Audio Processing

Rade Kutil, 2013

Contents

1 Linear Processing

In audio processing, there is a **control flow** in addition to the **signal flow**. In a control flow, the parameters of the process operating on the signal flow are modified. The control flow is usually slower than the signal flow, i.e. for every n signal samples the parameters are changed once, where n is in the range 16 to 4096.

To change the parameters of a linear FIR or IIR filter can be costly because each filter tap has to be calculated according to a certain scheme that defines the filter. Therefore, we now introduce a type of filter that allows for easy parameterization. Thus, they are called **parametric filters**.

The first building block of a parametric filter is a **parametric allpass filter**. The first-order version is given by

$$y[t] = (a * x)[t] = cx[t] + x[t-1] - cy[t-1]\,.$$

Its transfer function is

$$A(z) = \frac{c + z^{-1}}{1 + cz^{-1}}\,.$$

The magnitude response is indeed 1, since

$$|A(z)| = \frac{|c + z^{-1}|}{|1 + cz^{-1}|} = \frac{|c + z^{-1}|}{|z^{-1}| \cdot |z + c|} \overset{|z|=1}{=\!=} 1$$

The phase response $\varphi = \arg(A(e^{i\omega}))$ is 0 for $\omega = 0$ because $A(1) = 1$, and $-180°$ for the Nyquist rate $\omega = \pi$, because $A(-1) = -1$. (The sampling rate is set to 1, by the way.) The parameter c controls the slope of the phase response. To set the phase response to $-90°$ at the "cutoff"-frequency $\omega = 2\pi f_c$, we set $H(e^{-i\omega}) = -i$, which leads to

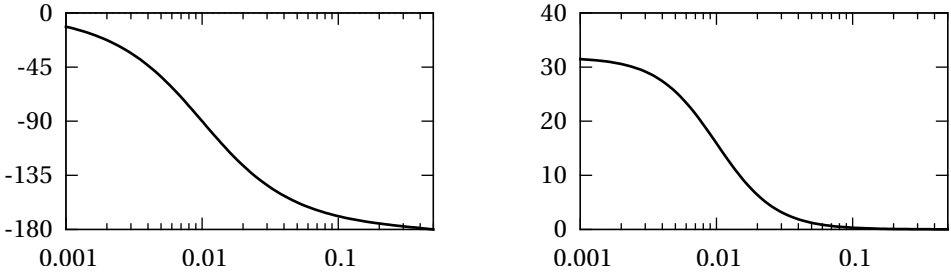
$$c = -\frac{\cos\omega}{1 + \sin\omega} = \frac{\tan(\pi f_c) - 1}{\tan(\pi f_c) + 1}\,.$$

Figure 1 shows the phase response and group delay ($\frac{d\varphi}{d\omega}$) of the parametric allpass filter with $f_c = 0.01$.

Now, a **parametric lowpass** and a **parametric highpass filter** can be implemented by using the allpass filter in the following way. The lowpass filter is

$$y = l * x = \frac{x + a * x}{2}\,, \qquad L(z) = \frac{1 + A(z)}{2}\,.$$

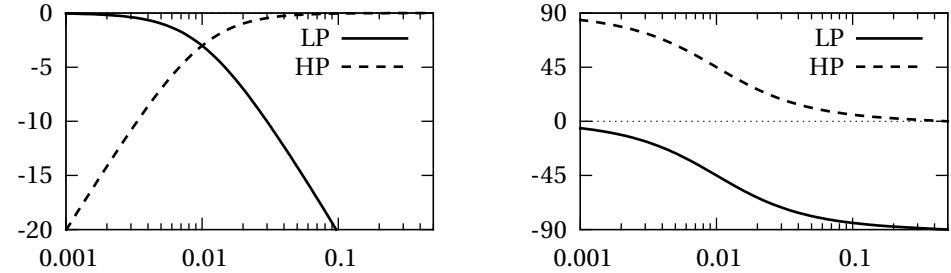
The **parametric highpass filter** simply substitutes a $-$ for the $+$, i.e. $h * x = \frac{x - a * x}{2}$. Figure 2 shows the magnitude and phase response of these filters.



(a) Phase

(b) Group delay in samples

Figure 1: Response of parametric allpass filter with $f_c = 0.01$ depending on frequency $f = \frac{\omega}{2\pi}$.



(a) Magnitude response in dB

(b) Phase

Figure 2: Response of parametric lowpass and highpass filters with $f_c = 0.01$ depending on frequency $f = \frac{\omega}{2\pi}$.

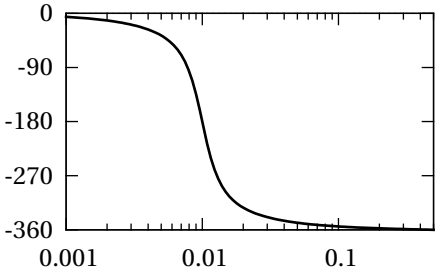


Figure 3: Phase response of second-order allpass filter for $f_c = 0.01$ and $f_d = 0.005$.

The implementation of parametric bandpass and bandreject filters can be achieved with a **second-order allpass filter**. It is given by

$$y[t] = (a_2 * x)[t] = -dx[t] + c(1-d)x[t-1] + x[t-2] - c(1-d)y[t-1] + dy[t-2],$$

and has the transfer function

$$A_2(z) = \frac{-d + c(1-d)z^{-1} + z^{-2}}{1 + c(1-d)z^{-1} - dz^{-2}}.$$

Again, the magnitude response is 1, since

$$|A_2(z)| = \frac{|-d + c(1-d)z^{-1} + z^{-2}|}{|1 + c(1-d)z^{-1} - dz^{-2}|} = \frac{|-d + c(1-d)z^{-1} + z^{-2}|}{|z^{-2}| \cdot |-d + c(1-d)z + z^2|} \overset{|z|=1}{=\!=} 1.$$

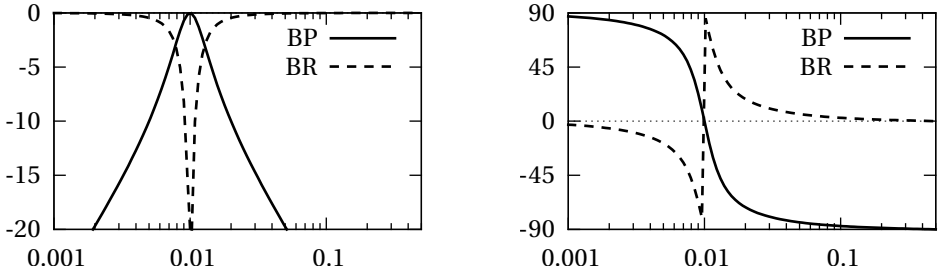
The phase response is 0 for $\omega = 0$ because $|A_2(1)| = 1$, and also 0 (or $-360°$) for the Nyquist rate $\omega = \pi$ because $|A_2(-1)| = 1$. We want the phase to pass through $-180°$at frequency $\omega = \frac{f_c}{2\pi}$, so we set $A(z) = A(e^{i\omega}) = -1$ and get

$$c = -\cos\omega = -\cos 2\pi f_c.$$

Furthermore, the parameter d controls the slope at which the phase changes from $0°$to $-360°$. It may be calculated by

$$d = \frac{\tan(\pi f_d) - 1}{\tan(\pi f_d) + 1}.$$

Figure 3 shows the phase response.



(a) Magnitude response in dB

(b) Phase

Figure 4: Response of parametric second-order bandpass and bandreject filters with $f_c = 0.01$ and $f_d = 0.005$ depending on frequency $f = \frac{\omega}{2\pi}$.

With the help of the second-order allpass filter we can now implement a **second-order bandpass filter**. It is defined by

$$y = b * x = \frac{x - a_2 * x}{2}, \qquad B(z) = \frac{1 - A_2(z)}{2}.$$

Similarly, the **second-order bandreject** filter is defined by

$$y = r * x = \frac{x + a_2 * x}{2}, \qquad R(z) = \frac{1 + A_2(z)}{2}.$$

Figure 4 shows the magnitude and phase responses.

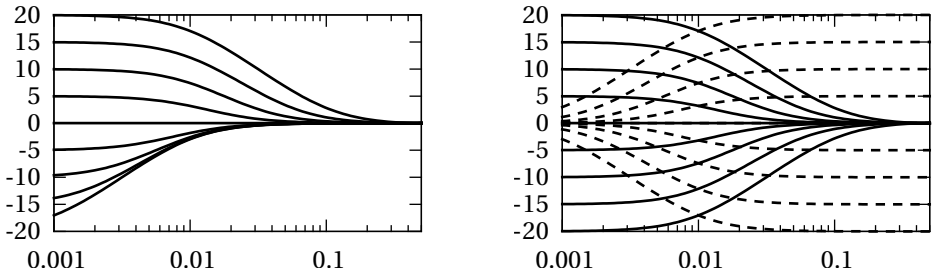
There are corresponding **second-order low-/highpass filters** as well, but the control of their coefficients is somewhat more complicated. With $K = \tan \pi f_c$, the lowpass filter is given by

$$y[t] = (l_2 * x)[t] = \frac{1}{1 + \sqrt{2}K + K^2}(K^2 x[t] + 2K^2 x[t-1] + K^2 x[t-2] \\ - 2(K^2 - 1)y[t-1] - (1 - \sqrt{2}K + K^2)y[t-2]),$$

and the highpass filter by

$$y[t] = (h_2 * x)[t] = \frac{1}{1 + \sqrt{2}K + K^2}(x[t] - 2x[t-1] + x[t-2] \\ - 2(K^2 - 1)y[t-1] - (1 - \sqrt{2}K + K^2)y[t-2]).$$

Often, one does not want to attenuate the stopbands to zero, but instead leave them as they are, and just increase or decrease the amplitude in certain bands. If



(a) uncorrected cut-frequency

(b) corrected cut-frequency

Figure 5: Magnitude response of low-frequency and high-frequency shelving filters for gain from −20dB to +20dB and $f_c = 0.01$, depending on frequency $f = \frac{\omega}{2\pi}$.

those bands include $\omega = 0$ or $\omega = \pi$, then the filters are closely related to lowpass and highpass filters, and are called **shelving filters**. The idea is simply to add the output of a low- or highpass filter to the original signal.

$$s_l * x = x + (v-1)\,l * x, \qquad \text{or} \qquad s_h * x = x + (v-1)\,h * x,$$

where v is the amplitude factor for the passband. s_l is the low-frequency and s_h the high-frequency shelving filter. If a gain in dB is given as V, then $v = 10^{V/20}$. If the cutoff frequency parameter c of the first-order low- or highpass filters is calculated in the usual way, we get an effect that can be seen in Figure 5(a). For gains $v < 1$ (**cut**), the attenuation retreats into the passband, which is not symmetrical to the $v > 1$ case (**boost**), where it extends more into the stopband, the higher the gain gets. To make this symmetrical, the parameter c for the first-order filters has to be calculated differently for $v < 1$, namely

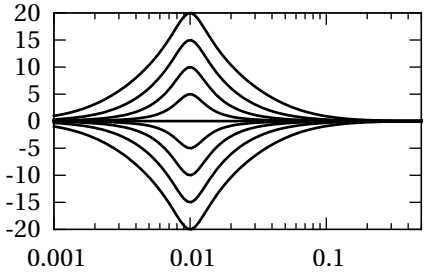
$$c = \frac{\tan(\pi f_c) - v}{\tan(\pi f_c) + v}, \qquad c = \frac{v\tan(\pi f_c) - 1}{v\tan(\pi f_c) + 1},$$

for the low-frequency and the high-frequency filter, respectively. The magnitude responses are shown in Figure 5(b).

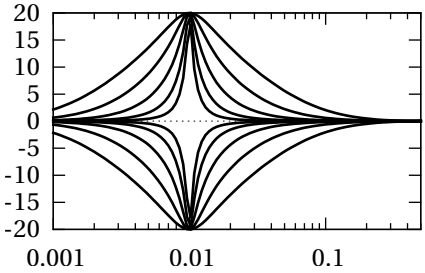
Second-order shelving filters based on second-order low- and highpass filters are also possible, or course. However, their parameters are again more complicated to calculate.

Following the same idea, but using bandpass filters, **peak filters** can be created. Peak filters increase or decrease the amplitude within a certain passband.

$$p * x = x + (v-1)\,b * x.$$



(a) varying gain, $f_d = 0.005$



(b) varying bandwidth $f_d = 0.0005, 0.001, 0.002, 0.004, 0.008$

Figure 6: Magnitude response of peak filters for $f_c = 0.01$.

For a similar reason as for shelving filters (band-narrowing in the cut case), the parameter d or the second-order bandpass filter has to be calculated differently for $v < 1$:

$$d = \frac{\tan(\pi f_d) - v}{\tan(\pi f_d) + v}.$$

Figure 6 shows the magnitude response for the peak filter.

With all these filters, an **equalizer** can be implemented by concatenating shelving and peak filters, one for each equalizer band:

$$e = s_l(f_{cl}, V_l) * p(f_{c1}, f_{d1}, V_1) * \cdots * p(f_{cn}, f_{dn}, V_n) * s_h(f_{ch}, V_h).$$

A **phaser** is a set of second-order bandreject filters with independently varying center frequencies. This can be implemented by a cascade of second-order allpass filters that are mixed with the original filter.

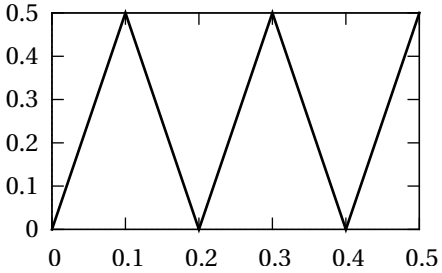
$$ph * x = (1-m)x + m \cdot a_2^{(n)} * \cdots * a_2^{(2)} * a_2^{(1)} * x.$$

$a_2^{(k)}$ are n different second-order allpass filters, controlled by low-frequency oscillators at unrelated frequencies. m is a mix parameter controlling the strength of the effect. Moreover, there is an extension of the scheme with a feedback loop over the allpass filters:

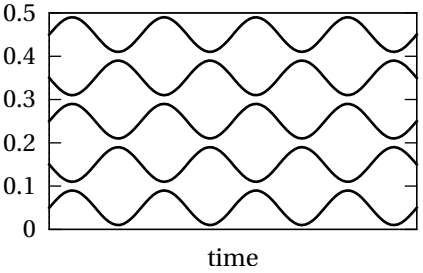
$$ph_3 * x = a_2^{(n)} * \cdots * a_2^{(2)} * a_2^{(1)} * ph_2 * x,$$

$$(ph_2 * x)[t] = x[t] + q \cdot (ph_3 * x)[t-1],$$

$$ph * x = (1-m)x + m \cdot ph_3 * x.$$



(a) Frequency mapping $f \mapsto g(f)$ so that $|P(e^{i2\pi 5f})| = |P(e^{i2\pi g(f)})|$.



(b) Peak frequencies of the Wah-Wah effect with $m = 5$ and peak frequency control by a low-frequency oscillation.

Figure 7: Frequency transforms of a 5-fold Wah-Wah effect.

This further increases the spacy effect of the phaser.

The **Wah-Wah** effect is basically a set of peak filters with varying center frequencies. However, it is implemented with only a single peak filter and an additional trick: The (single tap) delay unit in the filter is substituted by an m-tap delay. This means that the transfer function of the Wah-Wah effect is $W(z) = P(z^m)$. As a result, the amplitude response of the peak filter wraps around the unit circle in the z-plane $\frac{m}{2}$ times while the ω of the Wah-Wah effect moves from 0 to π. Together with the fact that the magnitude response for negative frequencies is the same as for positive ones, i.e.

$$|H(e^{i(-\omega)})| = |H(\bar{z})| = |\overline{H(z)}| = |H(z)| = |H(e^{i\omega})|,$$

and also because $e^{i\omega} = e^{i(\omega \pm 2\pi)}$, we can map the frequencies $m\omega$ to the range $[0, \pi]$ as is shown in Figure 7(a) for the case $m = 5$. Thus, if we control the peak frequency of P by a low-frequency oscillator, we get several incarnations of the peak frequency in a way shown in Figure 7(b). The result is the modification of the audio input signal that sound like "wah-wah", hence the name.

The other parameter, the bandwidth f_d of the peak filter, is often increased linearly with the peak frequency f_c. This means that the so-called **Q-factor**, which is defined as the ratio of bandwidth and cutoff/peak-frequency $q = \frac{f_d}{f_c}$, is held constant (**constant Q-factor**), so that $f_d = q f_c$. The Q-factor is rather high in the case of the Wah-Wah filter, i.e. $q \approx 0.5$.

The idea of extending the delay unit to an m-tap delay leads to the implementation of general **delay** effects. If there is no feedback or mix with the direct

signal, the signal is simply shifted in time, which brings no audible effect. However, if the time-shift m is varied according to a low-frequency oscillator between 0 and 3 ms, the result is a **vibrato** effect. Restricting m to integer values might not be fine grained enough, though. Therefore, **fractional delays** have to be used, which interpolate between $\lfloor m \rfloor$ and $\lceil m \rceil$. The simplest way to do so is **linear interpolation**

$$y[t] = (1-f)x[t-\lfloor m \rfloor] + f\,x[t-\lceil m \rceil]\,,$$

where f is the fractional part $f = m - \lfloor m \rfloor$. The correct way to do it is to use **sinc interpolation**, following Shannon's sampling theorem that states that, if there is no frequency above the Nyquist frequency 0.5, then the continuous signal is determined by

$$x(s) = \sum_{s=-\infty}^{\infty} x[t]\,\mathrm{sinc}(s-t)\,,$$

where s is a continuous time variable, t the discrete time, and $\mathrm{sinc}(s) = \frac{\sin \pi s}{\pi s}$. Because this sinc-kernel has infinite length, cannot be implemented by an IIR filter, and is not even causal, some approximation is necessary. A good solution is the **Lanczos kernel**

$$L(s) = \begin{cases} \mathrm{sinc}(s)\,\mathrm{sinc}(\frac{s}{a}) & -a < x < a \\ 0 & \text{else} \end{cases}.$$

a is the size of the Lanczos kernel. The interpolation is then

$$y[t] = \sum_{|r-m|<a} x[t-r]L(m-r)\,.$$

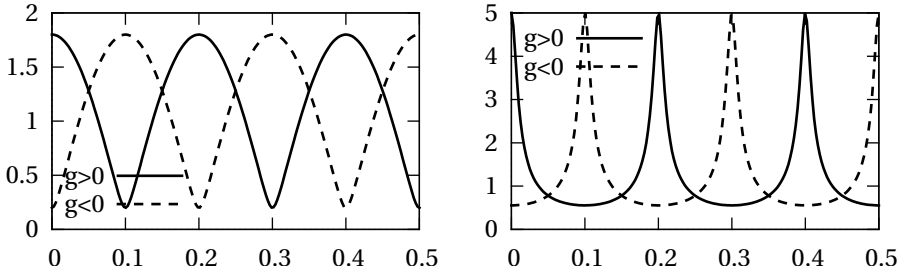
There is also an **allpass interpolation** approach

$$y[t] = (1-f)x[t-\lfloor m \rfloor] + x[t-\lceil m \rceil] - (1-f)\,y[t-1]\,,$$

and interpolation with spline functions.

Apart from the obvious vibrato, there is the **rotary speaker** effect, which was originally produced by real rotating loudspeakers. Two speaker cones oriented in opposite directions emit the same sound and are rotated so that in one moment they point to the left and right, and in the next moment ones points towards the listener and the other away from them. While this causes a variation in amplitude, since the speaker is heard louder when pointing at the listener than when pointing in other directions, it also causes a change in pitch when the cone moves towards the listener or away from them. This effect can be calculated by

$$y[t] = l(1+\sin \beta t)x[t-a(1-\sin \beta t)] + r(1-\sin \beta t)x[t-a(1+\sin \beta t)]\,,$$



(a) FIR comb filter

(b) IIR comb filter

Figure 8: Magnitude response of delay filters with $m = 5$ for $g = 0.8$ and $g = -0.8$.

where β is the rotation speed of the speakers, a is the depth of the pitch modulation, and l and r are the amplitudes of the two speakers, best set to equal values. A stereo effect can be achieved easily by setting l and r to unequal but symmetrical values for the left and right channel. For instance, y_l with $l = 0.7, r = 0.5$, and y_r with $l = 0.5, r = 0.7$.

If the delayed signal is mixed with the direct signal, a multiply mirrored lowpass or highpass filter results, similar to the Wah-Wah effect. The result is a so-called **comb filter**. The first approach is to use a simple FIR filter

$$y[t] = (c * x)[t] = x[t] + g\,x[t-m]\,, \qquad C(z) = 1 + g\,z^{-m}\,,$$

where g is the positive or negative feed-forward parameter. The magnitude response is shown in Figure 8(a). The second approach is an IIR variant.

$$y[t] = (c * x)[t] = x[t] + g\,y[t-m]\,, \qquad C(z) = \frac{1}{1 - g\,z^{-m}}\,,$$

where g is the positive or negative feedback parameter. The magnitude response is shown in Figure 8(b). Note that one is the inverse of the other with negative g. Thus, they could be combined to a delayed version of the first-order allpass filter.

Note also that the IIR comb filter can have a very high gain, which might have to be reduced. In order to retain L_∞-norm so that the range of the signal is not exceeded, the output has to by divided by $1 - |g|$. If unmodified loudness for broadband signals is necessary, the L_2-norm has to be retained by dividing the output by $\sqrt{1 - g^2}$.

Several audio effects can be implemented by delay filters. The **slapback** effect is an FIR comb filter with a delay of 10 to 25 ms (often used in 1950's rock'n'roll).

For delays over 50 ms an **echo** can be heard. For delays of less than 15 ms that are varied by a low-frequency oscillator, a **flanger** effect results. A **chorus** effect is achieved by mixing several delayed signals with the direct signal, where the delays are independent and randomly varied with low frequencies. All these effects can also be implemented with IIR comb filters for more intense effects and repeated slapback or echoes.

A **ring modulator** multiplies a carrier signal $c[t]$ and a modulator signal $m[t]$. For complex signals, their frequencies would be added because, if $c[t] = e^{i\omega_c t}$ and $m[t] = e^{i\omega_m t}$, then

$$c[t]\,m[t] = e^{i\omega_c t}\,e^{i\omega_m t} = e^{i(\omega_c+\omega_m)t}\,.$$

For real signals, however, we have to include mirrored negative frequencies, i.e. $\cos x = \frac{1}{2}(e^x + e^{-x})$. Thus, for $c[t] = \cos\omega_c t$ and $m[t] = \cos\omega_m t$ we get

$$\begin{aligned} c[t]\,m[t] &= \frac{1}{2}(e^{i\omega_c t} + e^{-i\omega_c t})\frac{1}{2}(e^{i\omega_m t} + e^{-i\omega_m t}) \\ &= \frac{1}{4}(e^{i(\omega_c+\omega_m)t} + e^{-i(\omega_c+\omega_m)t} + e^{i(\omega_c-\omega_m)t} + e^{-i(\omega_c-\omega_m)t}) \\ &= \frac{1}{2}(\cos(\omega_c+\omega_m)t + \cos(\omega_c-\omega_m)t)\,. \end{aligned}$$

We see that not only the sum but also the difference of the frequencies is included. The modulator signal usually has lower frequencies than the carrier signal and the carrier signal is a single sine wave, so the positive and negative frequency bands of the modulator signal appear as upper and lower **sidebands** around the carrier frequency. Note that the lower sideband is mirrored, so the higher the modulator frequencies get the lower they get in the lower sideband. The result is a strange non-harmonic sound.

If the roles are reversed, **amplitude modulation** can be implemented by

$$y[t] = (1 + \alpha\, m[t])\,x[t]\,.$$

Here, the modulator is a low-frequency oscillator or something similar with amplitude < 1 and $\alpha < 1$ controls the amount of amplitude variation of the input signal $x[t]$. The result is a **tremolo** effect.

Often we don't want the lower sideband to be audible. In order to get rid of it, we could get rid of the negative frequency band of the modulator and carrier signals first, so no lower sideband would be created in the first place. To do so, we need a filter that produces a 90°phase shift of the input signal which we can add as imaginary part, in order to eliminate negative frequencies. So $\cos\omega t$ should become

$$\cos(\omega t - \frac{\pi}{2}) = \frac{1}{2}(e^{i\omega t - \frac{\pi}{2}} + e^{-(i\omega t - \frac{\pi}{2})}) = \frac{1}{2}(-ie^{i\omega t} + ie^{-i\omega t})\,.$$

This means that the transfer function of the filter should be

$$H(e^{i\omega}) = \begin{cases} -i & \omega > 0 \\ i & \omega < 0 \,. \end{cases}$$

This filter is called **Hilbert filter**. Its impulse response is

$$\begin{aligned} h[t] &= \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{i\omega}) e^{i\omega t} \mathrm{d}\omega \\ &= \frac{1}{2\pi} \left(\int_{-\pi}^{0} -i e^{i\omega t} \mathrm{d}\omega + \int_{0}^{\pi} i e^{i\omega t} \mathrm{d}\omega \right) \\ &= \frac{1}{2\pi} \left(i \frac{e^{i\omega t}}{it} \Big|_{-\pi}^{0} - i \frac{e^{i\omega t}}{it} \Big|_{0}^{\pi} \right) \\ &= \frac{1}{2\pi t} \begin{cases} 1+1+1+1 & t \text{ odd} \\ 1-1-1+1 & t \text{ even} \end{cases} \\ &= \begin{cases} \frac{2}{\pi t} & t \text{ odd} \\ 0 & t \text{ even.} \end{cases} \end{aligned}$$

Unfortunately, this filter has infinite length and is not causal. Therefore, it is approximated by truncating it at $t = \pm 30$ for instance, multiplied by some window function, and shifted in time by 30 to make it causal. When mixed with the direct signal, it also has to be delayed by 30. We write $\hat{x} = h * x$ for the Hilbert-filtered signal.

Now, we can get the analytic version (that without negative frequencies) of c and m as $c + i\hat{c}$ and $m + i\hat{m}$. Multiplying them leads to

$$(c + i\hat{c})(m + i\hat{m}) = cm - \hat{c}\hat{m} + i(c\hat{m} + \hat{c}m) \,.$$

As we are only interested in the real part, we get our **single sideband** modulated signal as $cm - \hat{c}\hat{m}$.

In this way, the modulator signal can be shifted in frequency by f_c. Note, however, that harmonic frequencies $f_m, 2f_m, 3f_m, \ldots$ become non-harmonic after the frequency shift: $(f_m + f_c), (2f_m + f_c), (3f_m + f_c), \ldots$. As a result, a harmonic sound such as a plucked string can sound like a bell or a drum. On the other hand, real string sounds are not perfectly harmonic due to physical impurities, which gives them a warm sound. This effect could be achieved by shifting a perfectly harmonic sound such as a repeating wavetable by a small amount.

2 Nonlinear Processing

In linear processing, the signal values $x[t]$ are modified by linear operations, i.e. addition and multiplication by constant factors, time-varying values or even signals. In nonlinear processing, the signal values are modified by a nonlinear function $g(x)$ so that in the simplest case $y[t] = g(x[t])$. In this way new harmonics are generated and bandwidth expansion takes place. One way to avoid this is to apply signal strength modifications slowly.

In **dynamics processing**, the amplitude of the signal is modified for several purposes such as limiting the amplitude to avoid clipping or cancellation of noise when no other sound is present. In order to decide whether to amplify or attenuate the signal, the amplitude of the signal has to be known. This can be achieved by **amplitude followers**. They are comprised of two parts: the detector and the averager.

A **detector** transforms a signal value in order to approximate the amplitude of the wave. The half-wave **rectifier** simply permits only positive values, $d(x)[t] = \max(0, x[t])$. The full-wave rectifier calculates the absolute value $d(x)[t] = |x[t]|$. The squarer sets $d(x)[t] = x^2[t]$. And the instantaneous envelope is calculated with the help of the Hilbert transform $d(x)[t] = x^2[t] + \hat{x}^2[t]$.

The **averager** then just smoothes the output of the detector in order to avoid a jumping envelope for low frequencies. It can be implemented by a simple low-pass filter

$$y[t] = a(x)[t] = (1-g)x[t] + g\,y[t-1], \qquad \text{where} \qquad g = e^{-\frac{1}{\tau}},$$

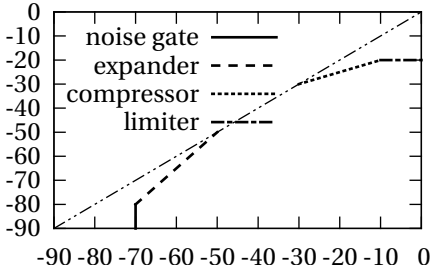
and τ is an attack and release time constant in samples. In order to have shorter attack than release times, two different constants may be used in the following way:

$$y[t] = a(x)[t] = \begin{cases} (1-g_a)x[t] + g_a\,y[t-1] & y[t-1] < x[t] \\ (1-g_r)x[t] + g_r\,y[t-1] & y[t-1] \geq x[t]. \end{cases}$$

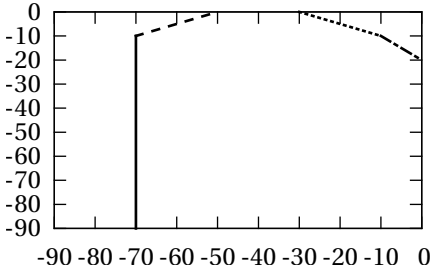
Dynamic range control is then performed by calculating a gain factor from the signal level to multiply the direct signal. The gain factor calculation r is done in the logarithmic domain to get linear level curves. See Figure 9 for typical operations.

$$y[t] = x[t-\tau] \cdot a_2(\exp(r(\log(a_1(d(x))))))[t].$$

In order to smooth out transitions of the gain factor, the gain factor is processed by a second averager a_2 with usually much longer attack and release times τ_2. Because the amplitude follower $a_1(d)$ and the second averager need some time to



(a) Output level over input level.



(b) Gain factor r over input level.

Figure 9: Dynamic range control. All levels and factors in dB, maximum level is 0 dB.

respond to changing input levels, the direct signal is delayed by τ. In this way the output level can be reduced smoothly before a sudden rise in input level would make the output level exceed the allowed range.

A **compressor** reduces the amplitude of loud signals so that the difference between loud and quiet signals is lessened. An **expander** does the opposite for quiet signals to increase the liveliness of the sound. Both use an RMS-style amplitude follower, i.e. a squarer as detector. Typical values for detector and adjustment times are $\tau_{1,a} = 5\,\text{ms}$, $\tau_{1,r} = 130\,\text{ms}$, $\tau_{2,a} = 1\ldots100\,\text{ms}$, $\tau_{2,r} = 20\ldots5000\,\text{ms}$. A **noise gate** entirely eliminates signals below a threshold below which no useful signal is expected. This makes noise disappear which would only be audible when no other sound is present. The purpose of a **limiter** is to reduce peaks in the audio signal. Therefore, it uses a rectifier as level detector. The attack and release times are faster than for compressors. An **infinite limiter**, or **clipper**, is basically a limiter with zero attack and release times. It operates directly on the signal and cuts off samples that exceed the clipping level. A polynomial curve below the clipping level may be used in order to reduce the distortion of hard clipping.

Such a function may be approximated by a Taylor expansion $g(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots$. It operates on the signal as

$$y[t] = g(x[t])\,,$$

To see what that might do to the frequency spectrum of a single oscillation, we

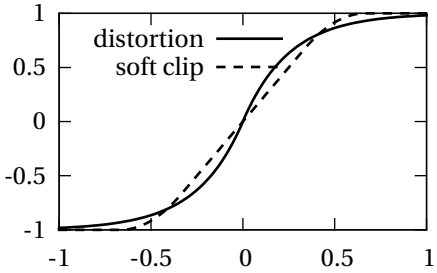


Figure 10: Distortion transforms.

look at

$$\cos^n(\omega t+\varphi)=\frac{1}{2^n}\sum_{k=0}^{n}\binom{n}{k}\cos((n-2k)(\omega t+\varphi))\,.$$

From this we see that a single exponentiation can introduce a number of new frequencies into the signal. For a whole polynomial, all integer multiples $\omega, 2\omega, 3\omega, \ldots$ of the original frequency ω will be present.

The amount of distortion this causes to the original sine wave can be calculated by the **total harmonic distortion**

$$\mathrm{THD}=\sqrt{\frac{A_2^2+A_3^2+A_4^2+\ldots}{A_1^2+A_2^2+A_3^2+\ldots}},$$

where A_k is the amplitude of frequency $k\omega$.

When there is more than one frequency in the input signal, the situation is even more critical. For two sine waves we get

$$(\cos\omega_1 t+\cos\omega_2 t)^n=\sum_{k=0}^{n}\binom{n}{k}\cos^k\omega_1 t\cos^{n-k}\omega_2 t\,.$$

So the sine waves and their harmonics are multiplied. And because we have learned that this produces the sum and difference of frequencies, we see that all frequencies $a\omega_1+b\omega_2$ for integers a and b will be present. For almost but not-perfect harmonic input signals, as often is the case for real recorded sounds, this means that a certain range around each frequency will be filled by sound. This can be heard as a warmth of the sound or, when the distortion is greater, a fuzziness of the sound.

Distortion based on such a transform $g(x)$ can be found in valve (tube) amplifiers and distortion effects. An effect similar to soft clipping can be implemented with

$$g(x) = \operatorname{sign}(x) \cdot \begin{cases} 2|x| & 0 \le |x| \le \frac{1}{3} \\ \frac{3-(2-3|x|)^2}{3} & \frac{1}{3} \le |x| \le \frac{2}{3} \\ 1 & \frac{2}{3} \le |x| \le 1\,. \end{cases}$$

For distortion, the following can be used:

$$g(x) = \operatorname{sign}(x)(1 - \mathrm{e}^{a|x|})\,,$$

where a controls the amount of distortion. See Figure 10. Depending on the amount of distortion, the following terms are used: **overdrive** is a small amount of distortion which makes the sound "warmer", **distortion** is clearly audible distortion where the original sound is still recognizable, and **fuzz** is heavy distortion where only single notes/tones can be played because mutual interaction between several notes would result in noise.

An **exciter** uses light distortion in order to increase the harmonics of a sound. It is often used on vocals and speech, i.e. signal that lack high frequency content, to produce a brighter and clearer sound. An **enhancer** is very similar but also uses equalization to shape the harmonic content.

A more extreme way to modify a signal in a nonlinear way is to use rectifiers to move a signal one or even two octaves up or down. Such effects are called **octavers**. For instance, a full-wave rectifier $g(x) = |x|$ transforms a sine-wave with wave-length τ into a $\frac{\tau}{2}$-periodic signal because the second (negative) half of the wave is now equal to the first half of the wave. Therefore, only even multiples of the original frequency are present, which means an upwards octave shift. For the other direction, zero-crossings of the signal are counted and this information is used to suppress all but every second positive half-wave. Another possibility is to invert every second wave. The result is a signal that is 2τ-periodic, which means a downwards octave shift. Because this turns out to sound very mechanical and synthetic, some octavers apply a low-pass filter to single out the half-frequency and mix that with the original signal.

An important thing to note when distorting discrete signals is that the bandwidth (i.e. the highest frequency in the signal) is multiplied by n if the distortion function $g(x)$ contains and x^n term. These frequencies may be aliased back into the frequency range from 0 to Nyquist-frequency. While this may be wanted as a source of even more distortion, there are two methods to avoid the aliasing. The first method is to upsample the signal by a factor of n using interpolation, creating $n-1$ new samples between two existing ones. The new frequencies from

distortion are then below the new Nyquist-frequency. After that, down-sampling is applied to return to the original sampling rate, where the signal has to be low-pass filtered to eliminate the frequencies that would cause aliasing. Both interpolation and anti-aliasing may use the Lanczos filter.

The second method is to split $g(x)$ into $a_1x + a_2x^2 + a_3x^3 + \ldots$, then also splitting x into n channels, each low-pass filtered by l_k with a cutoff frequency of $\frac{f_s}{2k}$ before being processed by exponentiation and summing the results.

$$y[t] = a_1x + a_2(l_2 * x)^2 + a_3(l_3 * x)^3 + \ldots .$$

Thus, the bandwidth enlargement of the low-pass filtered signals will not reach the aliasing region.

3 Time-Frequency Processing

The signals we investigate can be represented by the **sinusoidal+residual model**. In this model the signal is a sum of sinusoids of different time-varying frequencies and time-varying amplitudes, plus a residual signal with no particular frequency, i.e. noise with a certain time-varying spectral shape.

$$x[t] = \sum_k a_k[t]\cos(\varphi_k[t]) + e[t] .$$

where $a_k[t]$ is the amplitude of the k-th sinusoid, $e[t]$ is the residual signal, and $\varphi_k[t]$ is the instantaneous phase of the k-th sinusoid that cumulates the instantaneous frequency $\omega_k[t]$:

$$\varphi_k[t] = \sum_{s=0}^{t} \omega_k[s] .$$

The goal is now to extract these sinusoids from the signal $x[t]$, apply some modifications, and put them back together. The most common method is time-frequency processing.

In time-frequency processing, the audio signal is split in time into blocks (or frames) which are transformed by a Fourier transform. This is called **short-time Fourier transform** (**STFT**). Because the Fourier transform is made for signals that are periodic with the block length n, a signal that has a different period fills all frequency bins, where the amplitude drops only with $1/(\omega - \omega_0)$ where ω_0 is the frequency of the signal. To avoid this, a window function $h[t]$ is applied so that the narrow low-pass frequency response of the window is modulated by the signal frequency and, thus, moved as a window in the frequency domain to the frequency of the signal and attenuates frequency bins with a greater distance from the frequency of the signal.

The window or frame size n is always a compromise because a small window size leads to bad frequency resolution and a large window size leads to bad time resolution and higher latency. The **hop size** r is the distance between the centers of two consecutive windows. If the window size is greater than the hop size, then there is an **overlap** of windows. The percentage of overlap is defined by $1 - r/n$. Big overlaps lead to smoother transitions of the spectral feature points, but is computationally more expensive.

The STFT is given by

$$X[t,w] = \sum_{s=-n/2}^{n/2-1} h[s]x[rt+s]\mathrm{e}^{-\mathrm{i}2\pi ws/n}$$

Thus, the signal is now given in several frequency bands w with coarser time-resolution t. $X[t,w]$ contains amplitude and phase information which can be modified separately. After that, the signal is usually reconstructed in the time domain by **re-synthesis**. This can be done by the inverse Fourier transform. Another window function, the synthesis window h_s, has to be applied for two reasons: First, the analysis window has to be reversed. Second, in overlap regions the sum of the resulting windows has to be 1.

$$x[t] = \sum_{s:-\frac{n}{2}\le t-rs<\frac{n}{2}} h_s[t-rs]\sum_w X[s,w]\mathrm{e}^{\mathrm{i}2\pi w(t-rs)}.$$

This is called the **overlap-add** method. The summing condition is, therefore, given by

$$\sum_s h[t-rs]h_s[t-rs] = 1.$$

If the analysis and the synthesis window are the same, then the sum of squares of the window has to be 1. This is true for the Hann window, $h[t] = \frac{A}{2}(1+\cos 2\pi t/n)$, with a hop size of $r = n/4$, because for $t \ge n/4$:

$$\begin{aligned}
&h^2[t] + h^2[t-n/4] + h^2[t-n/2] + h^2[t-3n/4] \\
&= \frac{A^2}{4}(1+\cos 2\pi t/n)^2 + \frac{A^2}{4}(1+\cos 2\pi(t/n-1/2))^2 + \ldots + \frac{A^2}{4}(1+\cos 2\pi(t/n-3/4))^2 \\
&= \frac{A^2}{4}(1+\cos)^2 + \frac{A^2}{4}(1-\sin)^2 + \frac{A^2}{4}(1-\cos)^2 + \frac{A^2}{4}(1+\sin)^2 \\
&= \frac{A^2}{4}(1+2\cos+\cos^2+1-2\sin+\sin^2+1-2\cos+\cos^2+1+2\sin+\sin^2) \\
&= \frac{A^2}{4}(4+2(\cos^2+\sin^2)) = \frac{3A^2}{2} \overset{A=\sqrt{2/3}}{=\!=\!=} 1.
\end{aligned}$$

The method of STFT, followed by modifications of the result and inverse STFT, is called **phase vocoder** for historical reasons. The most common uses for the phase vocoder is time stretching and pitch shifting.

For **time stretching**, the idea is simply to use a different hop size r_s for synthesis than for analysis. There is one problem, however: If the frequency of a signal is not changed and a frame is shifted in time, then the phases have to be adjusted. To do this, we first need **phase unwrapping**. Let $\varphi[t,w]$ be the instantaneous phase of the STFT coefficient $X[t,w]$, so that

$$X[t,w] = A[t,w]\mathrm{e}^{\mathrm{i}\varphi[t,w]}.$$

Now, if the frequency would be exactly w, then the projected phase of $X[t+1,w]$ is

$$\varphi_p[t+1,w] = \varphi[t,w] + 2\pi wr/n,$$

which should be equal to the real phase $\varphi[t+1,w]$ modulo 2π. However, since the real frequency is not necessarily in the center of the frequency bin w, there is some difference. The unwrapped phase $\varphi_u[t+1,w]$ is, therefore, set so that

$$\varphi_u[t+1,w] = \varphi[t+1,w] \mod 2\pi, \qquad -\pi \le \varphi_u[t+1,w] - \varphi_p[t+1,w] \le \pi.$$

The total phase rotation between time t and $t+1$ in frequency bin w is then

$$\Delta\varphi[t+1,w] = \varphi_u[t+1,w] - \varphi[t,w].$$

Now back to time stretching. As said, we need to adjust the phases if we move from hop size r to hop size r_s. Our new frequency coefficients shall be

$$Y[t,w] = \sum_{s=-n/2}^{n/2-1} h[s]\,y[r_s t + s]\mathrm{e}^{-\mathrm{i}2\pi ws/n} = A[t,w]\mathrm{e}^{\mathrm{i}\psi[t,w]}.$$

The total phase rotation between t and $t+1$ must now be greater by a factor of r_s/r:

$$\psi[t+1,w] = \psi[t,w] + \frac{r_s}{r}\Delta\varphi[t+1,w].$$

Pitch shifting can be reduced to time stretching simply by applying resampling after time stretching to restore the original rate of frames per second. Note the pitch shifting is different from frequency shifting, as it is done with single sideband modulation. Frequency shifting adds a certain delta frequency to every frequency in the signal. Pitch shifting *multiplies* each frequency by a factor α. To achieve this with time stretching, we set $r_s = \alpha r$. After time stretching, the

resampling calculates $y[t] = x[\alpha t]$. Because time stretching does not modify the frequencies, the result has shifted frequencies by the factor α because $\cos(\omega t)$ becomes $\cos(\omega\alpha t)$.

It turns out that time stretching and pitch shifting works well for sums of sinusoids with slowly varying amplitude and frequency, but it has problems with amplitude and frequency transients, and noise such as consonants in speech. These sounds tend to be smeared in time. A possibility to cope with this is to separate stable from transient components. A frequency bin is defined as belonging to a stable sinusoid if the phase change itself does not change too much. More precisely,

$$\varphi[t,w] - \varphi[t-1,w] \approx \varphi[t-1,w] - \varphi[t-2,w] \mod 2\pi,$$

or even more precisely,

$$|\varphi[t,w] - 2\varphi[t-1,w] + \varphi[t-2,w]| < d \mod 2\pi,$$

where "$x < d \mod 2\pi$" means that the smallest non-negative $x + k \cdot 2\pi$ is smaller than d. Stable frequency bins are now subject to time stretching as explained, while transient ones are either dropped or used to construct the residual signal.

The **mutation** (**morphing**, **cross-synthesis**) of two sounds can be achieved by combining the time-frequency representation of two sounds. The most typical vocoder effect is to use the phase of one sound X_1 (from a keyboard for instance) and the magnitude of another sound X_2 (a voice for instance).

$$Y[t,w] = \frac{X_1[t,w]}{|X_1[t,w]|}|X_2[t,w]|.$$

In this way, the harmonic content, i.e. phases and therefore frequencies, of the first sound is modified to have the spectral shape of the second sound, so that the vowels can be heard as such, because vowels are defined by the spectral shape. As a very similar effect, **robotization** can be achieved by using only X_2 and setting all phases to zero in each frame and each bin. The result will be periodic with the hop size as period, and, therefore, have constant pitch. If the phase is randomized, then a **whisperization** effect is produced. For this effect, the frame and hop size must not be too large, lest the bin magnitudes will represent the frequencies too well.

Denoising is achieved by attenuating frequency bins with low magnitude while keeping high magnitudes unchanged. This may be done by a nonlinear function such as

$$Y[t,w] = X[t,w]\frac{|X[t,w]|}{|X[t,w]| + c_w},$$

where c_w is a parameter that controls the amount and level of attenuation. It can be chosen differently for different frequencies, so that noise levels as measured in a recording of silence are sufficiently suppressed while leaving frequencies with little noise content as unmodified as possible.

The main problem with the traditional phase vocoder techniques as presented so far is that sinusoids are not really extracted but spread over several neighboring frequency bins and possibly even overlap. A more recent development is to find and separate individual sinusoids by finding local peaks in the spectrum. These peaks can be located more precisely than the frequency resolution seems to allow by applying interpolation.

In **peak detection**, local maxima in the magnitude spectrum are found and associated with a sinusoid. Note that this association is not perfect because of noise, side lobes and spectrum overlaps. Moreover, the local maximum would only be accurate up to half a bin width in frequency, i.e. up to f_s/N, where f_s is the sampling rate and n is the Fourier transform size. To improve this, one could enlarge n by zero padding of data. Another possibility is to fit a parabola to the maximum and the two neighboring bins in the logarithmic representation of the magnitudes, and find the peak of the parabola. Let $a_w = 10\log_{10}|X[t, w_0 + w]|_2^2$, where w_0 is the bin of the local maximum. We want to fit a parabola $p(w) = \alpha w^2 + \beta w + \gamma$ so that $p(w) = a_w$ for $w \in \{-1, 0, 1\}$. This results in $\alpha - \beta + \gamma = a_{-1}$, $\gamma = a_0$, $\alpha + \beta + \gamma = a_1$, and from that $\alpha = \frac{1}{2}(a_1 - 2a_0 + a_{-1})$, $\beta = \frac{1}{2}(a_1 - a_{-1})$. Now, in order to find the peak of $p(w)$, we set $p'(w) = 0$, which leads to $2\alpha w + \beta = 0$. In this way we get

$$w = -\frac{\beta}{2\alpha} = \frac{a_{-1} - a_1}{2(a_{-1} - 2a_0 + a_1)}.$$

In **pitch detection**, the goal is to find the fundamental frequency whose integer multiples are called harmonics or partials and should cover all detected frequency peaks. As the fundamental frequency is not necessarily the peak with the highest magnitude, as it can even be missing entirely, this is not an easy task. There are several heuristic approaches. Most suggest a set of candidates by using the most prominent peaks and some integer fractions of them. Then the difference between the harmonics of the candidates and the measured peaks is calculated and the best match is chosen. Pitch detection can also improve the peak detection by dropping the peaks that don't fit in the detected pitch, assuming that those are probably sidelobes of real harmonics or just noise.

Another way to improve the peak detection is to look at the temporal development of the peaks so as to promote peaks that continue peaks of previous frames. This is called **peak continuation**. A simple way to do this is to assign to each peak

the one of the next frame that is closest in frequency. In the presence of noise and transients, this method is error prone, though, and the sinusoid trajectories may switch between different partials. A better approach is to set up "guides" that represent the current position of partials. In every new frame, these guides are updated in order to best match the fundamental frequency and the peaks. Guides may be turned off temporarily, killed entirely or created when new unmatched peaks appear. The result is a set of sinusoid trajectories which can be modified and synthesized into the reconstructed output signal.

The result of this process is a set of sinusoids with amplitudes and frequencies sampled at hop-size intervals. This is often called a *tracks* representation of the sound. In order to convert this representation back into the time domain, **synthesis** methods are required that are not as straight forward as the inverse FFT. The first method works in the time domain and implements each sinusoid by an oscillator.

The **oscillator** is a single wave signal that satisfies the following differential equation:

$$x''(t) = -ax(t)\,,$$

which means that the acceleration is negatively proportional to the amplitude. To turn this into a discrete version, we approximate the second derivative by

$$x''(t) \approx x[t+1] - 2x[t] + x[t-1]\,,$$

which gives

$$x[t+1] = (2-a)x[t] - x[t-1] =: (r * x)[t+1]\,.$$

This corresponds to an IIR filter r without excitation by an input signal, which is called the **digital resonator**. It has the transfer function

$$R(z) = \frac{1}{1-(2-a)z^{-1}+z^{-2}}\,,$$

and the pole of $R(z)$ is located at the actual frequency of oscillator. To examine this, we set the denominator to zero and get

$$\begin{aligned}(2-a)z^{-1} &= 1+z^{-2}\\ (2-a) &= z+z^{-1} = 2\cos\omega\,,\end{aligned}$$

so we can substitute $2\cos\omega$ for the factor $(2-a)$ in order to synthesize the frequency ω. The oscillator has to be initialized by calculating $x[0]$ and $x[1]$ directly.

This also has to be done when the frequency changes, i.e. when the factor $(2-a)$ changes. This can be seen from the following energy function.

$$E[t] = ax[t]x[t-1] + (x[t] - x[t-1])^2 .$$

It consists of two parts. The first one represents the potential energy, the second one the kinetic energy. $E[t]$ has the property that it remains constant if $x[t]$ evolves after the digital resonator scheme. We will show this:

$$\begin{aligned}
E[t+1] &= ax[t+1]x[t] + (x[t+1] - x[t])^2 \\
&= a((2-a)x[t] - x[t-1])x[t] + ((2-a)x[t] - x[t-1] - x[t])^2 \\
&= a(2-a)x[t]^2 - ax[t]x[t-1] + (x[t] - x[t-1] - ax[t])^2 \\
&= a(2-a)x[t]^2 - ax[t]x[t-1] + (x[t] - x[t-1])^2 - 2ax[t](x[t] - x[t-1]) + a^2x[t]^2 \\
&= a(2-a)x[t]^2 - ax[t]x[t-1] + (x[t] - x[t-1])^2 - a(2-a)x[t]^2 + 2ax[t]x[t-1] \\
&= ax[t]x[t-1] + (x[t] - x[t-1])^2 = E[t] .
\end{aligned}$$

What does this mean for the amplitude of the signal? When the signal reaches its maximum, then there is almost no kinetic energy so we have

$$E[t] \approx ax[t]x[t-1] \approx ax[t]^2 .$$

When a changes to a_2 in this situation, we get an oscillation with changed frequency, changed energy by a_2/a, but equal amplitude, which is desirable. However, when a changes at a zero crossing, i.e. when there is only kinetic energy, the energy remains the same. And this means that the amplitude will be changed because at the next peak the energy $ax[t]^2$ will still be the same, which can only be achieved by a changed amplitude because a has changed. This has to be compensated or, better, the signal has to be initialized again.

The second method for sinusoid synthesis is **synthesis by inverse Fourier transform**. Here, the spectral pattern of a sinusoid is added to the bins in the frequency domain, followed by an inverse Fourier transform and the application of a synthesis window, just as in the phase vocoder. To do this, first a pure sine wave has to be windowed and transformed in order to get the proper coefficients. These can be stored in a table and copied into the bins of a frame when needed.

Fortunately, not all combinations of frequencies, amplitudes and phases have to be stored. Amplitudes can be adjusted by simply multiplying the coefficients, so only a normed amplitude has to be stored. Similarly, the phase can also be adjusted by multiplication with $e^{i\phi}$. Moreover, as all coefficients of a single sinusoid should have the same phase, no phase information has to be stored at all.

Also, coefficients for two frequencies with an integer bin-distance are exactly the same, just shifted by a certain number of bins; so only coefficients for frequencies between bin 0 and 1 have to be stored. And finally, coefficients far from the frequency of the sinusoid are negligibly small, so only a small number of bins around the center frequency has to be considered. To sum up, we need the following coefficients:

$$C_f[w] = \sum_{s=-n/2}^{n/2-1} h[s] \mathrm{e}^{\mathrm{i}2\pi f s/n} \mathrm{e}^{-\mathrm{i}2\pi w s/n} = \sum_{s=-n/2}^{n/2-1} h[s] \mathrm{e}^{-\mathrm{i}2\pi (w-f) s/n},$$

where $w = -b, \ldots, b$ is integer, b is the approximation bandwidth, and $f \in [0,1)$, or better $f \in [-0.5, 0.5)$, is real. w and f can even be combined into $v = w - f$:

$$C(v) = \sum_{s=-n/2}^{n/2-1} h[s] \mathrm{e}^{-\mathrm{i}2\pi v s/n}.$$

This can be implemented by a strongly zero-padded Fourier transform of the window $h[s]$ for arbitrary detailed resolution of v. Higher resolutions can also be achieved by interpolation of $C(v)$. For symmetric windows $h[s]$, the coefficients $C(v)$ will be real.

Altogether, for the synthesis of a sinusoid with frequency f, amplitude A and instantaneous phase φ, we have to copy $AC(w-f)\mathrm{e}^{\mathrm{i}\varphi}$ into bin w. What are the benefits of this method? It seems at first that it is slower than the resonator method because, whereas the latter only requires one multiply- and one add-operation per sample, the IFFT-method requires $O(n \log n)$ operations per frame of size n for the inverse FFT, which means $O(\log n)/(1-\text{overlap})$ operations per sample. However, the $O(b)$ operations to fill the bins only have to be executed once for a frame. Therefore, the IFFT method will be faster if a large number of sinusoids have to be synthesized, because the inverse FFT has to be performed only once.

A problem with overlap-add IFFT synthesis is that a change in frequency can lead to interferences in the overlap regions. Also, the more overlap the lower the computational efficiency. Therefore, there exists an approach to use no overlap at all. The result of the inverse Fourier transform has to be inverse windowed with $h[s]^{-1}$. Depending on the bandwidth b, there will be approximation errors, especially at the borders. As a countermeasure, either the bandwidth b could be increased, or a bit of the border can be truncated. Both methods increase the computational complexity, either by increasing the work to fill more frequency bins, or by reducing the hop size, which is now equal to the FFT size minus the truncation, while the amount of computation per hop remains almost the same.

However, if the best compromise of b and truncation is found, the method turns out to be more efficient and has no overlap interference problems. Phases have to be calculated exactly so that frequency changes happen without phase jumps at the border.

The **residual signal** is found by subtracting the re-synthesized signal from the original signal. This can be in the time domain or in the frequency domain. If it is done in the time domain, then the window and hop sizes can be reduced. This is preferable because frequency resolution is not that important for the residual signal, while time resolution should be higher in order to better represent short noises such as consonants in speech or tone onsets of instruments. If the subtraction is done in the frequency domain, however, then no additional FFT has to be performed for the residual analysis.

The residual signal is – or should be – a stochastic signal, which means that only the spectral shape without the phase information is necessary for sufficient reproduction of the sound. The analysis of the signal is done in the frequency domain by curve fitting on the magnitude spectrum. The simplest case would be straight-line segment approximation: the frequency domain is decomposed into equally or logarithmically spaced sections, then the maximum magnitude is found in each segment, and each segment is substituted by a point with this magnitude; the points are linearly interpolated by straight-line segments. The segment number and sizes can be adjusted to the complexity of the sound. Another possibility would be spline interpolation.

The synthesis of the residual signal could be done by a convolution of white noise and the impulse response of the magnitude spectrum. A better way is, of course, to fill each frequency bin with a complex value that has the magnitude from the measured magnitude spectrum and a random phase. The phases must be re-randomized in each frame in order to avoid periodicity.

A simple application of the above method is a filter with arbitrary resolution. As we know the exact frequency of the involved sinusoids, we can drop them if they are slightly out of a specified range. This results in a very steep transition band which could hardly be achieved by a normal filter.

Pitch shifting can be implemented in this scheme very easily. The frequency of each sinusoid can be shifted or scaled individually. It is also possible to apply **timbre preservation**, which means that the spectral shape should remain the same while the frequencies are shifted. As an approximation of the spectral shape, linear or spline interpolation of the magnitudes between the sinusoids is calculated at the position of the shifted or scaled frequencies, and the interpo-

lated magnitude is used as the new magnitude of the sinusoid.

Time stretching can also be implemented in this scheme. The hop-size can be the same for analysis and synthesis. However, because the frames are read at a different rate in analysis as they are written in synthesis, sometimes analysis frames are used twice when time is stretched, or not at all when time is compressed. To avoid the smoothing of attack transients, analysis and synthesis frame rates can be set equal for a short time. Attack transients can be detected by fast changing energies in certain frequency ranges.

Pitch correction is achieved by first detecting the pitch of a signal, then quantifying it towards the nearest of the 12 semitones of the octave. All sinusoids are the pitch-scaled by the same factor so that the pitch matches the correct semitone. This enables unskilled singers to sing in perfect tune. It was popularized as a recording and performance effect by the Auto-Tune software.

A **spectral shape shift** is the opposite of pitch shifting with timbre preservation. The frequencies of the sinusoids remain the same while the spectral shape is moved up or down the frequency scale. This can change the timbre of a sound without changing its pitch. **Gender change** can be achieved by a combination of pitch scaling, by an octave for instance, and moving the spectral shape along with the pitch if the target gender is female, as this is a feature of the female voice. In the female-to-male case, the spectral shape has to be moved in the opposite direction to remove the feature.

Hoarseness can be simulated by simply increasing the magnitude of the residual signal.

Another way to represent the spectral shape of a signal is **linear predictive coding** (**LPC**). It models the signal $x[t]$ with a filter p that predicts $x[t]$ from previous values $x[t-k]$ so that the difference, the residual signal $e[t] = x[t]-(p*x)[t]$ is as small as possible.

$$(p*x)[t] = p[1]x[t-1] + p[2]x[t-2] + \ldots + p[m]x[t-m]\,.$$

To re-synthesize the signal from p, one uses $x[t] = (p*x)[t] + e[t]$. If the residual signal $e[t]$ is not known exactly because it has been quantized to $\tilde{e}[t]$ for data-compression purposes, or if it is substituted entirely by a new excitation signal $\tilde{e}[t]$ (or source signal), then we get

$$y[t] = (p*y)[t] + \tilde{e}[t]\,,$$

which is an all-pole IIR filter, similar to the digital resonator.

The question is now how to find the optimum filter coefficients $p[k]$ that minimize the residual signal. What we want to minimize is

$$E := \sum_t e^2[t] = \sum_t (x[t] - p[1]x[t-1] - p[2]x[t-2] - \ldots - p[m]x[t-m])^2 .$$

The optimum is found by deriving this with respect to all $p[k]$ and setting the result to zero.

$$0 = \frac{\mathrm{d}E}{\mathrm{d}p[k]} = \sum_t 2e[t]\frac{\mathrm{d}e[t]}{\mathrm{d}p[k]} = 2\sum_t e[t]x[t-k] = 2\sum_t \left(x[t] - \sum_j p[j]x[t-j] \right) x[t-k]$$

$$\Leftrightarrow \quad \sum_j p[j] \sum_t x[t-j]x[t-k] = \sum_t x[t]x[t-k] .$$

This a system of equations involving the autocorrelation of x, which can be substituted by a windowed version in order to get more stable filter coefficients.

$$r_{xx}[s] := \sum_t w[t]x[t]\, w[t-s]x[t-s] .$$

Thus, we get

$$\sum_j p[j] r_{xx}[k-j] = r_{xx}[k] ,$$

which is an equation system in the form of a Toeplitz matrix, i.e. it has constant diagonals $M_{k,k-i} = r_{xx}[k-(k-i)] = r_{xx}[i]$.

Such a system is best solved with the **Levinson-Durbin recursion**. Let $T^{(n)}$ be the upper left $n \times n$-sub-matrix of $M_{k,j} = r_{xx}[k-j]$, and $p^{(n)}$ the solution vector of $T^{(n)} p^{(n)} = y^{(n)}$ where $y^{(n)} = r_{xx}[1 \ldots n]$. Then

$$T^{(n+1)} \begin{pmatrix} p^{(n)} \\ 0 \end{pmatrix} = \begin{pmatrix} y^{(n)} \\ \epsilon \end{pmatrix} .$$

We want $r_{xx}[n]$ instead of ϵ, though. With the help of a vector $b^{(n)}$ which satisfies $T^{(n)} b^{(n)} = (0, \ldots, 0, 1)$ we can calculate

$$T^{(n+1)} p^{(n+1)} = T^{(n+1)} \left(\begin{pmatrix} p^{(n)} \\ 0 \end{pmatrix} + (r_{xx}[n] - \epsilon) b^{(n+1)} \right) = y^{(n+1)} .$$

Now we have to find those vectors $b^{(n)}$. For that, we will simultaneously find vectors $f^{(n)}$ satisfying $T^{(n)} f^{(n)} = (1, 0, \ldots, 0)$. Also in a recursive approach we get

$$T^{(n+1)} \begin{pmatrix} f^{(n)} \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ \epsilon_f \end{pmatrix} , \qquad T^{(n+1)} \begin{pmatrix} 0 \\ b^{(n)} \end{pmatrix} = \begin{pmatrix} \epsilon_b \\ 0 \\ \vdots \\ 1 \end{pmatrix} .$$

Now we find α and β so that

$$T^{(n+1)}b^{(n+1)} = T^{(n+1)}\left(\alpha\begin{pmatrix} f^{(n)} \\ 0 \end{pmatrix} + \beta\begin{pmatrix} 0 \\ b^{(n)} \end{pmatrix}\right) = \begin{pmatrix} \vdots \\ 0 \\ 1 \end{pmatrix},$$

which can be found by solving

$$\alpha + \beta\epsilon_b = 0\,, \quad \alpha\epsilon_f + \beta = 1\,.$$

The same has to be done to find $f^{(n+1)}$. Thus, by recursion from $n = 1$ to m (the length of filter p), the optimal filter coefficients can be found in $O(m^2)$ complexity, compared to $O(m^3)$ of normal equation solving.

To use this method, it is important to see that the order (length) m of the filter p determines how exact the spectral representation of the signal is. Low-order filters represent a coarse approximation of the spectrum, which corresponds to the spectral shape as in peak-interpolation. Also, when used as FIR filter on the source signal, the residual signal is "whitened", i.e. the spectrum is made flatter. This can be used for sound **mutation** by filtering the residual of a signal x_1 with the LPC-filter p_2 of signal x_2.

$$y[t] = (x_1 - p_1 * x_1)[t] + (p_2 * y)[t]\,.$$

The LPC-method is very well suited for speech processing, as the filter represents the formants of the vowels. Thus, the method is widely used in speech analysis, synthesis and compression.

Yet another method to represent the spectral shape is the **cepstrum**. It is basically a smoothing of the magnitude spectrum by a Fourier method. The first part is to inversely Fourier-transform the logarithm of the magnitude spectrum.

$$c[t,s] := \frac{1}{n}\sum_{w=-n/2}^{n/2-1} \log|X[t,w]|\mathrm{e}^{\mathrm{i}2\pi ws/n}\,.$$

The result is called the *real cepstrum* – the normal cepstrum would include the phase as an imaginary part before the inverse Fourier transform. It is then low-pass filtered in the s-domain by a window

$$l[s] = \begin{cases} 1 & -s_c \le s < s_c \\ 0 & \text{else}\,, \end{cases}$$

where s_c is the cutoff "quefrency". The forward Fourier transform then yields a smoothed spectrum in the logarithmic domain (dB):

$$C_l[t,w] = \sum_{s=-n/2}^{n/2-1} c[t,s]\,l[s]\mathrm{e}^{-\mathrm{i}2\pi ws/n}.$$

By using a high-pass window $h[s] = 1 - l[s]$ instead of $l[s]$, we get the complementary source envelope which satisfies

$$\log|X[t,w]| = C_l[t,w] + C_h[t,w],$$

or, according to the properties of the logarithm:

$$X[t,w] = \exp(C_l[t,w])\exp(C_h[t,w])\mathrm{e}^{\mathrm{i}\varphi[t,w]}.$$

So, $\exp(C_l[t,w])$ is the filter or spectral envelope, whereas $\exp(C_h[t,w])\mathrm{e}^{\mathrm{i}\varphi[t,w]}$ is the source signal (source-filter separation).

Sound mutation can again be implemented with this scheme by calculating the envelopes of two signals $x^{(1)}$ and $x^{(2)}$, and putting them together in the following way:

$$\begin{aligned} Y[t,w] &= \exp(C_l^{(1)}[t,w])\exp(C_h^{(2)}[t,w])\mathrm{e}^{\mathrm{i}\varphi^{(2)}[t,w]} \\ &= X^{(2)}[t,w]\exp(-C_l^{(2)}[t,w])\exp(C_l^{(1)}[t,w]). \end{aligned}$$

In the same style, formant changing can be done by scaling the spectral envelope:

$$\begin{aligned} Y[t,w] &= X[t,w]\exp(-C_l[t,w])\exp(C_l[t,w/k]) \\ &= X[t,w]\exp(C_l[t,w/k] - C_l[t,w]), \end{aligned}$$

where k is the scale factor. Of course, in case of pitch shifting, this method can be used to achieve timbre preservation by whitening the signal before pitch shifting and applying the spectral envelope afterwards.

The cepstrum can be used for **pitch detection**. Since the fundamental frequency and the partials of a harmonic sound appear in regular intervals in the frequency domain with a period that is equal to the fundamental frequency, the cepstrum will have a peak for this period. There will also be peaks for integer factors of the period, so the lowest peak should be chosen. The domain of the cepstrum, i.e. quefrency, is basically a time-measure, representing the period of the fundamental frequency.

4 Time-Domain Methods

Time stretching can be done in the time domain by shifting overlapping short segments of a signal in time. As these time-shifted overlaps would produce phase mismatches and, thus, amplitude fluctuations, the time shift has to be adjusted to avoid this as far as possible. The approach is as follows: The signal is cut into overlapping segments

$$x_k[t] = x[kr + t] \qquad \text{for} \quad t = 0, \ldots, n-1\,,$$

where k is the index of the segment, r is the hop-size, and n is the segment length. Now the hop-size is changed to r'. To account for phase matching, each segment shift can be adjusted by an additional shift s_k. The segments are then overlap-added with the help of a fade-in/fade-out window w_k.

$$y[t] = \sum_k x_k[t - kr' - s_k]\, w_k[t - kr' - s_k]\,.$$

The best fitting shifts s_k can be found by the cross-correlation of subsequent segments

$$c[s] = \sum_t x_{k-1}[t]\, x_k[r' + t + s]\,,$$

and finding the maximum

$$s_k = \arg\max_s c[s]\,.$$

This method is called **SOLA** (synchronous overlap-add). For more extreme scaling, segments can be repeated. This can be done by choosing a source segment $k(l)$ for each destination segment l and proceeding as before.

If the pitch of the signal can be determined, a variant, **PSOLA** (pitch-synchronous overlap-add) can be used. In this case, the shift $r' - r + s_k$ must be a multiple of the pitch period τ. Therefore,

$$s_k = \text{round}\left(\frac{r' - r}{\tau}\right)\tau - (r' - r)\,.$$

Another way to perform **pitch detection** is to use auto-correlation. $r_{xx}[s]$ is supposed to have a peak at a lag of $s = T_0/T_s$, where T_0 is the period of the signal, i.e. $T_0 = 1/f_0$, and T_s is the sampling interval, i.e. $T_s = 1/f_s$. Thus, we get $s = f_s/f_0$.

There are some problems with this. First, the lag has to be an integer value because the auto-correlation is only calculated at integer shifts. Therefore, the detected fundamental frequencies must not be too high, and the sampling rate should be high enough. The second problem is that the fundamental frequency

is not the only peak in the auto-correlation signal. Integer multiples of the fundamental lag also have peaks of the same order because a signal with a period T_s is also periodic with a period of kT_s. Therefore, the first peak should be chosen. However, since harmonics of the fundamental frequency also produce periods of their own, integer fractions of the fundamental lag also produce peaks in the autocorrelation signal.

To improve the second problem, several methods exist, including low-pass filtering in order to make the fundamental frequency more prominent. Another scheme uses long-term prediction. First, instead of auto-correlation, the signal $x[t]$ is predicted from several lags with a one-tap prediction filter:

$$x[t] \approx p_s x[t-s]\,.$$

The residual energy

$$E = \sum_t (x[t] - p_s x[t-s])^2$$

is minimized in the usual way, as in LPC, by setting the derivation to zero.

$$\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}p} = \sum_t 2(x[t] - p_s x[t-s])x[t-s] &= 0\,,\\
\sum_t x[t]x[t-s] &= p_s x[t-s]^2\,,\\
r_{xx}[s] &= p_s r_{xx}[0]\,,\\
p_s &= \frac{r_{xx}[s]}{r_{xx}[0]}\,.
\end{aligned}$$

Now if we put that back into the residual energy, we get

$$\begin{aligned}
E = \sum_t x[t]^2 - 2x[t]p_s x[t-s] + p_s^2 x[t-s]^2 &= r_{xx}[0] - 2\frac{r_{xx}[s]}{r_{xx}[0]} r_{xx}[s] + \frac{r_{xx}^2[s]}{r_{xx}^2[0]} r_{xx}[0]\\
&= r_{xx}[0] - \frac{r_{xx}^2[s]}{r_{xx}[0]} = r_{xx}[0] - r_{xx,\mathrm{norm}}[s]\,.
\end{aligned}$$

The second part depends on the lag s and is a normed squared auto-correlation signal. Its maximum minimizes the residual energy. It can be used to detect peaks for candidates of fundamental lags. Then, $r_{xx}[s]$ has to be checked for positivity. And then, p_s can be used as a quality check because p_s has to be close to 1 for a correct match.

5 Spatial Effects

Spatial effects can be achieved most easily with multiple loudspeakers, in the simplest case with two speakers, i.e. stereo speakers. The most basic spatial effect is **panorama**. It exploits the fact that, if both speakers emit the same signal at different volumes, the apparent source direction tends to the louder speaker. This can be formalized in the following way. Let θ_l be the angle of the speakers measured from a point in front of the listener, so that the speakers are $2\theta_l$ apart. θ be the angle of the apparent direction of the sound, and g_L and g_R are the gains that the speakers are playing at. Then

$$\tan\theta = \frac{g_L - g_R}{g_L + g_R} \tan\theta_l \,.$$

When choosing the gains g_L and g_R, one has to take care to preserve the total loudness of the sound. Linear interpolation (linear panning) does not preserve loudness, it yield a "hole" in the center. The proper way to do it is

$$g_L = \frac{1}{\sqrt{2}}(\cos\varphi + \sin\varphi)\,, \quad g_R = \frac{1}{\sqrt{2}}(\cos\varphi - \sin\varphi)\,,$$

where $\varphi = 45^\circ$ corresponds to the case $\theta = \theta_l$. All this is, however, only true for broadband signals and low frequencies. Therefore, one might choose to use a different φ for higher frequencies.

Another possibility to modify the apparent source direction of a sound is to introduce a short delay of up to 1 ms between the two speakers. The sound will appear to be nearer to the speaker that emits the sound first. This is called the **precedence effect**. It has a similar behavior to the panorama effect, but depending on the time difference between left and right speaker. However, the effect strongly depends on the type of sound being played and the frequency.

When operating with headphones **inter-aural differences** play a big role. The inter-aural intensity difference (IID) is basically a panorama effect, but it depends on the frequency. Because higher frequencies have less diffraction, the head shadows a sound more in the higher frequencies. Therefore, the IID is higher for higher frequencies for a sound source placed at an angle to the listener. The inter-aural time difference (ITD) is the time delay between the two channels. It also depends on the frequency: below 1 kHz, the difference is greater than above, but constant otherwise. Of course, IID and ITD both depend on the angle of the sound source.

IID and ITD, together with shoulder echoes and pinna reflections, are represented by a **head related transfer function** (**HRTF**). It can be measured by artificial dummy heads for sound sources at different angles and approximated by IIR

filters of an order of about 10. A different approach is to approximate the head by a sphere and calculate the IID filter as a first-order IIR filter. ITD is then implemented by a delay; shoulder echoes by a single echo, also with angle-dependent delay; and the pinna reflections is a short series of short-time echoes with angle-dependent delays of only a few samples implemented by an interpolated tapped delay line.

When listening to sound via headphones, the sound usually seems to come from inside of the head. Special measures have to be taken to achieve **sound externalization**, i.e. to push the apparent sound source out of the head. The best method to do so is **decorrelation**. The correlation of two sound signals is given by

$$r(\tau) = \frac{\int x_L(t) x_R(t+\tau) \mathrm{d}t}{\sqrt{\int x_L^2(t) \mathrm{d}t \int x_R^2(t) \mathrm{d}t}},$$

which has a strong peak for lags τ if one signal is just a copy of the other shifted by τ. Decorrelation can be achieved by complex reverberation or convolution with uncorrelated white noise.

Additionally, the correlation determines the apparent source width of the signal. This is important for stereo loudspeakers (not only headphones). A positively correlated signal seems to be a point-like sound source close to the listener. A non-correlated signal seems to be a wide sound source at half distance to the loudspeakers. A negatively correlated signal seems to be a point-like sound source between the loudspeakers.

A general approach to capture 3D audio is sound field recording. Several microphones are placed in several directions, and the sound is reproduced by loudspeakers placed in the same directions. The disadvantage is that the recording forces a fixed placement of loudspeakers, as in surround sound. A better approach is **Ambisonics**. It represents the sound field by a non-directional sound pressure component W and three directional components X, Y, and Z. So, if the sound can arrive from all directions, front, back, left, right, up, down, then

$$\begin{aligned} W &= \text{front} + \text{back} + \text{left} + \text{right} + \text{up} + \text{down} \\ X &= \text{front} - \text{back} \\ Y &= \text{left} - \text{right} \\ Z &= \text{up} - \text{down} \end{aligned}$$

If a signal x arrives from a direction $\vec{u}$, then

$$(W, X, Y, Z) = (\sqrt{2}/2, \vec{u}) \cdot x\,.$$

For decoding, i.e. playback of the sound field, at least four loudspeakers are required. A loudspeaker at direction $\vec{u}$ shall produce a sound according to

$$\frac{1}{2}(G_1 W + G_2 (X, Y, Z)^\top \vec{u}),$$

where G_1 and G_2 depend on the theory (there are several) and might be frequency-dependent, i.e. they might be filters. A disadvantage of Ambisonics is that there will probably be so-called "sweet spots", i.e. places where the sound field is reproduced convincingly, whereas one step away the sound field reproduction might collapse or be inverted. If the elevation component of the sound field is not needed, the Z channel can be ignored. On the other hand, there are higher-order versions of Ambisonics, where the first-order spatial derivatives X, Y, Z of the sound field are supplemented by higher derivatives.

The apparent distance of a sound from the listener can be easily influenced by **reverberation**. There is always the direct sound that reaches the listener first, and one or more delayed copies of the sound at lower volumes. This simulates reflections from walls. The ratio of direct to reverberating sound is a cue for the distance of the sound source, because the reverberating sound basically does not become more silent when the sound source is more distant, it fills the room continuously. The direct sound, however, loses energy with the distance because the sound energy is spread on the surface of a growing sphere originating from the sound source. Thus, by making the direct sound more quiet and the reverberation louder, the sound source seems more distant.

A more elaborate method is to calculate the delays of the reverberations from reflections of virtual walls. If T_d is the time the direct sound takes from the sound source to the listener, and T_r is the time the sound takes to be reflected from a wall and reaching the listener in this way, then the delay $T_r - T_d$ will give a cue for the position of the sound source, especially when several reflections are calculated. Such delays can easily be implemented by a delay line.

If reverberated sound is reproduced by loudspeakers, the room that contains the listener and the loudspeakers will introduce reverberations itself, which cannot be avoided. A robust method to cope with this situation is the **room-within-a-room** model. The sound source is placed in a virtual room surrounding the actual listening room which is assumed to have holes in the wall at loudspeaker positions, permitting sound from the outer room to arrive in the listening room. The sound, as it appears at the loudspeakers, is delayed according to the path length l from the sound source in the outer room to the loudspeaker (hole) position by l/c, where c is the speed of sound. The paths may, of course, include reflections on the walls of the outer room. The gain is set to $1/l$, with l being mea-

sured in meters, according to the energy distribution on spherical sound waves. The gain might be limited to 1 to avoid infinite (or too high) gains. Also, speakers having opposite directions of the sound source might be attenuated, giving a better localization of the sound source.

The calculation of sound paths with reflections can become computationally demanding when multiple reflections are included until the attenuation by distance and damping at the walls lets the amplitude drop below a threshold. However, starting from a certain path length, the sound waves become planar and are somehow aligned with the room geometry. For simplicity, we assume a rectangular room of size (l_x, l_y, l_z). The reverberating sound waves then are grouped in **normal modes**, represented by a discrete vector (n_x, n_y, n_z) $(n_i = 0, 1, \ldots)$, meaning that there is a standing wave with a wavelength of

$$\lambda_n = 2\left(\left(\frac{n_x}{l_x}\right)^2 + \left(\frac{n_y}{l_y}\right)^2 + \left(\frac{n_y}{l_y}\right)^2\right)^{-\frac{1}{2}}.$$

For $n = (1,0,0)$ there is a standing wave with air-pressure oscillating between the left and the right wall, or a wave traveling from left to right and back, so there is a wavelength of two times the room size in x-direction. For $n = (1,1,0)$, opposing edges of the room have the same air-pressure, which oscillates between neighboring corners of the room. According to these wave lengths, the impulse response of the room will have resonances at frequencies $f_n = c/\lambda_n$. For irreducible triplets n, this frequency will be a fundamental frequency with multiples of the triplets generating harmonic frequencies. These parts of the reverberation can thus be implemented by comb filters with a feedback delay line.

Often, one does not want the artificial reverberation to introduce "coloration" of the sound, i.e. the magnitude response should be flat. Therefore, the simplest tool of recursive reverberation is the delay-based all-pass filter:

$$y[t] = (a * x) = cx[t] + x[t-m] - cy[t-m].$$

The idea is that, for later echoes, the signal is repeated with exponentially decreasing magnitude in constant intervals that depend on the modes of the room, where m represents the round-trip time of a mode, i.e. λ_n/c.

A popular combination of the techniques is **Moorer's reverberator**. The overall structure is a first part that represents the early reflections with a delay-based FIR filter

$$y_1[t] = x[t] + a_1 x[t-l_1] + \ldots + a_n x[t-l_n],$$

followed by comb filters with a low-pass filter in the loop

$$y[t] = (c * x)[t] = x[t] + g(l * y)[t-m]$$

applied in parallel to the output of the FIR part

$$y_2[t] = c_1 * y_1 + c_2 * y_1 + \ldots + c_o * y_1\,,$$

which is fed into an all-pass filter, delayed and mixed together:

$$y_3[t] = y_1[t] + (a * y_2)[t - l_n - k]\,.$$

The last delay guarantees that the late diffuse reverberation from the comb and all-pass filters appear after the last early reflection from the FIR part.

A generalization of the recursive comb filter $y[t] = x[t] + g \cdot y[t-m]$ is the **feedback delay network** (**FDN**). The feedback coefficient g is substituted by a matrix G so that

$$\vec{y}[t] = x[t - \vec{m}]\vec{b} + G\vec{y}[t - \vec{m}] \qquad \text{and} \qquad y[t] = dx[t] + \vec{c}^{\top}\vec{y}[t]\,,$$

where $\vec{y}[t - \vec{m}]$ means that each component of $\vec{y}$ is delayed by a different delay m_i. If G is a diagonal matrix, the result is just a set of parallel comb filters as in Moorer's reverberator. The non-diagonal elements of G account for the interaction between the room's normal modes due to diffusive elements such as pieces of furniture.

Taking the z-transform, we get

$$\vec{Y}(z) = \mathrm{diag}(z^{-\vec{m}})(\vec{b}X(z) + G\vec{Y}(z))\,,$$

$$(\mathrm{diag}(z^{\vec{m}}) - G)\vec{Y}(z) = \vec{b}X(z)\,,$$

$$H(z) = \frac{Y(z)}{X(z)} = d + \vec{c}^{\top}(\mathrm{diag}(z^{\vec{m}}) - G)^{-1}\vec{b}\,.$$

Now the poles and zeros of this expression can be analyzed. The poles are found by the solution of $\det(\mathrm{diag}(z^{\vec{m}}) - G) = 0$. They should be kept inside the unit circle to achieve a stable system. Moreover, they should all have the same absolute value, so that all modes will decay at the same rate, so no "coloration" will happen. In order to achieve this, first a lossless prototype is found with the poles all on the unit circle, for instance by choosing G as a unitary matrix. Then, attenuation coefficients α^{m_i} are introduced in the feedback loops. Finally, one *does* want to make higher frequencies decay faster, so the attenuation coefficients are substituted by lowpass filters.

Feedback matrices should be of a special form so they can be implemented in a fast way. One possibility are circular Toeplitz matrices, which represent a circular convolution and can therefore be implemented by Fourier methods in $n \log n$ complexity.

Real room reverberations can be implemented by first measuring the **room impulse response** and then convolve the input signal with the impulse response. For the first step, one could emit an impulse from a loudspeaker at the sound source position and record the resulting sound at the listener position. The problem with this approach is that loud impulses are bad for the loudspeaker, whereas quiet impulses do not produce enough sound to record reliably. The reason is that an impulse is the signal with the highest possible **crest factor**

$$C = \frac{\text{peak}|x|}{\text{RMS}(x)}.$$

The ultimate solution to this problem are **maximum length sequences** (**MLS**). They are pseudo-random binary (bit) sequences, generated by linear feedback shift registers. An example with a shift register size of 4 (a_3, a_2, a_1, a_0) is

$$a_3[t] = a_0[t-1] \text{ XOR } a_1[t-1], \qquad a_k[t] = a_{k+1}[t-1] \quad \text{for} \quad k = 0,1,2.$$

For initial values 0001 for a, the result is

$$a_0[t] = 100010011010111\,100010011010111\,100010011010111\,\ldots$$

Such sequences have several nice properties. First, for a shift register of size m, the sequence will have a length of $2^m - 1$ (15 in our case) before it repeats itself. Half of the runs are of length 1, a quarter of length 2, an eighth of length 3, and so on. Approximately half of the bits are 1. If 0 is substituted by -1, the resulting signal has a crest factor of 1, the minimum possible.

And it has a correlation property stating that the auto-correlation function (when we have -1 instead of 0) is very close to a train of impulses at intervals of $2^m - 1$, i.e.

$$(a \star a)[k] = \sum_{t=0}^{2^m-2} a[t]\,a[t-k] \approx \begin{cases} 2^{m-1} & k = 0 \mod 2^{m-1} \\ 0 & \text{else.} \end{cases}$$

So, apart from the repetition at (long) intervals and the scaling, the result is approximately the Kronecker delta δ. This can be used to extract the room impulse from the recording of a MLS emitted from a loudspeaker in the room in the following way. If h is the impulse response, then the recording will be $y = h * a$. Now, if we correlate y with a again, we get

$$y \star a = h * a \star a = h * \delta = h.$$

The direct convolution of the impulse response with an input signal in the time domain is computationally very costly. Because of the convolution theorem, however, there are faster methods using the FFT. The signal is split into blocks of size n, transformed and multiplied with the transformed impulse response of size m, then transformed back. The resulting convolution is circular, though. To avoid the circular overlaps, both the block and the impulse response have to be zero-padded to length $n+m$. The result has to be overlap-added. Another possibility would be to let input blocks of size $n+m$ overlap and discard m samples of the result. The big problem of these methods is the latency introduced by the block size. The latter approach can, however, be modified to achieve almost zero latency with a bit more computation. The idea is to split the impulse response h into blocks of increasing power-of-two size, so that short term responses are calculated with short filters and later responses with larger filters. All these approaches are perfectly described at this web page:

http://www.music.miami.edu/programs/mue/Research/jvandekieft/jvchapter2.htm

6 Audio Coding

We will first look at **lossless audio coding**. The simplest approach is **silence compression**, where runs of zero values are encoding with runlength-coding, i.e. a single codeword followed by the coding of the length of the zero-run. Almost silent parts could be set to zero, making this approach a lossy coding scheme.

All state-of-the-art lossless audio codecs use linear prediction (linear predictive coding), where a filter with optimized coefficients (Levinson-Durbin recursion) is used to predict the samples. In lossless coding, the prediction error has to be encoded. The differences between prediction and real samples follow a two-sided geometrical distribution:

$$P(x[t] - (p * x)[t] = k) \propto q^{-|k|}.$$

Such values can be efficiently encoded with **Rice codes**, or Golomb-Rice codes. To encode k, an optimal parameter M for the variance of the distribution has to be found, where M is a power of two for Rice codes. k is divided by M to give a quotient q and a remainder r:

$$k = Mq + r\,.$$

Then q is encoded as a unary code, i.e. as q ones followed by a zero, and r is

simply encoded as $\log_2(M)$ bits. For $M = 4$ we get:

k	code	k	code	k	code	k	code
0	000	4	1000	8	11000	12	111000
1	001	5	1001	9	11001	13	111001
2	010	6	1010	10	11010	14	111010
3	011	7	1011	11	11011	15	111011

This is only suitable for positive k, though. To allow for signed k, positive numbers can be mapped by $k \mapsto 2k$ for $k \geq 0$, and $k \mapsto 2|k| - 1$ for $k < 0$.

In **forward-adaptive prediction**, the optimal coefficients are calculated at the encoder side, quantized, and encoded so that the decoder can used the same coefficients to get the same prediction values, and, by adding the encoded prediction error, the correct reconstructed samples. In **backward-adaptive prediction**, the optimal coefficients are calculated from a previously coded block of the signal. As this can also be done at the decoder side, the coefficients do not have to be encoded. Although coefficients are not directly optimized for the block to be encoded, it allows for longer filters and non-quantized coefficients.

In **long-term prediction**, the signal is not predicted from directly preceding signal values, but from values further in the past, i.e. delayed by some τ. One to five values around $t - \tau$ are chosen for prediction. This method is very efficient on periodic signals. τ has to be found as the optimal period of the signal and encoded. Short-term and long-term prediction can be combined.

Standards and software such as FLAC (Free Lossless Audio Codec) and MPEG-ALS uses these approaches with many optimization details.

For **lossy audio coding**, early approaches include μ-law and A-law encoding, which is just logarithmic quantization in order to decrease the quantization step size for low-magnitude signals. Then, linear prediction can be used while quantizing the prediction errors in order to reduce the required number of bits. In DPCM (differential pulse code modulation) and ADPCM (adaptive DPCM), only quantized prediction errors are encoded; in pure linear predictive coding, only the prediction filter coefficients are encoded; and in CELP (code excited linear predictor), both are encoded.

More advanced lossy audio codecs operate in the frequency domain, though (transform coding). The problem here is that, if the transform is executed on blocks, then the block borders will introduce high frequency artifacts, which leads to bad compression performance. On the other hand, when windowed overlapping blocks are used, then the number of Fourier coefficients is increased by the overlap factor, which contradicts the purpose of data compression. There are

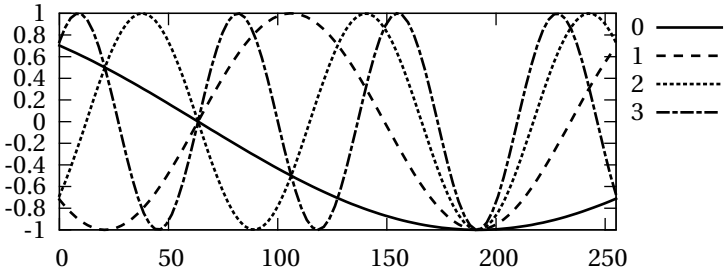


Figure 11: First four basis functions of the MDCT for $n = 128$.

two methods to cope with this problem: first, filter banks can be used instead of blocked transforms. This approach is taken in MPEG audio level 1–2.

Another possibility is the **modified discrete cosine transform** (**MDCT**). It is defined as

$$X[w,t] = \sum_{s=0}^{2n-1} x[nt+s] \cos\left(\frac{\pi}{n}\left(s+\frac{1}{2}+\frac{n}{2}\right)\left(w+\frac{1}{2}\right)\right),$$

where n is the hop-size, $2n$ the block size, and $w = 0,\ldots,n-1$. This means that each block of size $2n$ produces only n MDCT coefficients. However, there is a 50% overlap of the blocks, and this overlap is orthogonal, so no information is lost in the transform. Figure 11 shows the first four basis functions of the MDCT. The MDCT blocks can be windowed, where the window functions have to satisfy $w[s]^2 + w[s+n]^2 = 1$. The MDCT is used in MPEG audio layer 3 (MP3, in addition to filter banks), in MPEG-AAC (advanced audio coding), and in Vorbis.

Once audio data is represented in the transform domain, it can be quantized and encoded with entropy coders such as Huffman or arithmetic coding. There is, however, a huge gain in efficiency achievable with adaptively choosing quantization factors on a coefficient basis. The key for this is **psychoacoustics**.

The first effect in psychoacoustics is frequency masking, as shown in Figure 12. Apart from the general frequency-dependent audibility threshold, there is the following effect: Sinusoids with a frequency near a sinusoid with higher amplitude are masked out, i.e. the human ear or brain does not notice them. Therefore, coefficients below or near a masking threshold can be dropped or at least quantized more aggressively.

The second effect is an extension of frequency masking: temporal masking, as shown in Figure 13. The masking of a sinusoid continues for a short while after the masking sinusoid has already disappeared. All this masking information has to be calculated in the frequency domain and used in a quantization adaptation

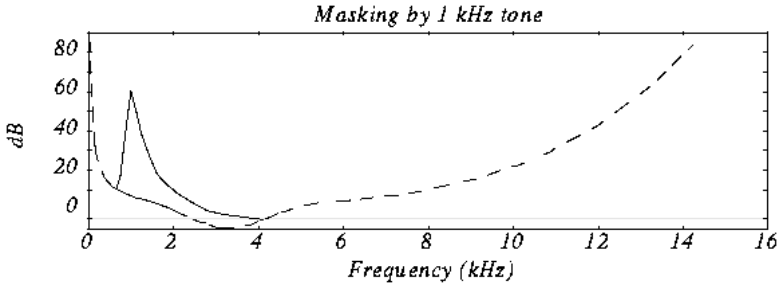


Figure 12: Frequency masking

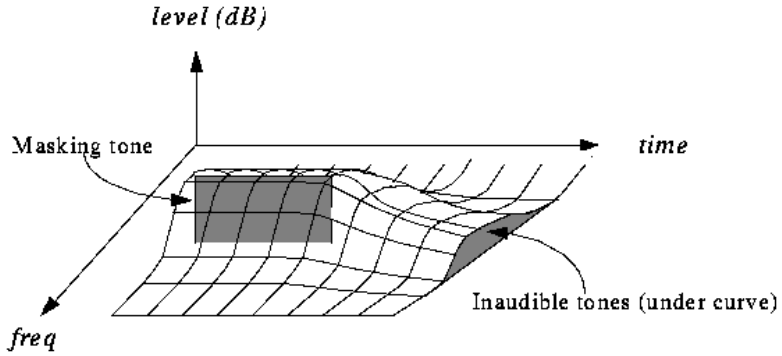


Figure 13: Temporal masking

loop with a target bit-rate as a goal. Psychoacoustics is used in all state-of-the-art lossy audio codecs, such as MP3, AAC, Vorbis.

Disadvantages of these codecs are (1) latency, i.e. the delay introduced by blocked processing, which makes them unusable for interactive audio, (2) bad compression performance for very low bit-rate and speech coding, where predictive techniques still have advantages, and (3) heavily patent covered techniques. The Opus codec aims to solve all these problems. It uses frequency-domain techniques for higher bit-rates, but can switch to predictive coding dynamically, whenever it achieves lower bit-rates. It also uses small block sizes and special techniques to overcome the resulting low frequency resolution.

Another improvement adopted recently by many audio codecs is **spectral band replication**. For low bit-rates, high frequencies are usually dropped entirely because they would require too many bits and are not necessary for "recognizable" audio quality. Spectral band replication aims to synthesize higher frequency bands by extrapolating frequency content in lower bands. Harmonic signals are supplemented with more harmonic frequencies in higher bands, and low-frequency noise with high-frequency noise. This post-processing step on the decoder side may be guided by low-bit-rate side information encoded by the encoder. The result is a signal that, although it only crudely approximates the original signal, sounds definitely "nice" and can improve comprehensibility of speech.

Index