly.

Figure 2.1 shows one period of two periodic signals, a *sawtooth* (top) and a *square wave* (bottom), and their representation with the model orders 3, 25, and unlimited. The coefficients $a(k)$ for these two signals are

$$a_{\text{saw}}(k) = \frac{2}{\pi \cdot k}, \tag{2.6}$$

$$a_{\text{rect}}(k) = \frac{4}{\pi \cdot (2k-1)}. \tag{2.7}$$

Due to the discontinuities of these two signals there are model errors in the form of overshoots around the point of discontinuity. It can be observed that the frequency of the overshoots increases and the duration of the overshoots decreases with increasing model order. However, the amplitude of the overshoots stays constant unless the order is infinite. This is called *Gibbs' phenomenon* [9, 10].

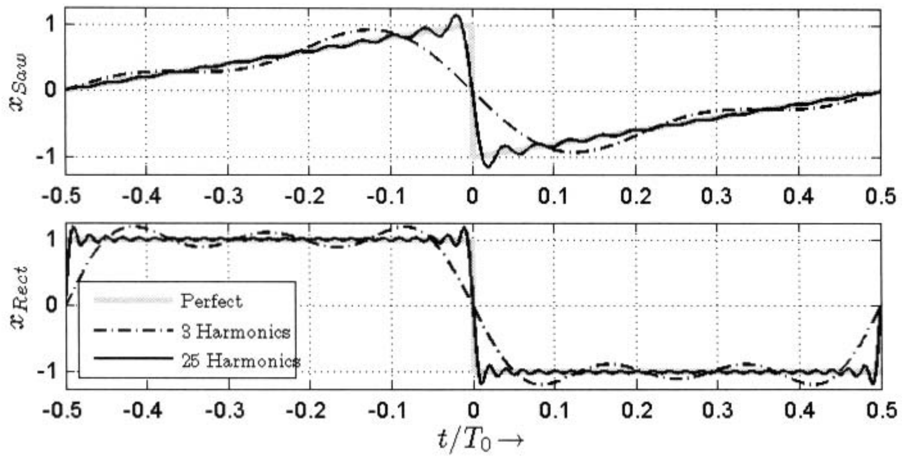

Figure 2.1 Approximation of periodic signals: sawtooth (top) and square reconstructed with 3 and 25 sinusoidal harmonics

2.1.2 Random Signals

In contrast to periodic signals, future values of *random signals* cannot be predicted no matter how long the signal has been observed. That means that there exists no fundamental frequency for this type of signal. Every observation of such a signal is therefore only an example of an unlimited number of possible incarnations of the signal. A complete signal description is theoretically only possible with an unlimited number of observations.

An important subcategory of random signals is built of *stationary* signals for which basic properties (such as arithmetic mean and higher central moments, see below) do not change over time. *White noise* is a typical example of a stationary random signal.

2.1.3 Sampling and Quantization

In the digital domain, we cannot deal with continuous signals. Therefore the analogue, continuous signal $x(t)$ has to be discretized in both amplitude and time, where *quantization* refers to the discretization of amplitudes and *sampling* refers to the discretization in time. The resulting signal is a series of quantized amplitude values $x(i)$.

2.1.3.1 Sampling

The discretization in time is done by *sampling* the signal at specific times. The distance T_S in seconds between sampling points is equidistant in the context of audio signals and can be derived directly from the *sample rate* f_S by

$$T_S = \frac{1}{f_S}. \qquad (2.8)$$

Figure 2.2 visualizes the process of sampling by plotting a continuous signal (top) and a sampled representation of it (bottom) sampled at a sample rate of $f_S = 700\,\text{Hz}$.

An important property of sampled signals is that the sampled representation is ambiguous as exemplified in Fig. 2.3. This example shows that different input signals may lead to the same series of samples.

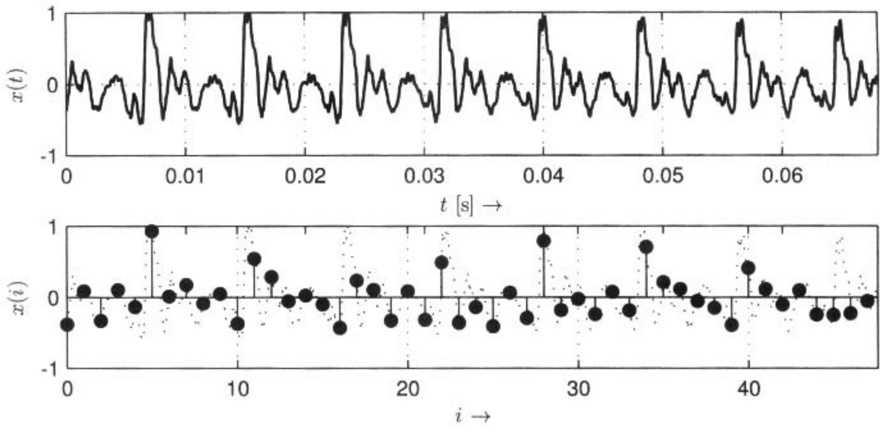

Figure 2.2 Continuous audio signal (top) and corresponding sample values at a sample rate of $f_S = 700\,\text{Hz}$)

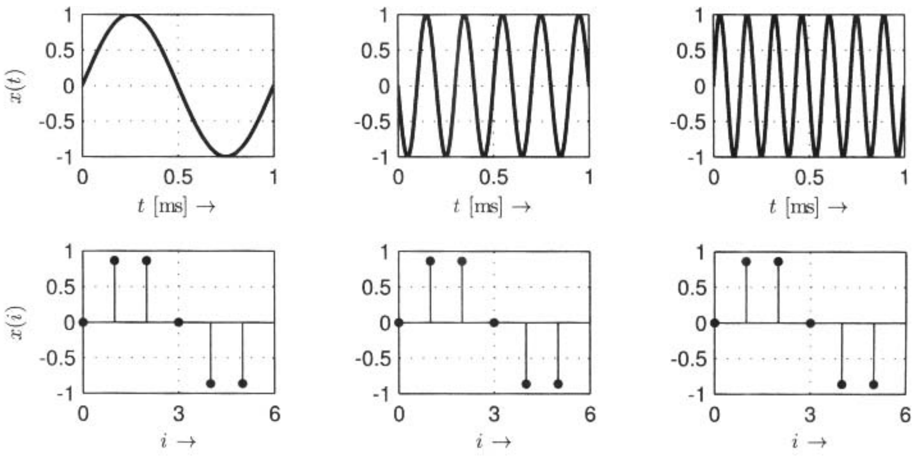

Figure 2.3 Continuous (top) and sampled (below) sinusoidal signals with the frequencies 1 kHz (left), 5 kHz (mid), 7 kHz (right). The sample rate is $f_S = 6\,\text{kHz}$

The signal can only be reconstructed unambiguously when certain requirements for audio signal and sample rates are met as defined by the *sampling theorem*:

Sampling Theorem

A sampled signal can only be reconstructed without loss of information if the sample rate f_S is higher than twice the highest frequency $f_{\max}$ in the sampled audio signal.

$$f_S > 2 \cdot f_{\max} \tag{2.9}$$

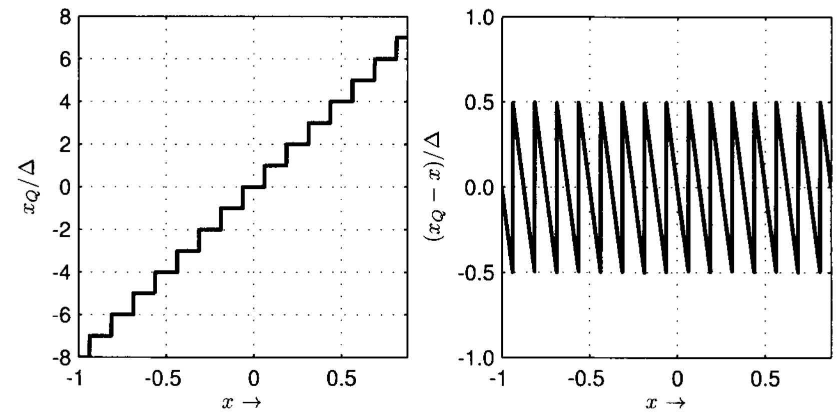

Figure 2.4 Characteristic line of a quantizer with word length $w = 4$ bits showing the quantized output amplitude for a normalized input amplitude (left) and the quantization error with respect to the input amplitude (right)

Historically the sampling theorem is attributed to Kotelnikov [11] and Shannon [12] and is based on work by Whittaker [13] and Nyquist [14].

If the signal contains frequency components higher than half the sample rate, these components will be mirrored back and *aliasing* occurs. See also Appendix B.3 for further explanation of this artifact.

2.1.3.2 Quantization

In order to discretize the signal amplitudes the signal is quantized. *Quantization* means that each amplitude of the signal is rounded to a pre-defined set of allowed amplitude values. Usually, the input amplitude range between the maximum and the minimum amplitude is assumed to be symmetric around 0 and is divided into $\mathcal{M}$ steps. If $\mathcal{M}$ is a power of 2, the amplitude of each quantized sample can easily be represented with a binary code of *word length*

$$w = \log_2(\mathcal{M}). \tag{2.10}$$

Since $\mathcal{M}$ is an even number, it is impossible to both represent zero amplitude as a quantization step and maintain a symmetric number of quantization steps above and below zero. The usual approach is to have one step less for positive values. Typical word lengths for audio signals are 16, 24, and 32 bits. Figure 2.4 shows the characteristic line of a quantizer and the corresponding *quantization error* in dependence of the input amplitude.

The distance Δ_Q between two neighboring quantization steps is usually constant for all steps. That means that the quantization error e_Q is always in the range $\left[-{}^{\Delta_Q}/_2; {}^{\Delta_Q}/_2\right]$ if the maximum input signal amplitude is within the range spanned by the maximum allowed quantized amplitudes (no *clipping*). The quantization error is the difference between original signal x and quantized signal x_Q:

$$e_Q(i) = x(i) - x_Q(i). \tag{2.11}$$

If the input signal amplitudes are much larger than the step size Δ_Q but in the range of allowed amplitudes, the distribution of quantization error amplitudes can be assumed to be rectangular. The power of the quantization error P_Q can then be estimated with its *Probability Density Function (PDF)* $p_q(e_Q)$ (compare Sect. 2.1.4.1):

$$P_Q = \int_{-\infty}^{\infty} e_Q^2 \cdot p_q(e_Q)\, de_Q = \frac{1}{\Delta_Q} \int_{-\Delta_Q/2}^{\Delta_Q/2} e_Q^2\, de_Q = \frac{\Delta_Q^2}{12}. \tag{2.12}$$

It is obvious that the power of the quantization error will be lower the smaller the step size Δ_Q and the higher the word length w is, respectively. The *Signal-to-Noise Ratio (SNR)* in decibels is a good criterion of the quality of a quantization system. The SNR can be calculated from the ratio of the signal power P_S and the quantization error power P_Q by

$$\text{SNR} = 10 \log_{10}\left(\frac{P_S}{P_Q}\right) \text{ [dB]}. \tag{2.13}$$

A high SNR thus indicates good quantization quality. Using Eqs.˜(2.12) and (2.13) it follows that increasing the word length by one bit gains approximately 6 dB:

Signal-to-Noise Ratio (SNR)

$$\text{SNR} = 6.02 \cdot w + c_S \text{ [dB]} \tag{2.14}$$

The constant c_S depends on the signal's amplification and PDF:

- square wave (full scale): $c_S = 10.80\,\text{dB}$
- sinusoidal wave (full scale): $c_S = 1.76\,\text{dB}$
- rectangular PDF (full scale): $c_S = 0\,\text{dB}$
- Gaussian PDF (full scale = $4\sigma_g$)[1]: $c_S = -7.27\,\text{dB}$

Assuming that a real-world audio signal is closer to the PDF of a noise with Gaussian PDF than to that of a sine wave, the typical SNR is approximately 8 dB lower than commonly stated. Furthermore, the above reference values are given for signals that are leveled more or less optimally; if the input signal is scaled, the SNR will decrease accordingly (by 6.02 dB per scaling factor $1/2$).

The term *full scale* refers to the highest quantization step. A full-scale sine wave will thus have an amplitude equaling the highest quantization step (compare also page 72).

These considerations are often only of limited use to the signal processing algorithm designer, as the quantized signal is usually converted to a signal in floating point format, effectively resulting in a non-linear characteristic line of the quantizer. The floating point stores the number's mantissa and exponent separately so that the quantization step size Δ_Q basically increases with the input amplitude. In practice the signals in floating point format are just used as if they had continuous amplitude values, although the quantization error still persists. The choice of a maximum amplitude is arbitrary in floating point format; the most common way is to map full scale to the range $[-1; 1]$ to make the processing independent of the quantizer's word length.

[1]Using the approximation that no values are clipped.

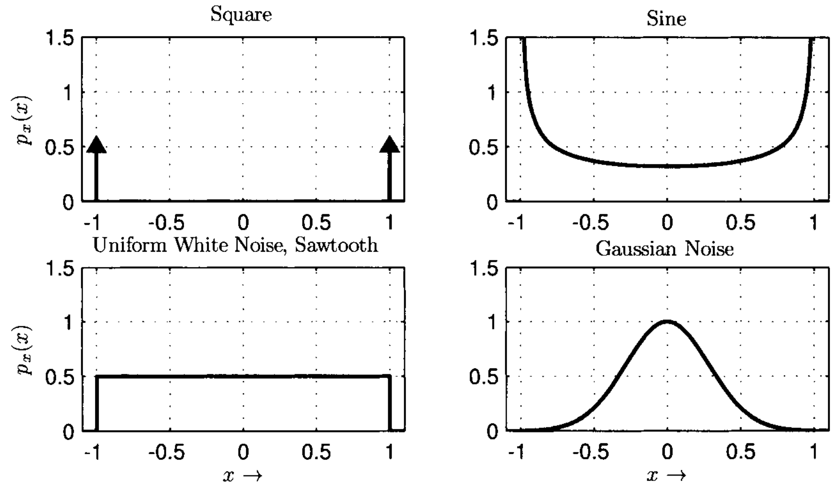

Figure 2.5 Probability density function of a square wave (top left), a sinusoidal (top right), uniform white noise or a sawtooth wave (bottom left), and Gaussian noise (bottom right)

2.1.4 Statistical Signal Description

In contrast to sinusoidal signals, noise cannot be described in the time domain with analytical functions. Statistics can help to identify some properties that describe the signal when neglecting its evolution in time. One of the most common representations is the PDF, which is a useful tool to describe stationary processes or signals that have the same properties for any point in time t.

2.1.4.1 Probability Density Function

The *PDF* $p_x(x)$ of a signal is the probability distribution of a signal. The abscissa of a PDF plot represents all possible amplitude values of the signal x and their probability is plotted on the ordinate. For example, a rectangular PDF means that all possible signal amplitudes occur with the same probability. The properties of the PDF are

$$p_x(x) \geq 0, \text{ and} \tag{2.15}$$

$$\int_{-\infty}^{\infty} p_x(x)\,dx = 1. \tag{2.16}$$

A set of prototypical PDFs is shown in Fig. 2.5. For a full-scale square wave, the PDF has only two peaks at maximum and minimum amplitudes while other amplitude values do not exist. A sine wave has a PDF in which values similar to the arithmetic mean (here: 0) are less frequent than values far from the mean. A uniform PDF can be found with noise generated with the `rand` function of different programming languages; a sawtooth has such a uniform PDF as well. A *Gaussian distribution*

$$p_g = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu_x)^2}{2\sigma^2}\right) \tag{2.17}$$

is assumed to be a typical distribution of "real-world" noise; a typical music signal has in many cases a distribution shaped similar to a *Laplace distribution* with a prominent peak

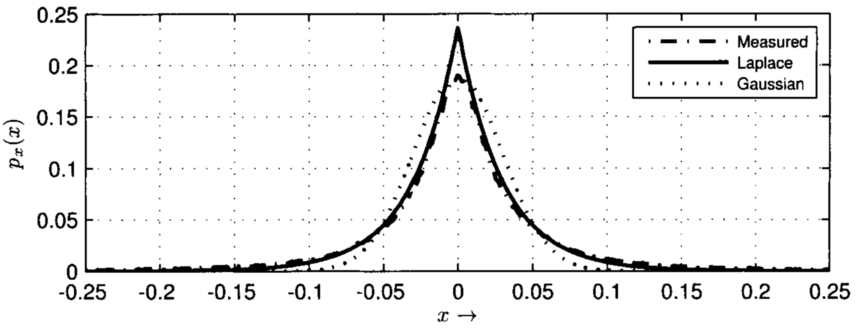

Figure 2.6 Probability density function estimated from a music signal compared to a Laplace distribution

at 0 amplitude (compare Fig. 2.6). The Laplace distribution is given by

$$p_l = \frac{1}{2\beta} \exp\left(-\frac{|x - \mu_x|}{\beta}\right). \tag{2.18}$$

If the input signal is a constant, for example, a DC offset, the PDF consists only of a single peak.

Comparing Figs. 2.5 (top right) and 2.6 makes it obvious that under the assumption that a PDF describes a signal sufficiently well, a sine wave is not a very good approximation of a real-world music or speech signal.

The PDF of quantized input signals has a limited set of amplitude classes as a quantized signal has only a limited set of amplitude values.

The PDF of a signal can be estimated from a sufficiently long block of samples by computing a histogram of signal amplitudes and dividing it by the number of observed samples. It is then sometimes referred to as *Relative Frequency Distribution (RFD)*. For the sake of simplicity the PDF and the RFD will not be differentiated in the following.

2.2 Signal Processing

The following chapters introduce methods for processing digital signals. There exists a vast amount of introductory literature on *digital signal processing* — as mentioned above we will not attempt to reiterate the theory and methods but instead give short summaries on the most important foundations for ACA.

2.2.1 Convolution

Every linear, time-invariant system can be completely described by its *impulse response* $h(i)$, which is the output of a system for an impulse as an input signal. Examples of such systems (or systems which can be approximated as such systems) are filters, speakers, microphones, rooms, etc. The output $y(i)$ of a discrete linear and time-invariant system can be computed from input signal $x(i)$ and the system's impulse response $h(i)$ of length

Convolution Properties

- Identity for convolution with the delta function $\delta(i)$:

$$x(i) = x(i) * \delta(i) \tag{2.19}$$

- Commutativity:

$$y(i) = x(i) * h(i) = h(i) * x(i) \tag{2.20}$$

- Associativity:

$$g(i) * h(i) * x(i) = \big(g(i) * h(i)\big) * x(i) = g(i) * \big(h(i) * x(i)\big) \tag{2.21}$$

- Distributivity:

$$g(i) * \big(h(i) + x(i)\big) = \big(g(i) * h(i)\big) + \big(g(i) * x(i)\big) \tag{2.22}$$

- Circularity: If $h(i)$ is periodic, then the convolution result $y(i) = h(i) * x(i)$ will also be periodic.

$\mathcal{J}$ by the *convolution* operation:

$$y(i) = x(i) * h(i) = \sum_{j=0}^{\mathcal{J}-1} h(j) \cdot x(i-j). \tag{2.23}$$

The most important properties of the convolution operation are summarized in Eqs. (2.19)–(2.22). Derivations of these properties can be found in Appendix A.

2.2.1.1 Simple Filter Examples

One typical linear system in digital signal processing is the filter. Every filtering process is a convolution, but for practical filter implementations this is only of interest for filters with a finite length impulse response, so-called *Finite Impulse Response (FIR)* filters. Filters with an infinite length impulse response are unsurprisingly referred to as *Infinite Impulse Response (IIR)* filters and have a feedback path which complicates the determination of the impulse response in comparison with FIR filters for which the impulse response simply equals its filter coefficients.

In the following, two simple and frequently used digital low-pass filters will be presented, the *Moving Average (MA)* filter as a prototypical FIR filter and the single-pole filter as a simple IIR filter.

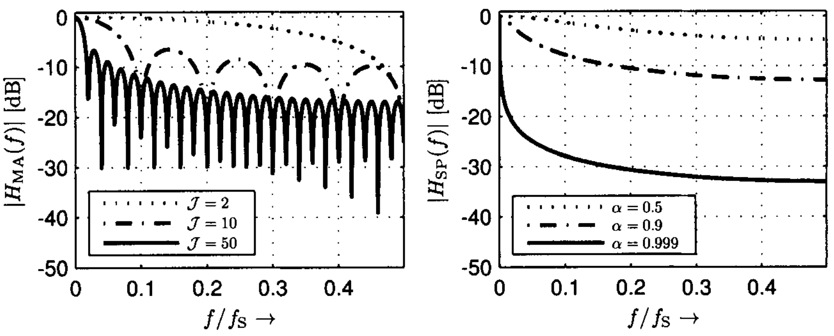

Figure 2.7 Magnitude frequency response of a moving average low-pass filter with the impulse response lengths of 2, 10, and 50 (left) and frequency response of a single-pole low-pass filter with an α of 0.5, 0.9, and 0.999 (right)

Moving Average Filter

The filter equation for an MA filter with an impulse response of length $\mathcal{J}$ is

$$y(i) = \sum_{j=0}^{\mathcal{J}-1} b(j) \cdot x(i-j). \tag{2.24}$$

The coefficients $b(j)$ of a typical MA filter have two main properties; the filter coefficients are identical:

$$b(0) = b(j) \quad \text{for } 0 \leq j \leq \mathcal{J}-1, \tag{2.25}$$

and the sum of all coefficients is normalized to 1:

$$\sum_{j=0}^{\mathcal{J}-1} b(j) = 1. \tag{2.26}$$

Equations (2.25) and (2.26) result in the coefficients $b(j)$ being identical and normalized to $b(j) = {}^1/_{\mathcal{J}}$. Alternatively they can also be weighted by an arbitrary window function. Typical window functions are symmetric and weight center samples higher than lateral samples. The window shape allows to elongate the impulse response. One of the simplest window shapes would be triangular.

Applying an MA filter with unweighted coefficients twice to the signal is equivalent to applying an MA filter with a triangular shaped impulse response of double length. Thus, the repeated use of an MA filter is similar to increasing the impulse response length and applying a window function to the coefficients.

The MA filter's cut-off frequency, the frequency at which the magnitude first drops by 3 dB, will decrease with an increasing number of coefficients as depicted in Fig. 2.7 (left). Smith gives a detailed introduction to MA filters in [15].

The number of multiply and add operations per hop increases linearly with the impulse response length of the MA filter. If no window function had been applied to the coefficients, i.e., all coefficients are identical $b(k) = b$, the number of operations can be drastically reduced by implementing the filter recursively:

$$\begin{aligned} y(i) &= \sum_{j=0}^{\mathcal{J}-1} b \cdot x(i-j) \\ &= b \cdot \big(x(i) - x(i-\mathcal{J})\big) + \underbrace{\sum_{j=1}^{\mathcal{J}} b \cdot x(i-j)}_{y(i-1)} \\ &= b \cdot \big(x(i) - x(i-\mathcal{J})\big) + y(i-1). \end{aligned} \tag{2.27}$$

A recursive implementation is not applicable with weighted filter coefficients.

If the filter coefficients of the MA filter are symmetric with $h(i) = h(\mathcal{J} - 1 - i)$ (which is only possible for an FIR filter), then the filter will have a linear phase response and thus a constant group delay.

Single-Pole Low-Pass Filter

The *single-pole filter* is — mainly due to its simplicity and efficiency — one of the most frequently used smoothing and low-pass filter. The filter equation is

$$y(i) = \alpha \cdot y(i-1) + (1-\alpha) \cdot x(i) \quad \text{with } 0 \le \alpha < 1. \tag{2.28}$$

Figure 2.7 (right) shows the magnitude of the frequency response for three different α. The coefficient α depends on the required integration time T_{I}; if the integration is defined to be the time required for the filter's response on a step function to rise from 10% to 90%,[2] the coefficient is

$$\alpha = \exp\left(\frac{-2.2}{f_{\mathrm{S}} \cdot T_{\mathrm{I}}}\right). \tag{2.29}$$

The cut-off frequency is then

$$\omega_{\mathrm{G}}(\alpha) = \arccos\big(2 - \cosh\left(\log(\alpha)\right)\big) \tag{2.30}$$

which is only defined for $\alpha \ge e^{-\operatorname{arccosh}(3)}$.

2.2.1.2 Zero Phase Filtering with IIRs

No non-trivial causal system provides a *zero phase response*. A zero phase response means that the group delay at each frequency is zero as well. Zero phase response systems output the unmodified timing characteristics of each individual frequency group.

In the case of linear phase FIR filters a zero phase response can be reconstructed in an anti-causal way by removing the introduced delay from the filter output (which could also be interpreted as moving the impulse response to be symmetric around time 0).

In the case of IIR filters, however, this procedure is not applicable. In a non-real-time environment in which the whole input file is available, it is, however, possible to generate a filtered signal with zero phase shift by using an IIR filter. This can be done by applying the filter twice in series to the signal while using the time-reversed input signal the second time. The second filter run has the effect of leveling out the phase shifts introduced by the first run. The magnitude response of the complete filter process is then the filter's squared magnitude response. Figure 2.8 visualizes the intermediate and the final result of this process in the time domain.

[2]Note that there exist other approaches to defining the integration time that may differ significantly.

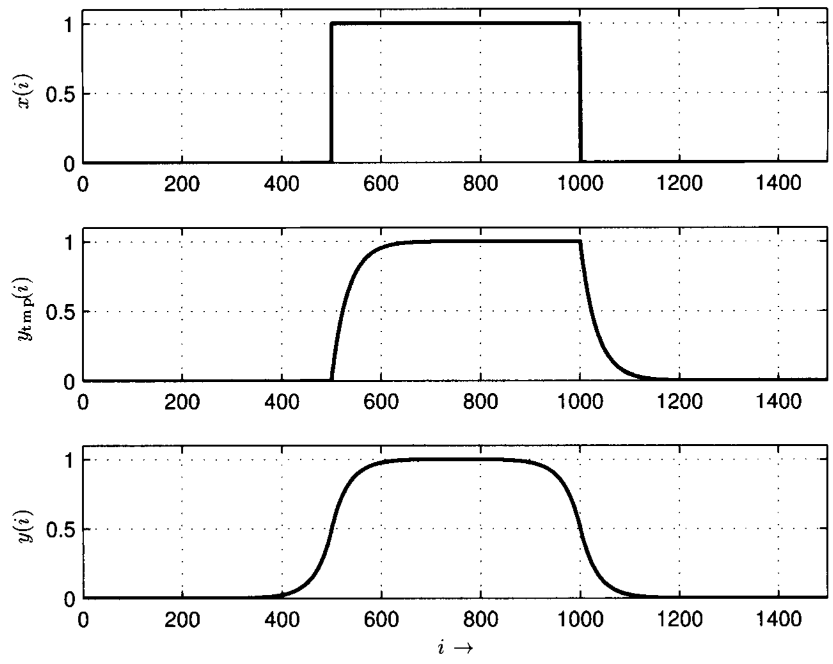

Figure 2.8 Example of zero phase filtering: input signal (top), low-pass filtered signal after the first forward pass (mid), and final result of both forward and backward paths (bottom)

To explain this behavior mathematically, the following property is of importance:

$$\mathfrak{F}\left\{h(-i)\right\} = H\left(-\mathrm{j}\omega\right), \tag{2.31}$$

meaning that the *Fourier Transform (FT)* $\mathfrak{F}$ (see Sect. 2.2.3) of a time-reversed signal equals the complex-conjugate of the transform of the original signal. The flip function is a tool which helps us in the derivation:

$$\mathrm{flip}\left(x(i)\right) = x(-i). \tag{2.32}$$

The output signal $y(i)$ is computed via the temporary output $y_{\mathrm{tmp}}(i) = h(i) * x(i)$ by

$$\begin{aligned} y(i) &= \mathrm{flip}\left(h(i) * y_{\mathrm{tmp}}(-i)\right) &(2.33)\\ &= h(-i) * y_{\mathrm{tmp}}(i) &(2.34)\\ &= h(-i) * \left(h(i) * x(i)\right). &(2.35)\end{aligned}$$

It follows in the Fourier domain:

$$\begin{aligned} Y(\mathrm{j}\omega) &= H(-\mathrm{j}\omega)\cdot\left(H(\mathrm{j}\omega)\cdot X(\mathrm{j}\omega)\right) &(2.36)\\ &= |H(\mathrm{j}\omega)|^2\cdot X(\mathrm{j}\omega). &(2.37)\end{aligned}$$

2.2.2 Block-Based Processing

Signal processing algorithms usually work block-based, meaning that the input signal is being split into consecutive blocks of frames with length $\mathcal{K}$. These blocks are processed

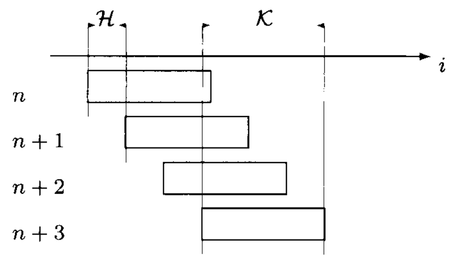

Figure 2.9 Schematic visualization of block-based processing: the input signal with the input sample index i is split into overlapping blocks of length $\mathcal{K}$ and block index n; the hop size $\mathcal{H}$ is the distance from the start of one block and the following block

one after another. In extreme cases, the block lengths can be either $\mathcal{K} = 1$ or equal the length of the processed audio file, but in most cases $\mathcal{K}$ is chosen arbitrarily between $\mathcal{K} = 32$ and $\mathcal{K} = 16,384$ (in many cases powers of 2 are an advantage). Some reasons for this *block-based processing* are

- the internal usage of block-based algorithms such as the *Discrete Fourier Transform (DFT)* (see Sect. 2.2.3 and Appendix B),
- the characteristics of the used audio *Input/Output (IO)* hardware that usually returns blocks of samples,
- reduced memory allocation (instead of the complete audio file length; only the block length has to be allocated), and
- computational efficiency compared to individual sample processing by avoiding function call overhead and vectorization with *Single Instruction Multiple Data (SIMD)* operations to optimize workload.

Systems working in *real time* have the constraint that the system has to be able to perform the processing of one block of data during the time frame this block consumes. Another condition of such systems is that they have no access to future input samples but only the present and possibly past samples. For real-time systems, the *latency* of the system, i.e., the delay between an input sample and the corresponding output value, can never be lower than the block length of the algorithm. The block length used to call the algorithm does not necessarily correspond to the algorithm's internal block length. If the input block length is not a multiple of the internal block length, then efficiency problems can arise since the computational workload cannot be distributed uniformly over the input blocks, resulting in occasional workload peaks.

Internally, most audio algorithms use overlapping blocks of data as depicted in Fig. 2.9. That means that the boundaries of subsequently processed blocks overlap, i.e. that the last sample of the first block is after the first sample of the second block. The sample indices of the block boundaries start sample $i_\mathrm{s}(n)$ and stop sample $i_\mathrm{e}(n)$ of processing block n can then be formulated as

$$i_\mathrm{s}(n) = i_\mathrm{s}(n-1) + \mathcal{H}, \tag{2.38}$$

$$i_\mathrm{e}(n) = i_\mathrm{s}(n) + \mathcal{K} - 1, \tag{2.39}$$

with $\mathcal{H}$ being the so-called *hop size* in samples. The hop size should be smaller than or equal the block length. The *block overlap ratio* can be calculated with

$$o_r = \frac{\mathcal{K} - \mathcal{H}}{\mathcal{K}}. \tag{2.40}$$

A typical block overlap ratio is $o_r \geq {}^1\!/_2$. When only one result $v(n)$ is computed per processing block, the assigned time stamp in samples $t_s(n)$ would usually be either the start frame of the block $t_s(n) = {}^{i_s(n)}\!/_{f_S}$ or — more common — its middle position:

$$t_s(n) = \frac{i_e(n) - i_s(n) + 1}{2 \cdot f_S} + \frac{i_s(n)}{f_S} = \frac{\mathcal{K}}{2 \cdot f_S} + \frac{i_s(n)}{f_S}. \tag{2.41}$$

If the time stamp $t_s(n)$ is chosen to be in the middle of the block, the practical problem arises that the start and end time stamps always represent a shorter time range than the audio signal itself. For example, if the audio signal starts at sample index 0, the first time stamp will be at ${}^{(\mathcal{K}-1)}\!/_2$. This is usually avoided by padding the signal with ${}^{(\mathcal{K}-1)}\!/_2$ zeros at beginning and end. However, in that case the results may not be reliable for the first and the last block because the computation is done with a smaller number of audio samples.

Some algorithms do not only compute results per block of audio data but combine the extracted results per block $v(n)$ again in a *texture window*. Texture windows may in turn overlap depending on texture window length and texture window hop. In this case, the same logic as above is followed with the difference that the window boundaries are now block indices instead of frame indices. Texture windows can be used to compute, e.g., windowed statistical descriptions from the series of results $v(n)$.

2.2.3 Fourier Transform

The *DFT* is one of the most important tools for processing and analyzing audio signals. In the following, the important properties and characteristics of this widely used transform are listed. A more detailed explanation — starting with the FT of continuous signals — can be found in Appendix B. The DFT of a block of the signal $x(i)$ is defined by

$$X(\mathrm{j}\Delta\Omega) = \mathfrak{F}\{x(i)\} = \sum_{i=0}^{\mathcal{K}-1} x(i) e^{-\mathrm{j}ki\Delta\Omega} \tag{2.42}$$

with $\Delta\Omega$ being the distance between two frequency bins as angular frequency difference

$$\Delta\Omega = \frac{2\pi}{\mathcal{K} \cdot T_S} = \frac{2\pi \cdot f_S}{\mathcal{K}}. \tag{2.43}$$

The frequency of bin index k can then be computed from the block length $\mathcal{K}$ and sample rate f_S by

$$f(k) = \frac{\Delta\Omega}{2\pi} k = \frac{f_S}{\mathcal{K}} k. \tag{2.44}$$

We will refer to the DFT of the nth block with length $\mathcal{K} = i_e(n) - i_s(n) + 1$ of the audio data as *Short Time Fourier Transform (STFT)*:

$$X(k, n) = \sum_{i=i_s(n)}^{i_e(n)} x(i) \exp\left(-\mathrm{j}k \cdot (i - i_s(n)) \frac{2\pi}{\mathcal{K}}\right). \tag{2.45}$$

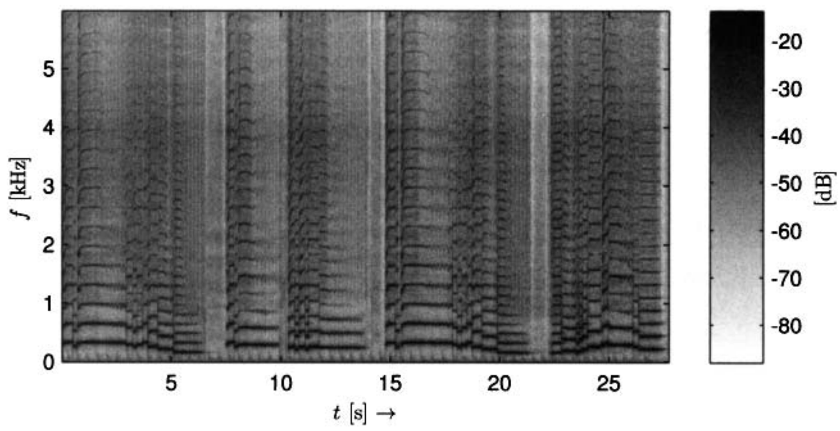

Figure 2.10 Spectrogram of a monophonic saxophone signal: each column is (the magnitude of) an STFT of a block with the time increasing from left to right; the level of each specific point in frequency and time is visualized by the darkness of the point

The most important properties of the STFT are listed on Pg. 22. See Appendix B for derivations.

A typical visualization of an audio signal that combines time and frequency components is the *spectrogram* as shown in Fig. 2.10. An STFT is calculated for each (overlapping) block of sample data; the resulting STFTs are then plotted in a pseudo-three-dimensional image in which (the magnitude of) each STFT is represented by a column and each bin is darkened according to its level.

2.2.3.1 Frequency Reassignment: Instantaneous Frequency

Frequency reassignment is a method to virtually improve the spectral resolution by mapping the frequency data to coordinates closer to its "true" coordinates. Frequency reassignment is not a fundamental frequency detection technique but can help to increase the frequency accuracy of spectral frequency analysis by utilizing the phase spectrum.

The reassignment process may basically cover both frequency reassignment as well as time reassignment. For an overview of (time-) frequency reassignment, see Fulop and Fitz [16] and Lagrange [17]. The focus will be exclusively on frequency reassignment in the following.

The basic idea of frequency reassignment is that it is possible to estimate the frequency of a signal via its phase. The frequency is the derivative of the phase $\Phi(k,t)$:

$$\omega(k,t) = \frac{\partial}{\partial t}\Phi(k,t). \tag{2.46}$$

Two common approaches are used to compute the so-called *instantaneous frequency* from the discrete STFT. The first one is to literally compute the phase difference of consecutive STFT spectra, and the second one is to compute two STFTs with different windows, namely with the second window being the derivative of the first window.

Fourier Transform Properties

- Invertibility

$$x\big(i_s(n)\dots i_e(n)\big) = \mathfrak{F}^{-1}\left\{X(k,n)\right\} = \frac{1}{\mathcal{K}}\sum_{0}^{\mathcal{K}-1} X(k,n)\cdot \exp\left(jki\frac{2\pi}{\mathcal{K}}\right). \tag{2.47}$$

- Linearity and Superposition

$$\begin{aligned}&\mathfrak{F}\left\{c_1\cdot x\big(i_s(n)\dots i_e(n)\big) + c_2\cdot y\big(i_s(n)\dots i_e(n)\big)\right\}\\ &\qquad = c_1\cdot X(k,n) + c_2\cdot Y(k,n).\end{aligned} \tag{2.48}$$

- Periodicity

$$X(k+\mathcal{K},n) = X(k,n). \tag{2.49}$$

- Symmetry [for real $x(i)$]

$$X(\mathcal{K}-k,n) = X^*(k,n). \tag{2.50}$$

- Circularity

$$\mathfrak{F}^{-1}\left\{H(k,n)\cdot X(k,n)\right\} = h'\big(i_s(n)\dots i_e(n)\big) * x\big(i_s(n)\dots i_e(n)\big) \tag{2.51}$$

with h' being a *periodically extended* sequence of the signal block with the sample boundaries $i_s(n)$ and $i_e(n)$.

- Time and Frequency Shifting

$$\begin{aligned}\mathfrak{F}\left\{x(i-i_0)\right\} &= X(k,n)\cdot \exp\left(-j\frac{2\pi k}{\mathcal{K}}i_0\right) &(2.52)\\ \mathfrak{F}^{-1}\left\{X(k-k_0,n)\right\} &= x\big(i_s(n)\dots i_e(n)\big)\cdot \exp\left(j\frac{2\pi i}{\mathcal{K}}k_0\right). &(2.53)\end{aligned}$$

- Time and Frequency Scaling

$$\mathfrak{F}\left\{x(c\cdot i)\right\} = \frac{1}{|c|}X\left(\frac{k}{c},n\right). \tag{2.54}$$

- Parseval's Theorem

$$\sum_{i=i_s(n)}^{i_e(n)} |x(i)|^2 = \frac{1}{\mathcal{K}}\sum_{k=0}^{\mathcal{K}-1}|X(k,n)|^2. \tag{2.55}$$

- Duality

$$\mathfrak{F}\left\{X(i)\right\} = \frac{1}{\mathcal{K}}x(k,n). \tag{2.56}$$

Explicit Phase Difference

The derivation operation from Eq. (2.46) can be easily approximated by computing the difference of the phases of consecutive STFT phases:

$$\omega_{\mathrm{I}}(k,n) = \frac{\Delta\Phi_{\mathrm{u}}(k,n)}{\mathcal{H}} \cdot f_{\mathrm{S}}. \tag{2.57}$$

The actual computation, however, is not as trivial as it seems at first glance as $\Delta\Phi_{\mathrm{u}}(k,n)$ represents the *unwrapped* phase difference while the computed phase spectrum gives the *wrapped* phase in the range of $]-\pi;\pi]$. In order to estimate the unwrapped phase difference we first compute the estimate of the unwrapped phase of the current analysis block by

$$\hat{\Phi}(k,n) = \Phi(k,n-1) + \frac{2\pi k}{\mathcal{K}}\mathcal{H}. \tag{2.58}$$

Since the phase $\Phi(k,n)$ is in the range of $]-\pi;\pi]$, the unwrapped phase can then be formulated as

$$\Phi_{\mathrm{u}}(k,n) = \hat{\Phi}(k,n) + \mathrm{princarg}\left[\Phi(k,n) - \hat{\Phi}(k,n)\right] \tag{2.59}$$

with the princarg function being the principal argument of the phase with a resulting range of $]-\pi;\pi]$. The phase difference is then

$$\begin{aligned}\Delta\Phi_{\mathrm{u}}(k,n) &= \Phi_{\mathrm{u}}(k,n) - \Phi(k,n-1)\\ &= \hat{\Phi}(k,n) + \mathrm{princarg}\left[\Phi(k,n) - \hat{\Phi}(k,n)\right] - \Phi(k,n-1).\end{aligned}$$

The final result can be retrieved by substituting $\hat{\Phi}(k,n)$ with Eq. (2.58):

$$\Delta\Phi_{\mathrm{u}}(k,n) = \frac{2\pi k}{\mathcal{K}}\mathcal{H} + \mathrm{princarg}\left[\Phi(k,n) - \Phi(k,n-1) - \frac{2\pi k}{\mathcal{K}}\mathcal{H}\right]. \tag{2.60}$$

To improve the reliability of the phase unwrapping process the hop size $\mathcal{H}$ should be as small as possible.

Phase Difference Using Transforms

An alternative to computing the phase difference directly is to use two different windows for computing two STFTs of one analysis block. Then, the instantaneous frequency can be computed by

$$\omega_{\mathrm{I}}(k,n) = \omega(k) + \Im\left\{\frac{X_D(k,n)\cdot X^*(k,n)}{|X(k,n)|^2}\right\} \tag{2.61}$$

with $X_D(k,n)$ being the STFT computed using the derivative of the window used for the calculation of $X(k,n)$ [18]. This approach is independent of the hop size but requires the computation of two DFTs per block.

2.2.4 Constant Q Transform

The *Constant Q Transform (CQT)* was introduced to overcome the insufficient frequency resolution of the STFT at low frequencies for common STFT lengths. The CQT calculates frequency coefficients similar to the DFT but on a logarithmic frequency scale [19]:

$$X(k,n) = \frac{1}{\mathcal{K}(k)}\sum_{i=i_{\mathrm{s}}(n)}^{i_{\mathrm{e}}(n)} w_k(i-i_{\mathrm{s}})\cdot x(i)\exp\left(\mathrm{j}2\pi\frac{Q\cdot(i-i_{\mathrm{s}})}{\mathcal{K}(k)}\right). \tag{2.62}$$

$\mathcal{Q}$ adjusts the desired frequency resolution per octave c, while $\mathcal{K}(k)$ depends on both the target frequency and the *quality factor* $\mathcal{Q}$:

$$\mathcal{Q} = \frac{f}{\Delta f} = \frac{1}{2^{1/c} - 1}, \tag{2.63}$$

$$\mathcal{K}(k) = \frac{f_S}{f(k)} \mathcal{Q}. \tag{2.64}$$

The CQT can be interpreted as computing a DFT only for specific, logarithmically spaced, frequency bins. It is not invertible and can in terms of efficiency be compared to a DFT [as opposed to the *Fast Fourier Transform (FFT)*]. There are approaches to make the CQT both computationally more efficient and quasi-invertible, see, e.g., [20, 21].

2.2.5 Auditory Filterbanks

An *auditory filterbank* is frequently used in psycho-acoustical or physiological ear models. It approximates the resolution and selectivity of the human ear. The filterbank's mid-frequencies are usually computed according to the mel scale (see Sect. 5.1).

In signal processing applications the usage of auditory filterbanks is not as widespread as one might assume. The most obvious reasons are (a) the computational workload produced by a filterbank with reasonable frequency resolution and (b) the restriction to analysis tasks since the computed frequency domain representation cannot be transformed back to the time domain (except for a few carefully designed filterbanks with other limitations). In the case of ACA it has not been consistently shown that approaches using a filterbank give more reliable results than, e.g., an STFT-based analysis. One possible explanation is that in many cases the subject of investigation is not what humans are able to hear in a signal but physical properties of the signal. Still the argument that ultimately every audio analysis task is about the extraction of *perceivable* features has merit as well. Auditory modeling is a lively subject of research; Lyon et al. review different approaches in [22].

2.2.5.1 Gammatone Filterbank

A widely used auditory filterbank is the so-called *gammatone filterbank*. A gammatone filter has an impulse response of the following form [23, 24]:

$$h(i) = \frac{a \cdot \left({}^{i}/_{f_S}\right)^{\mathcal{O}-1} \cdot \cos\left(2\pi \cdot f_c \frac{i}{f_S}\right)}{e^{2\pi i \Delta f / f_S}} \tag{2.65}$$

Figure 2.11 shows the impulse response of such a gammatone filter with the following parametrization: center frequency $f_c = 1\,\text{kHz}$, bandwidth $\Delta f = 125\,\text{Hz}$, and order $\mathcal{O} = 4$.

Slaney showed that gammatone filters can be efficiently implemented with cascaded second-order filters [25]. A survey of different gammatone filter implementations can be found in [26].

Figure 2.12 shows the frequency (magnitude) response of a gammatone filterbank as computed by the *Auditory Toolbox* [27] for 20 bands between 100 Hz and 24 kHz.

2.2.6 Correlation Function

The *correlation function* can be used to compute the similarity between two signals at a lag η. Given two signals $x(i)$ and $y(i)$, the so-called *Cross Correlation Function (CCF)* is

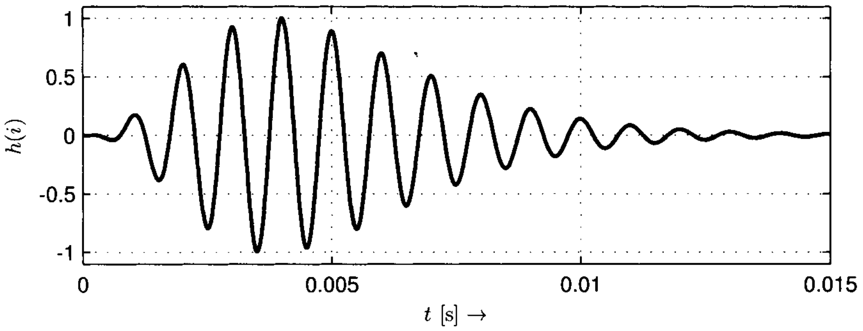

Figure 2.11 Impulse response of a gammatone filter with a center frequency $f_c = 1\,\text{kHz}$, a bandwidth $\Delta f = 125\,\text{Hz}$, and an order $\mathcal{O} = 4$

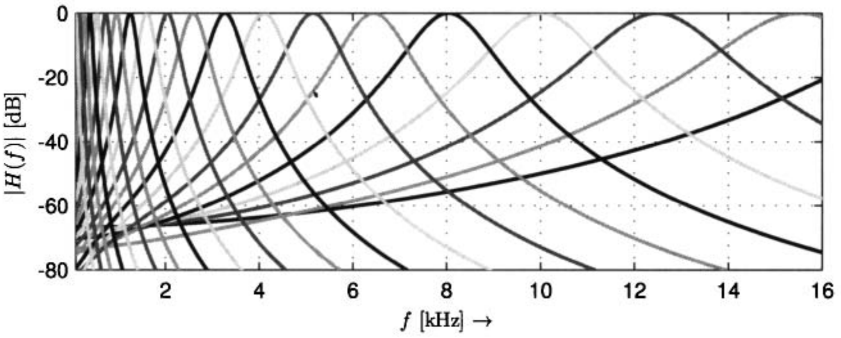

Figure 2.12 Magnitude frequency response of a gammatone filterbank with 20 bands between 100 Hz and 24 kHz

defined by

$$r_{xy}(\eta) = \sum_{i=-\infty}^{\infty} x(i) \cdot y(i+\eta). \tag{2.66}$$

Assuming the input signals are only blocks of $\mathcal{K} = i_e(n) - i_s(n) + 1$ samples, we can imagine an infinite number of zeros to the right and to the left of the actual samples; Eq. (2.66) can still be applied (although in a real application the multiplications with zero can obviously be omitted) and the CCF of two blocks of samples is thus

$$r_{xy}(\eta, n) = \sum_{i=i_s(n)}^{i_e(n)-\eta} x(i) \cdot y(i+\eta) \tag{2.67}$$

and will equal zero for all lags larger than $\mathcal{K} = i_e(n) - i_s(n) + 1$:

$$r_{xy}(\eta, n) = 0 \quad \text{if } |\eta| \geq \mathcal{K}. \tag{2.68}$$

Another possible interpretation of the block-wise CCF is the infinite correlation function of signals multiplied with a rectangular window $w_{\mathrm{R}}(i)$:

$$r_{xy}(\eta, n) = \sum_{i=-\infty}^{\infty} x(i) w_{\mathrm{R}}(i) \cdot y(i+\eta) w_{\mathrm{R}}(i+\eta). \tag{2.69}$$

2.2.6.1 Normalization

The correlation function is usually multiplied by a normalization factor λ_{c}. If only a block of samples of the signals $x(i)$ and $y(i)$ is being considered, then

$$\lambda_{\mathrm{c}} = \frac{1}{\sqrt{\left(\sum\limits_{i=i_{\mathrm{s}}(n)}^{i_{\mathrm{e}}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_{\mathrm{s}}(n)}^{i_{\mathrm{e}}(n)} y^2(i)\right)}}. \tag{2.70}$$

The number of non-zero multiplications decreases linearly with increasing lag until it is only one for $\eta = K - 1$. Therefore, the resulting correlation function is shaped triangular with the shape's maximum at lag $\eta = 0$. This triangular weighting can be avoided by taking into account the current lag η for the normalization:

$$\lambda_{\mathrm{c}}(\eta) = \frac{\mathcal{K}}{(\mathcal{K} - |\eta|) \cdot \sqrt{\left(\sum\limits_{i=i_{\mathrm{s}}(n)}^{i_{\mathrm{e}}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_{\mathrm{s}}(n)}^{i_{\mathrm{e}}(n)} y^2(i)\right)}}. \tag{2.71}$$

For large lags η the results will, however, become unreliable due to the limited amount of samples from which they are calculated.

A different way of avoiding the triangular shape of the correlation function is to use signal blocks of different sizes, namely a block length of $3\mathcal{K}$ for the signal x and a block length of $\mathcal{K}$ samples for the signal y. Typically, the number of samples for x is increased by appending $\mathcal{K}$ preceding and succeeding samples, respectively. While this leads to a extended theoretical non-zero range of r_{xy} for $|\eta| \leq 2\mathcal{K}$, the results for $|\eta| > \mathcal{K}$ are usually discarded. The normalization procedure is in this case unclear, although three options present themselves:

- *normalization per lag* requiring the computation of the denominator of Eq. (2.70) for every individual lag,
- *overall normalization* by computing the denominator from unequal length sequences, and
- *two-stage normalization* by first finding the maximum of the unnormalized cross correlation and then computing the denominator for two short length sequences around the maximum.

If only the current block of samples from signal x is available, this block may also be copied and pasted to the start and end of this block, resulting in the same amount of $3\mathcal{K}$ samples. If the block of samples from signal x would theoretically be pasted an infinite number of times, then the signal would be periodic with $\mathcal{K}$, which in turn would result in the periodicity of the correlation function: $r_{xy}(\eta) = r_{xy}(\eta + \mathcal{K})$. This is then called the *Circular Correlation Function (CiCF)*; the circularity aspects have to be taken into account when the CCF is computed in the frequency domain as described in Sect. 2.2.6.4.

Autocorrelation Function Properties

- *Autocorrelation Function (ACF)* at lag 0:
  $r_{xx}(0, n) = 1$ when normalized according to Eq. (2.70), otherwise it is the *Root Mean Square (RMS)* (see Sect. 4.3.1) of the block.

- Maximum:
  $|r_{xx}(\eta, n)| \leq r_{xx}(0, n)$ since the signal can at no lag be more similar to itself than without lag ($\eta = 0$).

- Symmetry:
  $r_{xx}(\eta, n) = r_{xx}(-\eta, n)$. The ACF is symmetric around lag $\eta = 0$. This means that the calculation of negative lags can be omitted for reasons of computational efficiency.

- Periodicity:
  The ACF of a periodic signal will be periodic with the period length of the input signal.

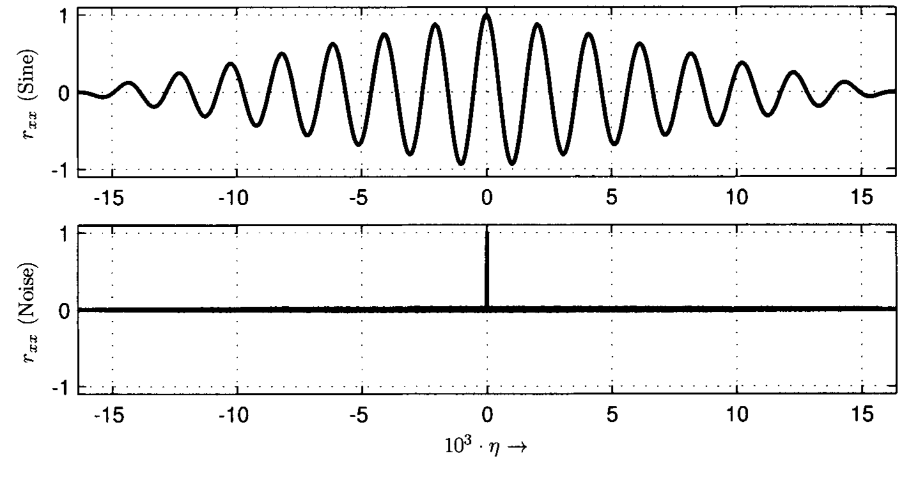

Figure 2.13 ACF of a sinusoid (top) and white noise (bottom)

2.2.6.2 Autocorrelation Function

The ACF is a special case of the correlation function with identical input signals $x(i) = y(i)$ and is therefore referred to as $r_{xx}(\eta)$. It is a measure of self-similarity, and its properties are summarized in the box above.

Figure 2.13 shows the ACF computed with a block of a sinusoid and a block of white noise.

2.2.6.3 Applications

The CCF can be used to find a specific latency or offset between a signal and the delayed signal. The lag of the maximum of the CCF indicates the delay between the signals. A typical example would be a radar signal with its echo coming back after being reflected by an obstacle. Finding the maximum of the CCF between the sent and the reflected signal should indicate the transmission time and thus the distance.

A CCF is sometimes also computed between a real signal and a constructed signal to find out whether the original signal shows similarities to the synthetic signal. An example would be to compute the CCF of the envelope of a drum loop with a series a weighted peaks to find a pattern in the signal.

The ACF is usually used to find periodicities within the signal. A periodic input signal will lead to a period ACF with peaks at the length of the period and its integer multiples. The ACF can be thus used to find a (fundamental) frequency (see Sect. 5.3.3.2) or to find the time interval between beats in music.

2.2.6.4 *Calculation in the Frequency Domain*

Rewriting either the CCF or the ACF as a convolution operation reveals an important property:

$$\begin{aligned} r_{xx}(\tau) &= \int\limits_{-\infty}^{\infty} x(\tau) \cdot x(t+\tau)\, d\tau \\ &= x(\tau) * x(-\tau). \end{aligned} \tag{2.72}$$

Using Eqs. (B.7) and (B.25) it can be deduced that

$$\begin{aligned} \mathfrak{F}\{r_{xx}(\tau)\} &= R_{xx}(\mathrm{j}\omega) \\ &= \mathfrak{F}\{x(\tau) * x(-\tau)\} \\ &= X(\mathrm{j}\omega) \cdot X^*(\mathrm{j}\omega) \\ &= |X(\mathrm{j}\omega)|^2. \end{aligned} \tag{2.73}$$

This relation is called the *Wiener-Khinchin theorem.*

The computation of the ACF in the time domain can thus be replaced by a simple multiplication in the frequency domain preceded and followed by an FT and an inverse FT, respectively. For long blocks this will save computing cycles and reduce computational workload when using an FFT algorithm. Note that with blocks of sampled data the result will be a *circular* ACF if the transformed block of data has not been correctly padded with zeros before transforming it. More specifically, computing the CCF of two blocks of length $\mathcal{K}$ requires a minimum FFT length of $2\mathcal{K}$.

Frequency Domain Compression

In certain applications such as auditory processing it might be of interest to apply a nonlinear compression function to the magnitude spectrum before transforming it back. Tolonen and Karjalainen named such a combined approach the *generalized ACF* to be computed by (see Sect. 5.3.4.2, [28])

$$r_{xx}^{\beta}(\eta, n) = \mathfrak{F}^{-1}\left\{|X(\mathrm{j}\omega)|^{\beta}\right\}. \tag{2.74}$$

A value $\beta = 2$ would result in the normal ACF.

2.2.7 Linear Prediction

The idea of *linear prediction* is to use preceding (sample) values to estimate (or *predict*) future values. In its most common form, the predicted value $\hat{x}(i)$ is a linear combination of the preceding samples

$$\hat{x}(i) = \sum_{j=1}^{\mathcal{O}} b_j \cdot x(i-j). \tag{2.75}$$

The filter order $\mathcal{O}$ can vary between only a few coefficients and up to thousands of coefficients. To allow adaption to a time-variant signal $x(i)$, the predictor filter is usually updated for every sample block to ensure good predictive qualities. In that case the process is called *adaptive linear prediction.*

In order to estimate the coefficients b_j from the signal, the usual approach is to minimize the power of the *prediction error*

$$\begin{aligned} e_{\mathrm{P}}(i)^2 &= (x(i) - \hat{x}(i))^2 & (2.76) \\ &= \left(x(i) - \sum_{j=1}^{\mathcal{O}} b_j \cdot x(i-j) \right)^2 . & (2.77) \end{aligned}$$

The minimum of this power can be found by minimizing $\partial e_{\mathrm{P}}^2(i)/\partial b_j$. The prediction coefficients b_j can be computed by solving the following system of equations:

$$\sum_{j=1}^{\mathcal{O}} b_j \cdot r_{xx}(|j-\eta|) = -r_{xx}(\eta), \qquad 1 \leq \eta \leq \mathcal{O}. \tag{2.78}$$

An efficient numeric solution to this system of equations has been presented by Levinson and Durbin [29, 30].

A signal can only be predicted well if there are statistical relations between the individual samples. Noise is an example of a signal type that cannot be predicted at all; purely periodic signals, however, can theoretically be perfectly predicted. For music signals this means that tonal parts of the signal can be predicted well while noisy and non-periodic parts such as initial transients cannot be predicted.

Figure 2.14 plots the original signal (top), the predicted signal (middle), and the prediction error (bottom) for an adaptive predictor of order $\mathcal{O} = 20$ and a block length of 2048 samples. The source signal is an excerpt of a song of popular music. The plot illustrates two noteworthy properties of predicted signals. First, the overall power of the prediction error is significantly lower than the power of the input signal. This is used in predictive waveform coding algorithms such as *Adaptive Differential Pulse Code Modulation (ADPCM)*, which basically translates the lower power into a bit rate reduction [31]. Secondly, the amplitude of the prediction error is low during the quasi-periodic states of the input signal and spikes at transient and thus noisy segments of the signal.

A detailed introduction to linear prediction and autoregressive modeling can be found in [32].

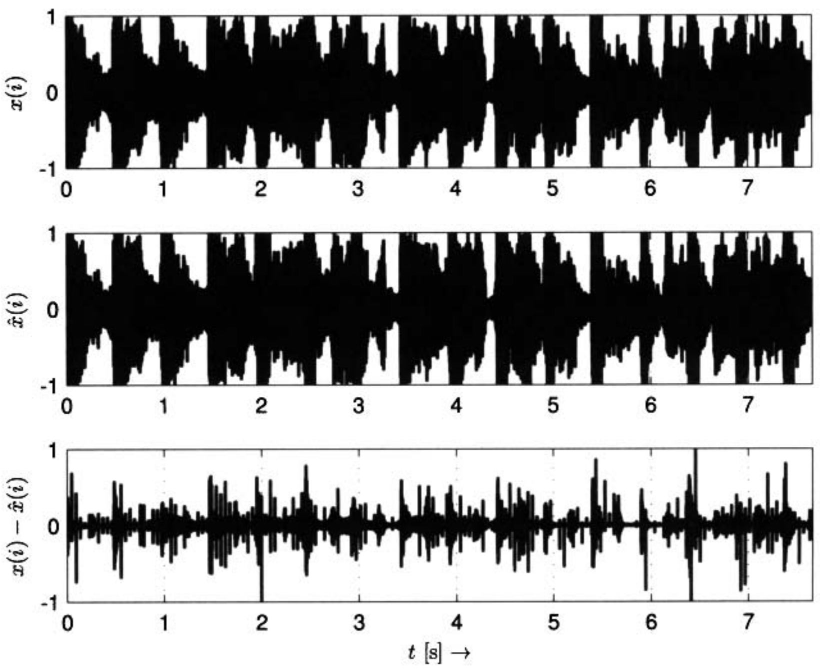

Figure 2.14 Plot of the original signal (top), the signal estimated with linear prediction (mid), and the resulting prediction error (bottom)