Part I

Analysis and Synthesis of Spatial Sound

1

Time–Frequency Processing: Methods and Tools

Juha Vilkamo[1] *and Tom Bäckström*[2]

[1]*Nokia Technologies, Finland*
[2]*Department of Signal Processing and Acoustics, Aalto University, Finland*

1.1 Introduction

In most audio applications, the purpose is to reproduce sounds for human listening, whereby it is essential to design and optimize systems for perceptual quality. To achieve such optimal quality with given resources, we often use principles in the processing of signals that are motivated by the processes involved in hearing. In the big picture, human hearing processes the sound entering the ears in frequency bands (Moore, 1995). The hearing is thus sensitive to the spectral content of ear canal signals, which changes quickly with time in a complex way. As a result of frequency-band processing, the ear is not particularly sensitive to small differences in weaker sounds in the presence of a stronger masking sound near in frequency and time to the weaker sound (Fastl and Zwicker, 2007). Therefore, a representation of audio signals where we have access to both time and frequency information is a well-motivated choice.

A prerequisite for efficient audio processing methods is a *representation* of the signal that presents features desirable to hearing in an accessible form and also allows high-quality playback of signals. Useful properties of such a representation are, for example, that its coefficients have physically or perceptually relevant interpretations, and that the coefficients can be processed independently from each other. The time–frequency domain is such a domain, and it is commonly used in audio processing (Smith, 2011). Spectral coefficients in this domain explain the signal content in terms of frequency components as a function of time, which is an intuitive and unambiguous physical interpretation. Moreover, time–frequency components are approximately uncorrelated, whereby they can be independently processed and the effect on the output is deterministic. These properties make the spectrum a popular domain for audio processing, and all the techniques discussed in this book utilize it. The first part of this chapter will give an overview of the theory and practice of the tools typically needed in time–frequency processing of audio channels.

The time–frequency domain is also useful when processing the spatial characteristics of sound, for example in microphone array processing. Differences in directions

Parametric Time–Frequency Domain Spatial Audio, First Edition. Edited by Ville Pulkki, Symeon Delikaris-Manias, and Archontis Politis.

Companion Website: www.wiley.com/go/pulkki/parametrictime-frequency

of arrival of wavefronts are visible as differences in time of arrival and amplitude between microphone signals. When the microphone signals are transformed to the time–frequency domain, the differences directly correspond to differences in phase and magnitude in a similar fashion to the way spatial cues used by a human listener are encoded in the ear canal signals (Blauert, 1997). The time–frequency domain differences between microphone channels have proven to be very useful in the capture, analysis, and reproduction of spatial audio, as is shown in the other chapters of this book. The second part of this chapter introduces a few signal processing techniques commonly used, and serves as background information for the reader.

This chapter assumes understanding of basic digital signal processing techniques from the reader, which can be obtained from such basic resources as Oppenheim and Schafer (1975) or Mitra and Kaiser (1993).

1.2 Time–Frequency Processing

1.2.1 Basic Structure

A block diagram of a typical parametric time–frequency processing algorithm is shown in Figure 1.1. The processing involves transforms between the time domain input signal $x_i(t)$, the time–frequency domain signal $x_i(k, n)$, and the time domain output signal $y_j(t)$, where t is the time index in the time domain, and k and n are indexes for the frequency and time frame in the time–frequency domain, respectively; i and j are then the channel indexes in the case of multi-channel input and/or output. Additionally, the processing involves short-time stochastic analysis and parameter-driven processing, where the time domain signal $y(k, n)$ is formed based on the parameters and $x(k, n)$. The parametric data consists of any information describing the frequency band signals, for example the stochastic properties, information based on the audio objects, or user input parameters. In some use cases, such as in parametric spatial audio coding decoders, the stochastic estimation block is not applied, and the processing acts entirely on the parametric data provided in the bit stream.

The parametric processing techniques typically operate on several different sampling rates: The sampling rate F_s of the wide-band signal, the sampling rate F_s/K of the

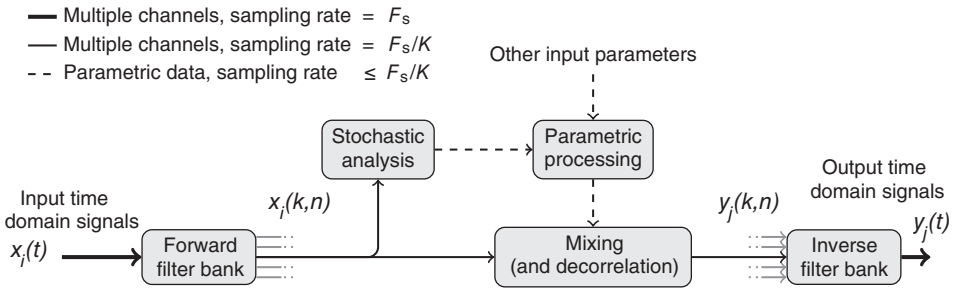

Figure 1.1 Block diagram of a typical parametric time–frequency processing algorithm. The processing operates on three sampling rates: that of the wide-band signal, that of the frequency band signal, and that of the parametric information.

frequency band signals, where K is the downsampling factor, and the sampling rate of the parametric information. Since the samples in the parametric information typically describe the signal properties over time frames, it potentially operates at a sampling rate below F_s/K. The parametric processing can also take place using a varying sampling rate, for example when the frame size adapts with the observed onsets of audio signals. In the following sections, the required background for the processing blocks in Figure 1.1 is discussed in detail.

Audio signals are generally time-varying signals whereby the spectrum is not constant in time. Should we analyze a long segment, then its spectrum would contain a mixture of all the different sounds within that segment. We could then not easily access the independent sounds, but only see their mixture, and the application of efficient processing methods would become difficult. It is therefore important to choose segments of the signal of such a length that we obtain good temporal separation of the audio content. Also, other properties such as constraints on algorithmic delay and requirements on spectral resolution impose demands on the length of analysis windows. It is then clear that while the spectrum or the frequency domain is an efficient domain for audio processing, the time axis also has to be factored in to the representation.

Computationally effective algorithms for time–frequency analysis have enabled their current popular usage. Namely, the basis of most time–frequency algorithms is the fast Fourier transform (FFT), which is a practical implementation of the discrete Fourier transform (DFT). It belongs to the class of super-fast algorithms that have an algorithmic complexity of $\mathcal{O}(N \log N)$, where N is the analysis window length. In practice, the signal is divided into partially overlapping time frames, each of which is processed with an FFT; this is commonly called a short-time Fourier transform (STFT; Smith, 2011). These methods are assumed to be known to the reader, and the rest of the chapter deals with methods based on filter banks, which are more common in practical audio applications.

1.2.2 Uniform Filter Banks

Figure 1.2 illustrates the generic design of uniform filter banks. The block diagram of the forward transform (the left-hand part of the figure) shows a set of band-pass analysis filters with different pass-band frequency regions, followed by downsampling operators, to produce the time–frequency data. The inverse transform (the right-hand side of the figure) is performed with upsampling operations, followed by band-pass synthesis filters. The block diagram is conceptual, and represents equivalent operations to practical

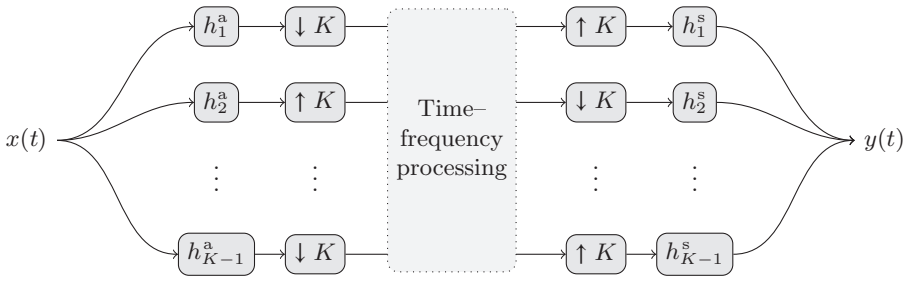

Figure 1.2 General block diagram of a uniform filter bank.

implementations that employ efficient digital signal processing structures, as discussed in Section 1.2.4.

Spatial sound processing techniques in the time–frequency domain typically perform the frequency decompositions with uniform filter banks, where the signal components appear on linear steps in frequency. A uniform grid is practical since it enables the application of efficient signal processing algorithms such as the FFT. From a perceptual viewpoint, *analysis* on a perceptually motivated non-uniform grid can be useful, but time–frequency *processing algorithms* can usually be applied on a uniform grid, since the effect of non-uniform sampling can, for most purposes, be achieved by scaling the magnitudes of spectral components (Bäckström, 2013). When a non-uniform grid is required, a uniform filter bank can be cascaded with filters at the lowest frequency bands to obtain the non-uniform properties (see Section 1.2.7).

Uniform filter banks can be over-, critically, or undersampled. A critically sampled filter bank preserves the overall sampling rate, which means that the sum of the sampling rates of the frequency band signals is the same as that of the input signal. This feature is desirable in coding applications where the objective is to reduce the bit rate by adaptive quantization of the spectral coefficients, whereby an increase in the overall data rate would be counterproductive. Critically sampled time–frequency transforms such as the modified discrete cosine transform are a fundamental part of the well-known MPEG Audio Layer III (mp3; Brandenburg, 1999) and the MPEG Advanced Audio Coding (AAC; ISO/IEC JTC1/SC29/WG11, 1997; Bosi *et al.*, 1997) standards.

Oversampled filter banks, on the other hand, have a higher combined sampling rate than the input signal. A typical oversampled filter bank in this scope is the complex-modulated filter bank, where representation of the signal using the real and the imaginary parts doubles the sampling rate; this is discussed further in Section 1.2.3. Such filter banks are used in processing methods where the objective is signal modification. An intuitive way to consider the sampling rate and the robustness in signal modification is to note that after the transform the frequency bands have a degree of spectral overlap. For robust independent processing of the frequency bands, the partially overlapping spectral data must be expressed by both of the adjacent bands.

Undersampled filter banks are also possible, though since they imply an inherent loss of information, they are generally useful only in analysis applications such as audio segmentation and voice activity detection. Such applications can compromise temporal accuracy in favor of a lower algorithmic complexity, whereby undersampled filter banks become a viable option.

Filter banks can be further categorized according to whether the frequency representation is real or complex valued. In processing applications where the signal is modified on purpose, we usually prefer complex-valued representations, since that gives us direct access to the phase of the signal. Since the phase of signals has a multitude of physical interpretations, complex-valued representations enable the design of algorithms that employ physically motivated techniques. However, since critical sampling is cumbersome with complex-valued filter banks, most audio coding methods rely on real-valued representations.

1.2.3 Prototype Filters and Modulation

The performance of a filter bank depends mainly on analysis and synthesis filters, h_k^{a} and h_k^{s}, respectively (see Figure 1.2), which are based on a designed prototype filter

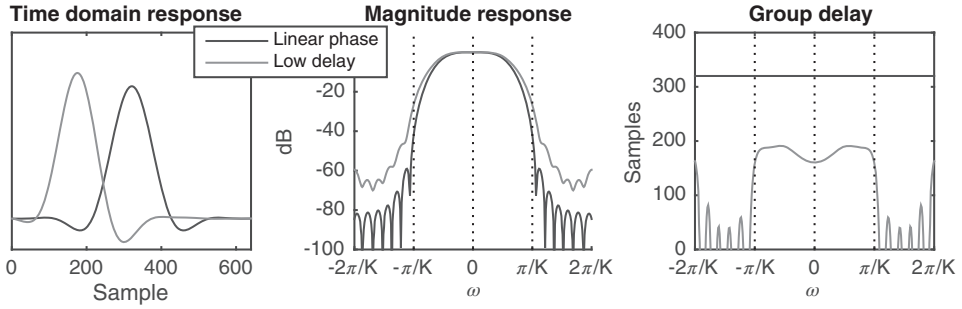

Figure 1.3 Linear-phase and low-delay prototype filters, using $K = 64$. Filter banks using a low-delay prototype filter are useful in applications where the constraints on delay are so stringent that the performance in terms of frequency response must be compromised.

h_0. They are band-pass filters that extract the desired frequency component from the input signal $x(n)$. The characteristics of these filters determine the performance of the filter bank in terms of frequency response. In this section, we discuss a straightforward approach which uses a linear-phase prototype filter. As an alternative, low-delay structures can be obtained using a prototype filter free from the linear-phase property, which requires a different synthesis filter bank structure (Schuller and Smith, 1995; Allamanche *et al.*, 1999). The linear-phase and low-delay prototype filters of the complex-modulated quadrature mirror filter bank type (QMF), applied in several codecs such as Herre *et al.* (2012, 2015) or 3GP (2014), are illustrated in Figure 1.3.

Let $h_0(t)$ be the coefficients of a linear-phase prototype filter and $H_0(e^{i\omega})$ its transfer function, where ω is the angular frequency. The objective in the design of prototype filters is to find a filter $H_0(e^{i\omega})$ such that

$$\begin{cases} |H(e^{i\omega})| & \approx 0 \quad \text{for } \omega > \pi/K, \\ |H(e^{i\omega})|^2 + |H(e^{i(\omega-\pi/K)})|^2 & \approx 1 \quad \text{for } 0 \le \omega \le \pi/K, \end{cases} \tag{1.1}$$

where K is the downsampling factor. In other words, the objective is to find a filter whose response at the stop-band is zero, and the response of the pass-band added with a shifted version of itself is unity. A set of pass-bands shifted by π/K then add up to unity (see Figure 1.4(a)). Specific methods for designing a linear-phase prototype filter can be found in, for example, Creusere and Mitra (1995) or Cruz-Roldán *et al.* (2002).

The band-pass filters $h_k(t)$ of uniform filter banks, where k is the frequency band index, are obtained by shifting the spectrum of the prototype filter using modulation. The complex-valued modulation can be expressed by

$$h_k(t) = h_0(t)e^{i\omega_k t + \psi_k}, \tag{1.2}$$

where $\omega_k = \pi(k + \alpha)/K$ is the angular frequency, α is an offset constant, and ψ_k determines the phase of the modulator. For example, defining $\alpha = 0$ means that the band $k = 0$ is centered at the DC frequency $\omega_0 = 0$, whereas defining $\alpha = 0.5$ means a half-band shift. The real-valued modulation is given by

$$h_k(t) = h_0(t)\cos(i\omega_k t + \psi_k) = h_0(t)\frac{e^{i\omega_k t+\psi_k} + e^{-i\omega_k t-\psi_k}}{2}. \tag{1.3}$$

The characteristics of complex- and real-valued modulations are illustrated in Figure 1.4. If the prototype filters are complex modulated, the resulting band-pass

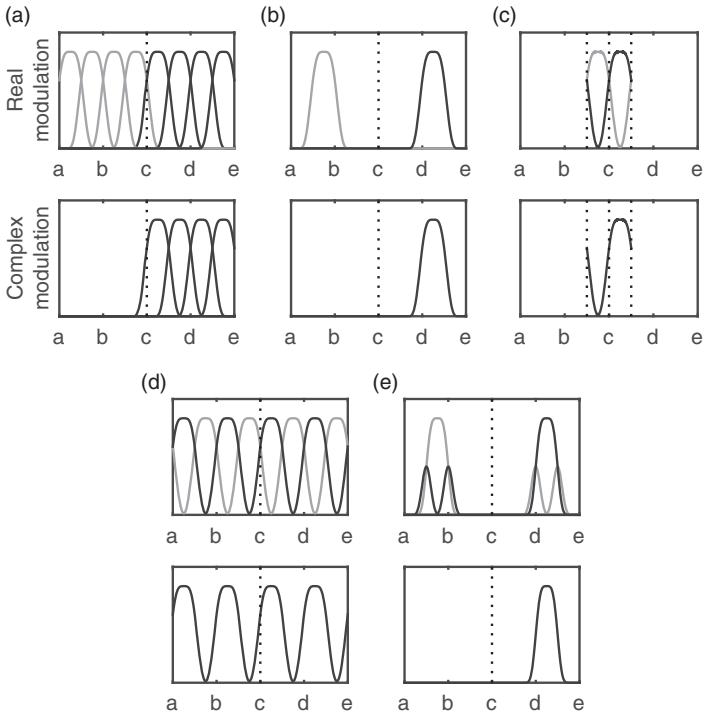

Figure 1.4 Effect of real and complex modulation in uniform filter banks. (a) A set of band-pass filters, (b) individual band-pass filters, (c) their downsampled spectra, (d) the re-upsampled spectra, and (e) the band-pass synthesis-filtered spectra.

spectrum appears either on the positive or the negative frequencies. For real-valued modulators the pass-band spectra appears on both sides, which is also evident from Equation (1.3). Due to the downsampling, spectral information is folded between the new Nyquist frequencies, as shown in Figure 1.4(c). The half-band offset $\alpha = 0.5$ is necessary for real-valued transforms; the spectral overlap of the positive and negative frequencies is thereby minimized and the aliasing components of the real transform are inverses between the adjacent bands. When the frequency bands are not processed independently they cancel each other at the overall output of the filter bank. The complex-valued transform does not have these aliasing components in the first place, which enables the robust independent processing of the bands. The reader is referred to Bosi and Goldberg (2003), Malvar (1992), and Jayant and Noll (1984) for further information on typical time–frequency transforms in audio coding.

1.2.4 A Robust Complex-Modulated Filter Bank, and Comparison with STFT

Our goal, in this section, is to derive the equations of a robust complex-modulated filter bank in a form that can be efficiently implemented using existing FFT algorithms. By default, we use $\alpha = 0$, whereby we obtain a frequency representation with $K + 1$ bands from frequency $\omega_0 = 0$ to $\omega_0 = \pi$, that is, from DC to the Nyquist frequency. The center frequencies of each band are the same as those of the first $(K + 1)$ coefficients of a

$2K$-point FFT. The first and the last bands of this representation are real-valued, and thus the representation has an oversampling factor equal to 2.

As a first step, let us express the equations that correspond to convolving an input signal sequence $x(t)$ with the band-pass filters $h_k(t)$ and performing the downsampling with the factor K. The length of the prototype filter $h_0(t)$ is assumed to be $N_h = 2KC$, where C is a positive integer. An element of the frequency band k is denoted by $X(k, n)$, where n is the downsampled time index. Since applying the downsampling to the band-pass finite impulse response filter output is the same as processing the filter output only every Kth sample, the operations can be expressed as

$$\begin{aligned} X(k,n) &= \sum_{t=0}^{N_h-1} x(t+nK)h_k(t) \\ &= \sum_{t=0}^{N_h-1} \underbrace{x(t+nK)h_0(t)}_{\text{Processing with prototype filter}} \underbrace{e^{-i\pi kt/K+\psi'_k}}_{\text{Modulation}}, \end{aligned} \tag{1.4}$$

where $\psi'_k = \psi_k + \pi k N_h/K$. The notation in Equation (1.4) involves the assumption that the input signal up to $t = (N_h + nK)$ has been accumulated. The first key feature of the equation is that the prototype filter $h_0(t)$ does not depend on the band index k. In other words, for a practical filter bank implementation, we do not need to explicitly derive each of the band-pass filters $h_k(t)$, but only apply $h_0(t)$ to the signal frame. The modulation part in Equation (1.4) accounts for the spectral positions of the band-pass signals.

Let us, then, compare Equation (1.4) with the definition of a $2K$-point STFT which also defines the window length, that is, the case where the prototype filter is of length $N_h = 2K$:

$$X_{\text{STFT}}(k,n) = \sum_{t=0}^{2K-1} x(t+nK)w(t)e^{-i\pi kt/K}, \tag{1.5}$$

where $w(t)$ is the analysis window, which, in a typical configuration, is a square-root Hann window (for a discussion about windowing, see Section 1.2.5). By comparing Equations (1.4) and (1.5), we can readily observe that the STFT is a complex-modulated filter bank with $h_0(t) = w(t)$ and $\psi'_k = 0$. The short prototype filter length of the STFT is a limiting factor for obtaining reasonable stop-band attenuation, as is illustrated in Figure 1.5. In other words, the aliasing effects beyond the adjacent bands are potentially audible at the processed output. Without processing the spectral coefficients, the aliasing effects cancel each other. Thus, the STFT can provide low aliasing when the frequency bands are processed conservatively, such as when applying a spectrally smooth equalizer.

Another important insight from the above derivation is that for the case $\alpha = 0$ and $\psi'_k = 0$, the modulator in Equation (1.4) is cyclic with a period of $2K$ samples. Using the definition $N_h = 2KC$, the equation can be rewritten as

$$X(k,n) = \sum_{t=0}^{2K-1} \left[\sum_{c=0}^{C-1} x(t+(n+2c)K)h_0(t+2cK) \right] e^{-i\pi kt/K}, \tag{1.6}$$

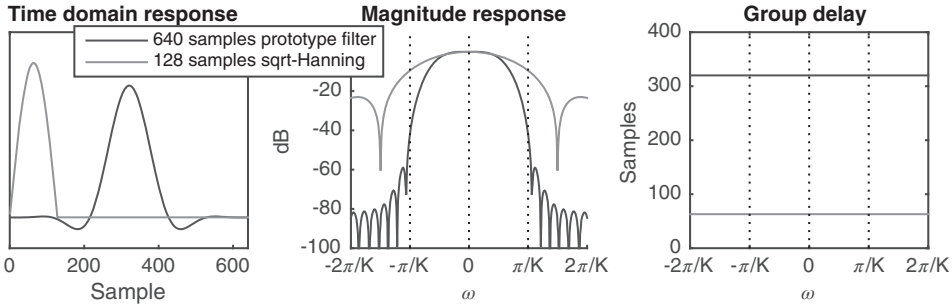

Figure 1.5 Linear-phase prototype filter applied by the MPEG QMF bank, and the square-root Hann window typical in STFT, using $K = 64$.

which is equivalent to performing a $2K$-point FFT on the signal sequence within square brackets. Therefore, even if the applied prototype filter is longer than the transform size, i.e., $C > 1$, we can readily use the existing $2K$-point FFT algorithms to perform the transform itself. The processing block diagram for this approach, equivalent to Equation (1.6), is shown in Figure 1.6. As a result, we obtain an efficient forward transform equivalent to that of the generalized filter bank structure in Figure 1.2.

Similar steps can be taken to also obtain an efficient and robust inverse transform. Let us consider a processed frequency band signal $Y(k, n)$ with $(K + 1)$ bands from the DC frequency to the Nyquist frequency. For formulating the transform, the frequency bands are considered in a $2K$-point conjugate-symmetric form usual for DFTs for real-valued signals,

$$Y(2K - k, n) = Y^*(k, n) \qquad \text{for } k = 1, \ldots, (K - 1). \tag{1.7}$$

We now express the equations showing the combined effect after the inverse transform in Figure 1.2 of all frequency bands of $Y(k, n)$ to the output signal $y(t)$. Let us first consider the effect of a single time index n, and denote its corresponding output $\hat{y}_n(t)$. The

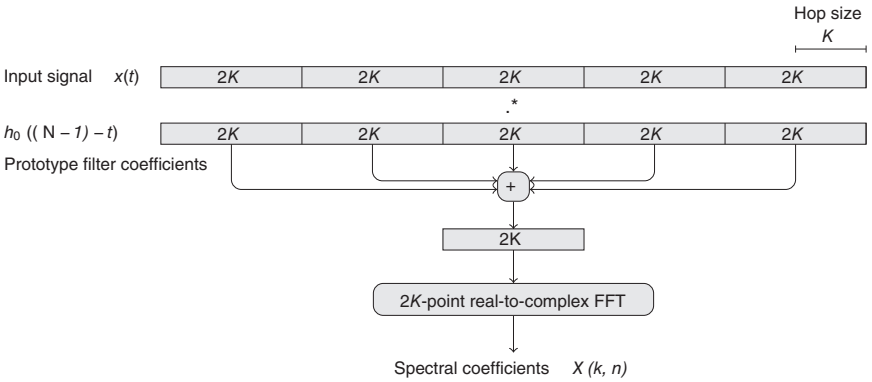

Figure 1.6 Block diagram of a forward transform of a robust filter bank employing the FFT.

upsampling operation determines only the temporal position of the reproduced output. After the band-pass filters $h_k(t)$, and combining the bands, the output is given by

$$\begin{aligned}\hat{y}_n(t+nK) &= \sum_{k=0}^{2K-1} X(k,n)h_k(t) = \sum_{k=0}^{2K-1} X(k,n)h_0(t)e^{i\pi kt/K+\psi_k} \\ &= h_0(t)\underbrace{\sum_{k=0}^{2K-1} X(k,n)e^{i\pi kt/K+\psi_k}}_{2K\text{-point IDFT when } \psi_k=0},\end{aligned} \tag{1.8}$$

where it is assumed that $\hat{y}_n(t)$ is zero outside of the range $t = nK, \ldots, (nK + N_h - 1)$. By setting $\psi_k = 0$, we can readily see the equivalence of the processing to using the $2K$-point inverse discrete Fourier transform (IDFT). Since the output of an IDFT is cyclic, Equation (1.8) is equal to the processing in the block diagram in Figure 1.7, which includes a single $2K$-point IFFT operation, repetition of the result as a C-times longer sequence, and performing a point-wise multiplication with the coefficients of the prototype filter $h_0(t)$. If the prototype filter is a $2K$-length square-root Hann window sequence, the expression becomes equivalent to the traditional inverse STFT. For any prototype filter length, the overall output is obtained with overlap-add processing:

$$y(t) = \sum_{n=0}^{\infty} \hat{y}_n(t). \tag{1.9}$$

Similarly to the analysis filter structure, the definition of STFT with $C = 1$ limits the obtained stop-band attenuation accounting for the spectral aliasing components due to the upsampling. A Matlab script realizing the filter bank structure as described in this section is given in Section 1.2.6.

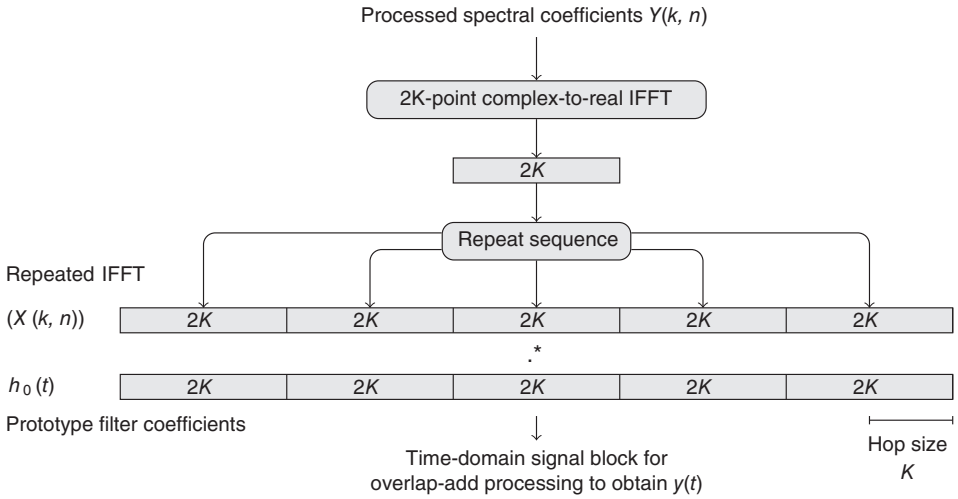

Figure 1.7 Block diagram of an inverse transform of a robust filter bank employing the FFT.

1.2.5 Overlap-Add and Windowing

An important feature of time–frequency transforms in processing applications is that the forward and backward operations themselves, without other processing, do not distort the signal. Such transforms are said to provide *perfect reconstruction*.

In Section 1.2.4 we demonstrated that most filter banks can be implemented using windowing by a prototype filter $h_0(t)$ and application of the discrete Fourier transform. It is well known that the discrete Fourier transform is an orthonormal transform, whereby that part of the transform provides perfect reconstruction. After all, an orthonormal transform is full-rank and, as an added advantage, it can be implemented in a numerically stable manner. Thus, the essential remaining question is whether windowing by $h_0(t)$ and its reverse operation in the inverse transform together provide perfect reconstruction.

Specifically, the question is whether the signal is retained after the processing in Figure 1.2. Observe that the input signal is windowed twice: first with the input window h_0^{a}, and again after processing with the output window h_0^{s}. The output of a given frame n is thus

$$x(t)h_0^{\mathrm{a}}(t - nK)h_0^{\mathrm{s}}(t - nK). \tag{1.10}$$

Similarly, the output of the following frame is

$$x(t)h_0^{\mathrm{a}}(t - (n+1)K)h_0^{\mathrm{s}}(t - (n+1)K). \tag{1.11}$$

Given that the length of the non-zero part of the frame is $2K$ (assuming $C = 1$), it follows that the overlap between windows is of length K. At the overlap region, $t \in [nK, (n+1)K - 1]$, the sum of the two outputs should be equal to the original signal,

$$x(t) = x(t)h_0^{\mathrm{a}}(t - nK)h_0^{\mathrm{s}}(t - nK) + x(t)h_0^{\mathrm{a}}(t - (n+1)K)h_0^{\mathrm{s}}(t - (n+1)K), \tag{1.12}$$

which means that

$$h_0^{\mathrm{a}}(t)h_0^{\mathrm{s}}(t) + h_0^{\mathrm{a}}(t - K)h_0^{\mathrm{s}}(t - K) = 1. \tag{1.13}$$

Often we choose the input and output windows to be equal, $h_0^{\mathrm{a}}(t) = h_0^{\mathrm{s}}(t)$, whereby we can drop the superscripts and write

$$h_0^2(t) + h_0^2(t + K) = 1 \quad \text{for} \quad t \in [0,\, K - 1]. \tag{1.14}$$

This relation, known as the Princen–Bradley condition, ensures that we obtain a perfect reconstruction system (Malvar, 1992). It means that the sum of the square of the windowing function $h_0^2(t)$ and its time-shifted version $h_0^2(t + K)$, where the time shift is half the frame length, must be equal to unity. This relation is illustrated in Figure 1.8, where we see in Figure 1.8(b) that adding the squared windowing functions of subsequent frames adds up to unity.

A typical windowing function that satisfies the Princen–Bradley condition is the square root of the Hann or raised-cosine window, which is equivalent to the half-sine window and is defined as

$$h_0(t) = 0.5\left[1 - \cos\left(\frac{\pi(t + 1/2)}{2K}\right)\right]^{1/2} = \sin\left(\frac{\pi(t + 1/2)}{2K}\right). \tag{1.15}$$

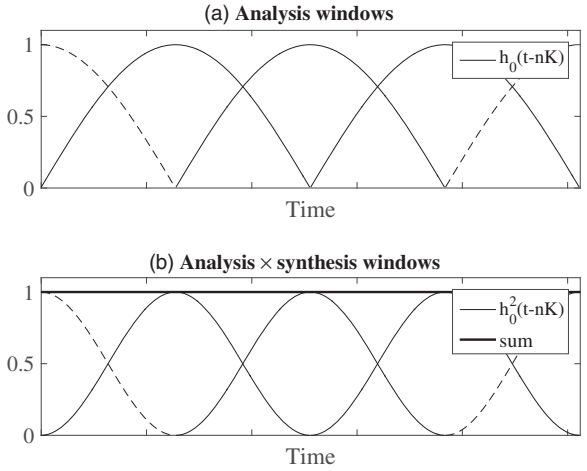

Figure 1.8 (a) Analysis windowing of subsequent windows; (b) overlap-add at the synthesis stage.

1.2.6 Example Implementation of a Robust Filter Bank in Matlab

An example Matlab implementation[1] of a robust filter bank for time–frequency processing is given in Listings 1.1–1.4. Listing 1.1 consists of an initialization function, including the formulation of the prototype filter coefficients. The cutoff frequencies of the equiripple prototype filters were preformulated for $K = 32, 64, 128, 256$ using the iterative method outlined in Creusere and Mitra (1995). Listings 1.2 and 1.3 respectively provide the forward and the inverse transforms according to the block diagrams in Figures 1.6 and 1.7. Listing 1.4 shows an example script that applies the filter bank to band-pass filter a noise sequence in the transform domain. The resulting spectrum of the script is illustrated in Figure 1.9.

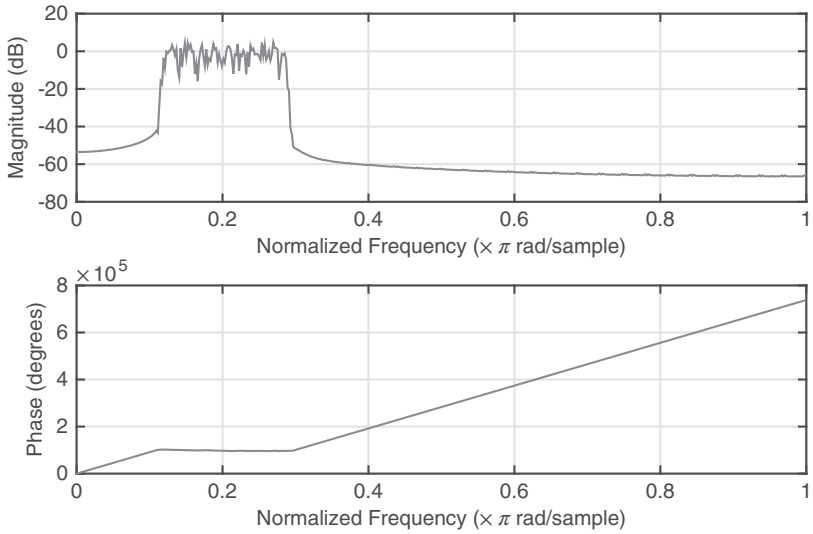

Figure 1.9 Output frequency plot of the Matlab script in Listing 1.1.

[1] An open-source C library implementing a filter bank structure of this type is available at https://github.com/jvilkamo/afSTFT/.

Listing 1.1: Initialization for a Filter Bank

```
function TF_struct = TF_transform_init(K,numInChans,numOutChans)
% Prototype filter frequencies have been pre-formulated ...
    according to Creusere and Mitra.
switch K
    case 32,  f=0.018716082;
    case 64,  f=0.0093552526;
    case 128, f=0.0046768663;
    case 256, f=0.0023382376;
    otherwise, warning('Use K=32, 64, 128, or 256)'); return;
end
TF_struct.C = 5;
TF_struct.N = 2*K*TF_struct.C;
TF_struct.prototypeFilter = firceqrip(TF_struct.N-1, f, ...
    [1e-6 1e-5])';
TF_struct.K = K;
TF_struct.numInChans = numInChans;
TF_struct.numOutChans = numOutChans;
TF_struct.inBuffer=zeros(TF_struct.N, numInChans);
TF_struct.outBuffer=zeros(TF_struct.N, numOutChans);
```

Listing 1.2: Forward Transform

```
function [TFdata,TF_struct] = TF_transform_forward(timeData, ...
    TF_struct)
if size(timeData,2)  ≠ TF_struct.numInChans
    warning('Wrong number of input channels');return;
end
K=TF_struct.K;
numTFsamples = floor(size(timeData,1)/K);
numBands=K+1;
TFdata=zeros(numBands,numTFsamples,TF_struct.numInChans);
for TFsample = 1:numTFsamples
    timeDataFrame = timeData((TFsample-1)*K+[1:K],:);
    TF_struct.inBuffer = [TF_struct.inBuffer(K+1:end,:); ...
        timeDataFrame];
    prototypeFiltered = TF_struct.inBuffer.*repmat(flipud ...
        (TF_struct.prototypeFilter),[1 TF_struct.numInChans]);
    folded = prototypeFiltered(1:2*K,:);
    for c=2:TF_struct.C
        folded = folded + prototypeFiltered([1:2*K]+2*K*(c-1),:);
    end
    transformed = fft(folded);
    for ch=1:TF_struct.numOutChans
        TFdata(:,TFsample,ch)=transformed(1:K+1,ch);
    end
end
```

Listing 1.3: Inverse Transform

```
function [timeData,TF_struct] = TF_transform_inverse(TFdata, ...
    TF_struct)
if size(TFdata,3)  ≠ TF_struct.numOutChans
    warning('Wrong number of output channels');return;
end
K=TF_struct.K;
numTFsamples=size(TFdata,2);
numTimeSamples = floor(numTFsamples*K);
timeData=zeros(numTimeSamples,TF_struct.numOutChans);
FFTdata = zeros(2*K,TF_struct.numOutChans);
for TFsample = 1:numTFsamples
    FFTdata(1:K+1) = squeeze(TFdata(:,TFsample,:));
    inverseTransformed = ...
        repmat(ifft(FFTdata,'symmetric'),[TF_struct.C, 1])*2*K^2;
    prototypeFiltered = inverseTransformed.*repmat ...
        (TF_struct.prototypeFilter,[1 TF_struct.numOutChans]);
    TF_struct.outBuffer = [TF_struct.outBuffer(K+1:end,:); ...
        zeros(K,TF_struct.numOutChans)];
    TF_struct.outBuffer = TF_struct.outBuffer + prototypeFiltered;
    timeData([1:K]+(TFsample-1)*K,:) = TF_struct.outBuffer(1:K,:);
end
```

Listing 1.4: Example Script Using the Filter Bank

```
function TF_transform_example
K=64;
numInChans=1;
numOutChans=1;
TF_struct = TF_transform_init(K,numInChans,numOutChans);
inAudio = randn(44100,1)/sqrt(44100);
[inAudioTF, TF_struct] = TF_transform_forward(inAudio,TF_struct);
inAudioTF(1:8,:,:)=0;
inAudioTF(20:end,:,:)=0;
[outAudio, TF_struct] = TF_transform_inverse(inAudioTF,TF_struct);
freqz(outAudio)
```

1.2.7 Cascaded Filters

When a prototype filter with a high stop-band attenuation is applied in a complex-modulated filter bank, the aliasing at the downsampled frequency bands becomes negligible. Therefore, the frequency band signals can be processed and filtered without restriction. In particular, it is typical to apply cascaded filters at the lowest uniform frequency bands to obtain further spectral selectivity to better approximate the perceptual scales. For example, in a typical configuration of the MPEG QMF bank, a 64-band uniform transform operation is applied, of which the first three frequency bands are

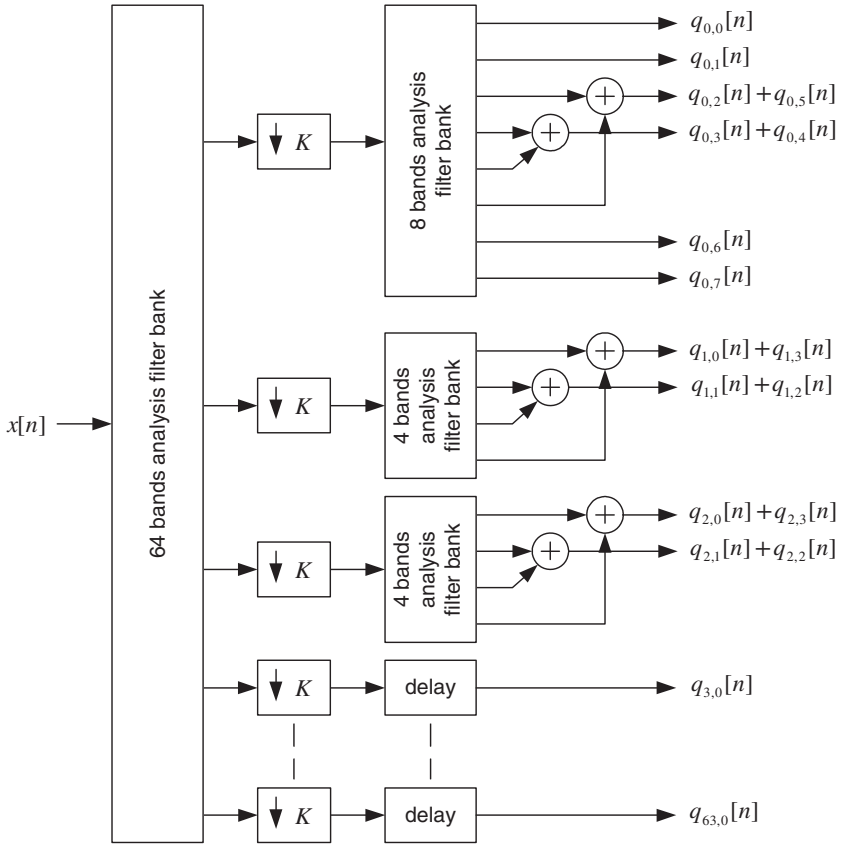

Figure 1.10 Cascaded filters applied in MPEG systems to obtain a higher frequency selectivity at the low frequencies. Adapted from Breebaart *et al.* (2005).

subdivided with linear-phase filters to form in total a 71-band representation (Figure 1.10). The configuration is commonly called the hybrid QMF bank. The filters are designed to be complementary in terms of the amplitude, and not the energy, and no further downsampling is applied. A corresponding delay is applied at the remaining bands. With such a configuration, the synthesis filter bank consists of a simple summation of the hybrid frequency bands, followed by the inverse QMF bank.

1.3 Processing of Spatial Audio

The block diagram of a parametric time–frequency processing algorithm shown in Figure 1.1 also represents most of the parametric spatial audio processing methods discussed in this book. The methods all utilize some kind of stochastic analysis of time–frequency domain differences between microphone signals, which will be discussed in the next section. In the reproduction of spatial audio, the computed time–frequency-dependent parameters are then typically used to control the mixing process,

where in some cases decorrelation of signals is used to control the coherence between output channels, as discussed in Section 1.3.2. An optimal method to control the level of decorrelated signal energy is utilized in some of the spatial audio methods discussed in the book; this is described in Section 1.3.3.

1.3.1 Stochastic Estimates

A property in many applications in parametric time–frequency audio processing is that the multiple audio signals are measured by their short-time stochastic properties in frequency bands. Let us define an N_x-channel time–frequency signal vector

$$\mathbf{x}(k,n) = \begin{bmatrix} X_1(k,n) \\ X_2(k,n) \\ \vdots \\ X_{N_\mathrm{x}}(k,n) \end{bmatrix}. \tag{1.16}$$

The general expression for the signal covariance matrix is

$$\mathbf{C}_\mathbf{x}(k,n) = \mathrm{E}[\mathbf{x}(n,k)\mathbf{x}^\mathrm{H}(n,k)], \tag{1.17}$$

where $\mathrm{E}[\cdot]$ is the expectation operator and $\mathbf{x}^\mathrm{H}(n,k)$ is the conjugate transpose of $\mathbf{x}(n,k)$. In adaptive techniques the covariance matrix is estimated using a windowing function over segments of the frequency band signal,

$$\mathbf{C}_\mathbf{x}(k,n) = \sum_{n_w=n-N_w/2}^{n+N_w/2-1} w(n_w + N_w/2)\mathbf{x}(n_w,k)\mathbf{x}^\mathrm{H}(n_w,k), \tag{1.18}$$

where N_w is the length of the window, typically corresponding to tens of milliseconds. Another typical approach is to use first-order infinite impulse response (IIR) estimation,

$$\mathbf{C}_\mathbf{x}(k,n) = \beta\mathbf{C}_\mathbf{x}(k,n-1) + (1-\beta)\mathbf{x}(n,k)\mathbf{x}(n,k)^\mathrm{H}, \tag{1.19}$$

where β is a parameter determining the decay rate of the IIR window. Estimation of the covariance matrix can be performed along an interval in the frequency axis as well, which can be useful at the higher frequency bands in which the uniform frequency resolution is more selective than the perceptual scales. Such a procedure is applied, for example, for reducing the data rate of the parametric spatial information in audio coding, or for reducing the computational complexity of any parameter-driven processing.

Let us study the entries of the covariance matrix,

$$\mathbf{C}_\mathbf{x}(k,n) = \mathrm{E}[\mathbf{x}(n,k)\mathbf{x}(n,k)^\mathrm{H}] = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N_\mathrm{x}} \\ c_{21} & c_{22} & & c_{2N_\mathrm{x}} \\ \vdots & & \ddots & \vdots \\ c_{N_\mathrm{x}1} & \cdots & & c_{N_\mathrm{x}N_\mathrm{x}} \end{bmatrix}, \tag{1.20}$$

where

$$c_{ab} = \mathrm{E}[X_a(k,n)X_b^*(k,n)]. \tag{1.21}$$

The diagonal values $a = b$ express the estimated channel energies, and the non-diagonal values $a \neq b$ express the cross-correlation of the channels. From these measures, it is possible to obtain the typical measures of the inter-channel level difference,

$$\mathrm{ICLD}_{ab} = 10 \log_{10} \left(\frac{c_{aa}}{c_{bb}} \right), \tag{1.22}$$

the inter-channel phase difference,

$$\mathrm{IPD}_{ab} = \arg(c_{ab}), \tag{1.23}$$

and the inter-channel coherence,

$$\mathrm{ICC}_{ab} = \frac{|c_{ab}|}{\sqrt{c_{aa} c_{bb}}}. \tag{1.24}$$

The channel energies, IPDs, and ICCs provide a set of information equivalent to the covariance matrix, and thus it is only a matter of practicality which representation to apply as part of a parametric processing technique. In techniques that do not account for the phase differences between the channels, the ICC is sometimes considered only by its real part, using an alternative definition:

$$\mathrm{ICC}'_{ab} = \frac{\mathrm{Re}\{c_{ab}\}}{\sqrt{c_{aa} c_{bb}}}. \tag{1.25}$$

The perceptual parametric processing techniques typically analyze and/or synthesize the spatial sound based on the information contained in these stochastic estimates, using either expressions similar to Equations (1.22)–(1.25) or the signal covariance matrix. Since the estimators in Equations (1.18) and (1.19) are essentially low-pass filters, it is meaningful in a computational sense to consider the parameters only sparsely in time. An often-met practical approach is to use a rectangular window estimator with window length N_w corresponding to a time frame in the range of tens of milliseconds, and to apply the estimation every $N_w/2$ time indices. At each of these frames, the parametric data is processed to obtain a rule to mix, process, and/or to gate the frequency band signals. These operations are typically expressible as a matrix of complex-valued mixing weights to process the output frequency band signals based on the input frequency band signals. The weights are formulated for every frame, that is, sparsely in time, and they can be linearly interpolated between the frames prior to applying them to the frequency band signals.

1.3.2 Decorrelation

In many use cases of parametric processing, the input signals consist of a smaller number of prominent independent signals than is required for the output signals. One example is in upmixing, where we want to increase the number of channels—for example, from a two-channel stereophonic signal to a five-channel surround signal. Another example is spatial sound reproduction from a microphone array, where the input signals are highly coherent at low frequencies due to the wavelength with respect to the array size. Signal mixing alone does not increase the number of independent signals, or in the case of array processing, mixing the microphone signals to reduce coherence would result

in excessive amplification of the microphone self-noise. Techniques known as *decorrelating* methods are therefore frequently used to synthesize new incoherent signals from the existing signals.

Ideally, decorrelators produce signals which are incoherent with respect to their inputs and with respect to the outputs of other decorrelators. In addition, the methods should minimize any perceptual modifications of the signal. Various designs of decorrelators have been developed. A simple yet effective method is to apply different delays at different bands and channels (Boueri and Kyriakakis, 2004). Another common class of decorrelators is based on all-pass filters, as applied, for example, in Breebaart *et al.* (2005).

Regardless of the type of decorrelator, its design and tuning are practical engineering tasks. None of the decorrelator structures can preserve the sound quality in a general sense, since there are signal types such as transients and speech for which the phase spectrum has perceptual significance (Laitinen *et al.*, 2013). Typical methods for mitigating the distortions caused by decorrelation include switching the type of decorrelator applied depending on signal type (Kuntz *et al.*, 2011), and bypassing transients from the decorrelating processes.

In the following sections, we denote the decorrelating operation by

$$\mathbf{x}_{\mathrm{D}}(k,n) = \mathrm{D}[\mathbf{x}(k,n)]. \tag{1.26}$$

The objective is that, for the decorrelated signal, $\mathrm{E}[\mathbf{x}(n,k)\mathbf{x}_{\mathrm{D}}^{\mathrm{H}}(n,k)] \equiv \mathbf{0}$.

1.3.3 Optimal and Generalized Solution for Spatial Sound Processing Using Covariance Matrices

Although many parametric time–frequency processing techniques can be designed with dedicated signal processing structures, for many tasks it is also an option to use a generalized solution as proposed in Vilkamo, Bäckström, and Kuntz (2013). If the task of the parametric processing is defined using expressions that can be applied for a wide range of techniques, it is possible to derive a single solution and implementation that is applicable to each of them. Most importantly, we can employ least-squares techniques to optimize the audio fidelity for all use cases in scope. For example, the methods described in the following have been applied in spatial sound reproduction from microphone arrays (Chapters 6 and 8 of this book; Vilkamo and Pulkki, 2013; Politis *et al.*, 2015), binaural sound reproduction (Chapters 6 and 8 of this book; Delikaris-Manias *et al.*, 2015), spatial enhancement (Vilkamo and Pulkki, 2015), and stereo-to-surround upmixing (Vilkamo *et al.*, 2013).

Perceptual spatial sound processing is defined here, in a generic and generalized form, as the design of the output signal covariance matrix in frequency bands. The covariance matrix contains the perceptually crucial measures of the channel energies, the inter-channel coherences, and the phase differences. It does not, however, contain information about the actual fine spectral information in the channels, which is addressed separately.

For brevity of notation, let us omit the frequency and the time indices (k,n) and consider a single time–frequency segment in which an application has measured the covariance matrix $\mathbf{C}_{\mathbf{x}} = \mathrm{E}[\mathbf{x}\mathbf{x}^{\mathrm{H}}]$ of the $N_{\mathrm{x}} \times 1$ input signal $\mathbf{x}$. The $N_{\mathrm{y}} \times 1$ output

time–frequency signal is defined as $\mathbf{y}$. In the generalized form, the parametric processing task is expressed by two matrices:

- a target covariance matrix $\mathbf{C}_\mathbf{y} = \mathrm{E}[\mathbf{y}\mathbf{y}^\mathrm{H}]$, which contains the perceptually relevant *spatial information* for the target channels, such as loudspeaker or headphone channels; and
- a prototype matrix $\mathbf{Q}$, which determines the relevant *signal information* for the target channels.

The prototype matrix $\mathbf{Q}$ can be interpreted as an example of the mapping between $\mathbf{x}$ and $\mathbf{y}$, that is, it determines a prototype signal

$$\mathbf{y}_\mathbf{Q} = \mathbf{Q}\mathbf{x}, \tag{1.27}$$

which is a linear combination or a selection of the input channels. The prototype matrix is designed for each application such that it produces for each output channel the signal content that is most meaningful in a spatial sense. An example of a prototype matrix in the context of microphone array processing is a set of beamformers that are spatially selective towards the directions corresponding to those of the target loudspeaker setup. In the application of spatial enhancement and effects, on the other hand, the prototype matrix is defined to be an identity matrix, since it is desirable that the signal content in the output channels is least affected by the parametric processing.

The method to determine the target covariance matrix $\mathbf{C}_\mathbf{y}$ also depends on the task. For example, in spatial sound reproduction based on sound field models, $\mathbf{C}_\mathbf{y}$ is determined such that it represents the modeled parameters of the sound field for the target loudspeaker array. This includes placing sound energy in the directions corresponding to the analyzed arriving sound in the original sound field, and spatially spreading the portion of the sound energy that was analyzed as ambience. The adaptive design of the target covariance matrix is exemplified further in the context of the specific use cases in Chapters 6 and 8.

The task for the parametric processing is thus to obtain an output signal $\mathbf{y}$ with the application-determined target covariance matrix $\mathbf{C}_\mathbf{y} = \mathrm{E}[\mathbf{y}\mathbf{y}^\mathrm{H}]$, which is usually different from that of $\mathbf{y}_\mathbf{Q}$. The input–output relation for the processing is defined as

$$\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{M}_\mathrm{D}\mathrm{D}[\mathbf{Q}\mathbf{x}], \tag{1.28}$$

where $\mathbf{M}$ is the primary mixing matrix that is solved to produce $\mathbf{C}_\mathbf{y}$. However, solving $\mathbf{M}$ is subject to regularization to ensure robust sound quality. $\mathrm{D}[\cdot]$ is a set of decorrelating signal processing operations, and $\mathbf{M}_\mathrm{D}$ is a secondary mixing matrix that is designed to process the decorrelated signals to produce a covariance matrix that is complementary to the effect of the regularization of $\mathbf{M}$. Assuming incoherence of the decorrelated signals $\mathrm{D}[\mathbf{Q}\mathbf{x}]$ and the input signals $\mathbf{x}$, the expression of the complementary property of the covariance matrices is

$$\mathbf{C}_\mathbf{y} = \mathbf{M}\mathbf{C}_\mathbf{x}\mathbf{M}^\mathrm{H} + \mathbf{C}_\mathrm{D}, \tag{1.29}$$

where $\mathbf{C}_\mathrm{D}$ is the covariance matrix for the processed signal $\mathbf{M}_\mathrm{D}\mathrm{D}[\mathbf{Q}\mathbf{x}]$.

Let us temporarily assume that $\mathbf{M}_\mathrm{D} = \mathbf{0}$, that is, that the decorrelated signals are not applied. Equation (1.29) simplifies to

$$\mathbf{C}_\mathbf{y} = \mathbf{M}\mathbf{C}_\mathbf{x}\mathbf{M}^\mathrm{H}. \tag{1.30}$$

The set of solutions for $\mathbf{M}$ fulfilling Equation (1.30) can be found using matrix decompositions. First, let us determine any unitary matrices $\mathbf{P_x}$ and $\mathbf{P_y}$, which satisfy the properties $\mathbf{P_x}\mathbf{P_x}^{\mathrm{H}} = \mathbf{I}$ and $\mathbf{P_y}\mathbf{P_y}^{\mathrm{H}} = \mathbf{I}$, and define a set of decompositions of the covariance matrices:

$$\begin{aligned}\mathbf{C_x} &= \mathbf{K_x}\mathbf{K_x}^{\mathrm{H}} = \mathbf{K_x}\mathbf{P_x}\mathbf{P_x}^{\mathrm{H}}\mathbf{K_x}^{\mathrm{H}},\\ \mathbf{C_y} &= \mathbf{K_y}\mathbf{K_y}^{\mathrm{H}} = \mathbf{K_y}\mathbf{P_y}\mathbf{P_y}^{\mathrm{H}}\mathbf{K_y}^{\mathrm{H}}.\end{aligned} \tag{1.31}$$

An example of obtaining $\mathbf{K_x}$ and $\mathbf{K_y}$ based on $\mathbf{C_x}$ and $\mathbf{C_y}$ is the Cholesky decomposition. The unitary matrices $\mathbf{P_x}$ and $\mathbf{P_y}$ widen the scope of the decompositions in Equation (1.31) to cover all decompositions fulfilling the defined property. Substituting Equation (1.31) in (1.30), we obtain

$$\mathbf{K_y}\mathbf{P_y}\mathbf{P_y}^{\mathrm{H}}\mathbf{K_y}^{\mathrm{H}} = \mathbf{M}\mathbf{K_x}\mathbf{P_x}\mathbf{P_x}^{\mathrm{H}}\mathbf{K_x}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}}, \tag{1.32}$$

from which we obtain the set of solutions

$$\boxed{\mathbf{M} = \mathbf{K_y}\mathbf{P_y}\mathbf{P_x}^{\mathrm{H}}\mathbf{K_x}^{-1} = \mathbf{K_y}\mathbf{P}\mathbf{K_x}^{-1},} \tag{1.33}$$

where $\mathbf{P} = \mathbf{P_y}\mathbf{P_x}^{\mathrm{H}}$ is any unitary matrix. To recap the steps so far, for any unitary $\mathbf{P}$, the signal $\mathbf{y} = \mathbf{M}\mathbf{x} = \mathbf{K_y}\mathbf{P}\mathbf{K_x}^{-1}\mathbf{x}$ has the target covariance matrix $\mathbf{C_y}$.

As the second step of the derivation, we find a $\mathbf{P}$ that provides the smallest difference in the waveforms of the reproduced signal $\mathbf{y}$ and the prototype signal $\mathbf{y_Q}$. This difference is expressed by an error measure,

$$e = \mathrm{E}[\|\mathbf{G}\mathbf{y_Q} - \mathbf{y}\|^2], \tag{1.34}$$

where $\mathbf{G}$ is a non-negative diagonal matrix that is formulated such that the channel energies of $\mathbf{y_Q}$ match the diagonal of $\mathbf{C_y}$. This normalization ensures that the error measure is weighted with the energies of the reproduced channels. Defining $\mathbf{A} = (\mathbf{G}\mathbf{y_Q} - \hat{\mathbf{y}})$, we can use the matrix trace operator $\mathrm{tr}(\cdot)$ to obtain the error measure in the form

$$e = \mathrm{E}[\mathbf{A}^{\mathrm{H}}\mathbf{A}] = \mathrm{E}[\mathrm{tr}(\mathbf{A}^{\mathrm{H}}\mathbf{A})] = \mathrm{E}[\mathrm{tr}(\mathbf{A}\mathbf{A}^{\mathrm{H}})], \tag{1.35}$$

where we used the equivalence $\mathrm{tr}(\mathbf{A_1}^{\mathrm{H}}\mathbf{A_2}) = \mathrm{tr}(\mathbf{A_1}\mathbf{A_2}^{\mathrm{H}})$. Substituting $\mathbf{y_Q} = \mathbf{Q}\mathbf{x}$ and $\mathbf{y} = \mathbf{M}\mathbf{x} = \mathbf{K_y}\mathbf{P}\mathbf{K_x}^{-1}\mathbf{x}$ in Equation (1.35), and substituting the definition of $\mathbf{A}$, we obtain the error measure

$$\begin{aligned}e &= \mathrm{E}\big[\mathrm{tr}\big((\mathbf{G}\mathbf{Q}\mathbf{x} - \mathbf{K_y}\mathbf{P}\mathbf{K_x}^{-1}\mathbf{x})(\mathbf{G}\mathbf{Q}\mathbf{x} - \mathbf{K_y}\mathbf{P}\mathbf{K_x}^{-1}\mathbf{x})^{\mathrm{H}}\big)\big]\\ &= \mathrm{tr}(\mathbf{G}\mathbf{Q}\mathbf{C_x}\mathbf{Q}^{\mathrm{H}}\mathbf{G}^{\mathrm{H}}) - 2\mathrm{tr}\big(\mathbf{G}\mathbf{Q}\mathbf{K_x}\mathbf{P}^{\mathrm{H}}\mathbf{K_y}^{\mathrm{H}}\big) + \mathrm{tr}\big(\mathbf{K_y}\mathbf{K_y}^{\mathrm{H}}\big),\end{aligned} \tag{1.36}$$

where we have applied the definition $\mathbf{C_x} = \mathrm{E}[\mathbf{x}\mathbf{x}^{\mathrm{H}}]$, the middle term has been simplified using $\mathbf{C_x}(\mathbf{K_x}^{-1})^{\mathrm{H}} = \mathbf{K_x}$, from Equation (1.31), and the last term has been simplified using the identities $\mathbf{K_x}^{-1}\mathbf{C_x}(\mathbf{K_x}^{-1})^{\mathrm{H}} = \mathbf{I}$ and $\mathbf{P}\mathbf{P}^{\mathrm{H}} = \mathbf{I}$. In Equation (1.36), only the second term depends on the variable $\mathbf{P}$, and we can thus formulate the minimization problem as

$$\begin{aligned}\arg\min_{\mathbf{P}} e &= \arg\max_{P} \mathrm{tr}\big(\mathbf{G}\mathbf{Q}\mathbf{K_x}\mathbf{P}^{\mathrm{H}}\mathbf{K_y}^{\mathrm{H}}\big)\\ &= \arg\max_{P} \mathrm{tr}\big(\mathbf{K_x}^{\mathrm{H}}\mathbf{Q}^{\mathrm{H}}\mathbf{G}^{\mathrm{H}}\mathbf{K_y}\mathbf{P}\big),\end{aligned} \tag{1.37}$$

where we used the relation $\mathrm{tr}(\mathbf{X}^\mathrm{H}\mathbf{Y}) = \mathrm{tr}(\mathbf{Y}\mathbf{X}^\mathrm{H})$ (Petersen and Pedersen, 2012). Finally, assuming first that $N_\mathrm{y} = N_\mathrm{x}$, the optimal solution is found using the fact that, for a non-negative diagonal matrix $\mathbf{S}$ and any unitary matrix $\mathbf{P}_\mathrm{s}$, $\mathrm{tr}(\mathbf{S}) \geq \mathrm{tr}(\mathbf{S}\mathbf{P}_\mathrm{s})$, which is a consequence of the Cauchy–Schwartz inequality. Defining a singular value decomposition (SVD) with the relation $\mathbf{U}\mathbf{S}\mathbf{V}^\mathrm{H} = \mathbf{K}_\mathbf{x}^\mathrm{H}\mathbf{Q}^\mathrm{H}\mathbf{G}^\mathrm{H}\mathbf{K}_\mathbf{y}$, where $\mathbf{S}$ is non-negative and diagonal, and $\mathbf{U}$ and $\mathbf{V}$ are unitary, it follows that

$$\mathrm{tr}\left(\mathbf{K}_\mathbf{x}^\mathrm{H}\mathbf{Q}^\mathrm{H}\mathbf{G}^\mathrm{H}\mathbf{K}_\mathbf{y}\mathbf{P}\right) = \mathrm{tr}(\mathbf{U}\mathbf{S}\mathbf{V}^\mathrm{H}\mathbf{P}) = \mathrm{tr}(\mathbf{U}\mathbf{S}\mathbf{V}^\mathrm{H}\mathbf{P}\mathbf{U}\mathbf{U}^\mathrm{H}) = \mathrm{tr}(\mathbf{S}\mathbf{V}^\mathrm{H}\mathbf{P}\mathbf{U}) \leq \mathrm{tr}(\mathbf{S}) \tag{1.38}$$

for any unitary $\mathbf{P}$. The equality holds for

$$\mathbf{P} = \mathbf{V}\mathbf{U}^\mathrm{H}, \tag{1.39}$$

which is thus a solution that minimizes the error measure e in Equation (1.34). Furthermore, the optimal solution when $N_\mathrm{x} \neq N_\mathrm{y}$ is (Vilkamo *et al.*, 2013)

$$\boxed{\mathbf{P} = \mathbf{V}\mathbf{\Lambda}\mathbf{U}^\mathrm{H},} \tag{1.40}$$

where $\mathbf{\Lambda}$ is an identity matrix appended with zeros to match the dimensions of $\mathbf{P}$.

Regardless of being optimal in a least-squares sense, the aforementioned mixing solution does not yet account for the cases when:

- the matrix $\mathbf{K}_\mathbf{x}$ in Equation (1.33) is not invertible;
- the matrix inverse $\mathbf{K}_\mathbf{x}^{-1}$ has very large coefficients (when it is ill-conditioned) which typically causes processing artifacts; or
- the number of output channels is larger than the number of input channels.

To account for the first two cases we need to regularize the inverse of matrix $\mathbf{K}_\mathbf{x}$. A robust method is to employ the SVD with the relation $\mathbf{K}_\mathbf{x} = \mathbf{U}_\mathbf{x}\mathbf{S}_\mathbf{x}\mathbf{V}_\mathbf{x}^\mathrm{H}$, whereby regularization of $\mathbf{K}_\mathbf{x}$ is equivalent to regularizing $\mathbf{S}_\mathbf{x}$ to form the regularized $\mathbf{S}'_\mathbf{x}$. We then obtain the regularized matrix as $\mathbf{K}'^{-1}_\mathbf{x} = \mathbf{V}_\mathbf{x}\mathbf{S}'^{-1}_\mathbf{x}\mathbf{U}_\mathbf{x}^\mathrm{H}$.

A practical regularization is thresholding from below: Let scalars $s_k > 0$ be the diagonal elements of $\mathbf{S}_\mathbf{x}$ and set the regularization rule as

$$s'_k := \begin{cases} s_k & \text{for } s_k > \alpha \cdot \max\{s_k\}, \\ \alpha \cdot \max\{s_k\} & \text{for } s_k \leq \alpha \cdot \max\{s_k\}, \end{cases} \tag{1.41}$$

where $\alpha = 0.2$ is a typical value. In other words, we increase the singular values of $\mathbf{K}_\mathbf{x}$ until they are at least 0.2 times the largest singular value $\max\{s_k\}$. The regularized inverse $\mathbf{K}'^{-1}_\mathbf{x}$ is then substituted for $\mathbf{K}_\mathbf{x}^{-1}$ in Equation (1.33), such that we obtain a regularized version $\mathbf{M}$.

Using the regularized $\mathbf{M}$, the mixing solution $\mathbf{M}\mathbf{C}_\mathbf{x}\mathbf{M}^\mathrm{H}$ no longer yields the target covariance matrix $\mathbf{C}_\mathbf{y}$. Consequently, the complementary covariance matrix $\mathbf{C}_\mathrm{D}$ in Equation (1.29) becomes non-zero. The secondary mixing matrix $\mathbf{M}_\mathrm{D}$ is then designed to process the decorrelated signals to obtain $\mathbf{C}_\mathrm{D}$. Mixing the resulting signal with the main signal $\mathbf{M}\mathbf{x}$ as in Equation (1.28) produces the covariance matrix $\mathbf{C}_\mathbf{y}$ for the output signal $\mathbf{y}$ by definition. The method to obtain the secondary matrix $\mathbf{M}_\mathrm{D}$ is equivalent to the steps to obtain $\mathbf{M}$; however, the matrix $\mathbf{C}_\mathrm{D}$ is defined as the target covariance matrix, and an identity matrix as the prototype of matrix $\mathbf{Q}$.

As the main result of the above derivation, we have obtained a generalized technique for parametric spatial sound processing using least-squares optimization, which performs the mixing of the signal and decorrelated signals with respect to three parameter matrices: the measured covariance matrix $\mathbf{C_x}$, the target covariance matrix $\mathbf{C_y}$, and a prototype matrix $\mathbf{Q}$. Application of this technique to specific use cases requires only the design of matrices $\mathbf{C_y}$ and $\mathbf{Q}$, which will be demonstrated in Chapters 6 and 8.

References

3GP (2014) *TS 26.445, EVS Codec Detailed Algorithmic Description; 3GPP Technical Specification (Release 12)*, ETSI, Sophia Antipolis.

Allamanche, E., Geiger, R., Herre, J., and Sporer, T. (1999) MPEG-4 low delay audio coding based on the AAC codec. *Audio Engineering Society Convention 106*, Audio Engineering Society.

Bäckström, T. (2013) Vandermonde factorization of Toeplitz matrices and applications in filtering and warping. *IEEE Transactions on Signal Processing*, **61**(24), 6257–6263.

Blauert, J. (1997) *Spatial Hearing: The Psychophysics of Human Sound Localization* (rev. ed.), MIT Press, Cambridge, MA.

Bosi, M., Brandenburg, K., Quackenbush, *et al.* (1997) ISO/IEC MPEG-2 advanced audio coding. *Journal of the Audio Engineering Society*, **45**(10), 789–814.

Bosi, M. and Goldberg, R.E. (2003) *Introduction to Digital Audio Coding and Standards* Kluwer Academic Publishers, Dordrecht.

Boueri, M. and Kyriakakis, C. (2004) Audio signal decorrelation based on a critical band approach *Audio Engineering Society Convention 117*, Audio Engineering Society.

Brandenburg, K. (1999) MP3 and AAC explained. *Audio Engineering Society 17th International Conference*, Audio Engineering Society.

Breebaart, J., Disch, S., Faller, C., *et al.* (2005) The reference model architecture for MPEG spatial audio coding. *Audio Engineering Society Convention 118*, Audio Engineering Society.

Creusere, C.D. and Mitra, S.K. (1995) A simple method for designing high-quality prototype filters for M-band pseudo QMF banks. *IEEE Transactions on Signal Processing*, **43**(4), 1005–1007.

Cruz-Roldán, F., Amo-López, P., Maldonado-Bascón, S., and Lawson, S.S. (2002) An efficient and simple method for designing prototype filters for cosine-modulated pseudo-QMF banks. *IEEE Signal Processing Letters*, **9**(1), 29–31.

Delikaris-Manias, S., Vilkamo, J., and Pulkki, V. (2015) Parametric binaural rendering using compact microphone arrays. *International Conference on Acoustics, Speech, and Signal Processing*, IEEE.

Fastl, H. and Zwicker, E. (2007) *Psychoacoustics: Facts and Models*. Springer, Berlin.

Herre, J., Hilpert, J., Kuntz, A., and Plogsties, J. (2015) MPEG-H Audio: The new standard for universal spatial/3D audio coding. *Journal of the Audio Engineering Society*, **62**(12), 821–830.

Herre, J., Purnhagen, H., Koppens, J., *et al.* (2012) MPEG spatial audio object coding: The ISO/MPEG standard for efficient coding of interactive audio scenes. *Journal of the Audio Engineering Society*, **60**(9), 655–673.

ISO/IEC JTC1/SC29/WG11 (MPEG) (1997) *MPEG-2 Advanced Audio Coding, AAC*, International Standards Organization.

Jayant, N.S. and Noll, P. (1984) *Digital Coding of Waveforms: Principles and Applications to Speech and Video*, Prentice-Hall, Englewood Cliffs, NJ.

Kuntz, A., Disch, S., Bäckström, T., and Robilliard, J. (2011) The transient steering decorrelator tool in the upcoming MPEG unified speech and audio coding standard, *Audio Engineering Society Convention 131*, Audio Engineering Society.

Laitinen, M.V., Disch, S., and Pulkki, V. (2013) Sensitivity of human hearing to changes in phase spectrum. *Journal of the Audio Engineering Society*, **61**(11), 860–877.

Malvar, H.S. (1992) *Signal Processing with Lapped Transforms*. Artech House, Inc., Norwood, MA.

Mitra, S. and Kaiser, J. (eds.) (1993) *Handbook of Digital Signal Processing*. John Wiley & Sons, Ltd, Chichester.

Moore, B.C.J. (ed.) (1995) *Hearing*. Academic Press, New York.

Oppenheim A.V. and Schafer, R.W. (1975) *Digital Signal Processing*, Prentice-Hall, Englewood-Cliffs, NJ.

Petersen, K.B. and Pedersen, M.S. (2012) *The Matrix Cookbook* vol. 7, Technical University of Denmark.

Politis, A., Vilkamo, J., and Pulkki, V. (2015) Sector-based parametric sound field reproduction in the spherical harmonic domain. *IEEE Journal of Selected Topics in Signal Processing*, **9**(5), 852–866.

Schuller, G. and Smith, M.J. (1995) A new algorithm for efficient low delay filter bank design. *Proc. International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 1472–1475, IEEE.

Smith. J.O. (2011) *Spectral Audio Signal Processing*, W3K Publishing.

Vilkamo, J., Bäckström, T., and Kuntz, A. (2013) Optimized covariance domain framework for time–frequency processing of spatial audio. *Journal of the Audio Engineering Society*, **61**(6), 403–411.

Vilkamo, J. and Pulkki, V. (2013) Minimization of decorrelator artifacts in directional audio coding by covariance domain rendering. *Journal of the Audio Engineering Society*, **61**(9), 637–646.

Vilkamo, J. and Pulkki, V. (2015) Adaptive optimization of interchannel coherence with stereo and surround audio content. *Journal of the Audio Engineering Society*, **62**(12), 861–869.