APPLIED SIGNAL PROCESSING

2004

Chapter 1

Digital filtering

In this section digital filters are discussed, with a focus on IIR (Infinite Impulse Response) filters and their applications. The most important kinds of IIR filter prototypes are discussed in section 1.1, and structures used in the implementation of IIR filters are presented in section 1.2. In section 1.3 some important filters designed for special purposes are discussed. Finally, the application of digital filtering will be illustrated in section 1.4 by two case studies:

- Dual-tone multifrequency (DTMF) generation and detection in digital telephony, and
- Plucked-string filters in synthetic music.

1.1 IIR filter prototypes

An IIR filter of order N is described by the difference equation

$$y(n) + a_1 y(n-1) + \cdots + a_N y(n-N) = b_0 x(n) + b_1 x(n-1) + \cdots + b_M x(n-M) \tag{1.1}$$

and it has the transfer function

$$H(z) = \frac{b_0 z^N + b_1 z^{N-1} + \cdots + b_M z^{N-M}}{z^N + a_1 z^{N-1} + \cdots + a_N} \tag{1.2}$$

IIR filters provide more flexibility than FIR filters due to the feedback from previous outputs. Compared to FIR filters, the denominator polynomial of $H(z)$ provides more freedom to shape the frequency response of the filter. Therefore an IIR filter which achieves a given set of specifications is usually simpler and has much fewer parameters than a FIR filters designed for the same specifications. This property makes IIR filters well suited for a number of applications.

On the other hand, IIR filters have some limitations which one should be aware of. Firstly, high-order IIR filters are very sensitive to quantization errors in the coefficients,

which is due to the feedback from previous filter outputs $y(n-k)$. Quantization effects can therefore severely degrade the filter performance, or even make it unstable. In practice this problem can, however, be overcome by using filter implementations which are less sensitive to rounding errors. Secondly, in contrast to FIR filters it is not possible to achieve exact linear phase shift with IIR filters. IIR filters can, however, be designed to achieve an approximately linear phase shift in a frequency band. Morevover, in off-line (i.e., not real-time) computations, we will see that the filtering of a signal can always be arranged in a way which guarantees zero phase shift.

Standard IIR lowpass, highpass, bandpass and bandstop filters are usually designed using a set of standard IIR filter prototypes. These have originally been obtained from classical analog filter prototypes using the bilinear transformation, which takes a continuous-time filter to a discrete-time filter in a way which preserves the general form (lowpass, highpass etc) of the frequency response.

The digital prototype filters can, however, also be derived without referring to their analog counterparts. In order to construct digital IIR filters we need a building block whose frequency response magnitude varies with frequency. The transform variable z^{-1} is not suitable, as its frequency response $e^{-j\omega}$ has a constant magnitude for all frequencies. It turns out that a very useful building block is given by the function

$$F(z) = \frac{1 - z^{-1}}{1 + z^{-1}} = \frac{z - 1}{z + 1} \tag{1.3}$$

which has the frequency response

$$F(e^{j\omega}) = \frac{e^{j\omega} - 1}{e^{j\omega} + 1} = \frac{e^{j\omega/2} - e^{-j\omega/2}}{e^{j\omega/2} + e^{-j\omega/2}} = j \tan(\omega/2) \tag{1.4}$$

Hence the frequency response of $F(z)$ has an amplitude which grows from zero at $\omega = 0$ to infinity at $\omega = \pi$, and the constant phase $\pi/2$ (90°). This can be exploited as follows. Define the filter

$$H(z) = \frac{1}{1 + F(z)/\tan(\omega_c/2)} \tag{1.5}$$

Using (1.4), its frequency response is given by

$$\begin{aligned} |H(e^{j\omega})|^2 &= H(e^{j\omega})H(e^{-j\omega}) \\ &= \left(\frac{1}{1 + j\tan(\omega/2)/\tan(\omega_c/2)}\right)\left(\frac{1}{1 - j\tan(\omega/2)/\tan(\omega_c/2)}\right) \\ &= \frac{1}{1 + [\tan(\omega/2)/\tan(\omega_c/2)]^2} \end{aligned} \tag{1.6}$$

Hence $H(z)$ is a simple lowpass filter with $H(e^{j\omega}) = 1$ for $\omega = 0$, $H(e^{j\omega}) = 0$ for $\omega = \pi$, and the cutoff frequency ω_c. All the IIR filter prototypes discussed below are obtained by various generalizations of this idea.

The most important digital IIR filter prototypes are the *Butterworth*, *Chebyshev* and *elliptic* filters. They are characterized by the form of their magnitude frequency

responses. Butterworth filters have a monotonically varying response in both passband and stopband. Chebyshev filters of type I have an equiripple response in the passband and a monotonic response in the stopband, whereas type II Chebyshev filters have a monotonic response in the passband and equiripple response in the stopband. Elliptic filters have an equiripple response in both bands. The filters are usually derived as lowpass filters, from which the other filter types (highpass, bandpass and bandstop) are obtained by frequency transformations (see below).

The standard filter types are all based on the building block (1.3). Thus, (1.5) is a Butterworth filter of order one with the frequency response defined by (1.6). By replacing the denominator of (1.6) by a Chebyshev polynomial with argument $\tan(\omega/2)/\tan(\omega_c/2)$ gives a Chebyshev filter, while using an elliptic function results in an elliptic filter.

Butterworth filters.
A digital low-pass Butterworth filter $B_N(z)$ is a generalization of (1.5) and is constructed in such a way that the magnitude of its frequency response (gain) is

$$|B_N(e^{j\omega})| = \left[\frac{1}{1+[\tan(\omega/2)/\tan(\omega_c/2)]^{2N}}\right]^{1/2} \tag{1.7}$$

where N is the filter order and ω_c is the cutoff frequency. We can see that the frequency response magnitude is monotonically decreasing, with $B_N(e^{j\omega}) = 1$ at $\omega = 0$ and $B_N(e^{j\omega}) = 0$ at the Nyquist frequency $\omega = \pi$. At the cutoff frequency ω_c, the gain is $|B_N(e^{j\omega_c})| = 1/\sqrt{2}$. Increasing filter order N results in a steeper transition from passband to stopband.

Digital Butterworth filters can be synthesized with the Matlab routine

```
[B,A]=butter(N,Wn,'ftype')
```

where `Wn` denotes the cutoff frequency or, for a bandpass filter, information about the passband. The frequencies are normalized to the interval $0 <$ `Wn` < 1, where `Wn` $= 1$ corresponds to half the sampling frequency, i.e., π radians per sample. For a lowpass or bandpass filter `ftype` is omitted, whereas `ftype=high` or `stop` is used if a highpass or bandstop filter is designed. The outputs `B` and `A` are vectors with the coefficients of the numerator and denominator polynomials, respectively (cf. equation (1.2). The required minimum order of a digital Butterworth filter for a given set of specifications can be determined with

```
[N,Wn]=buttord(Wp,Ws,Rp,Rs).
```

where `Wp` and `Ws` are the normalized passband and stopband edge frequencies, `Rp` is the maximum deviation in the passband (in dB), and `Rs` is the minimum attenuation in the stopband (in dB).

Chebyshev filters.
There are two types of Chebyshev filters. The gain of a Chebyshev filter of type I is equiripple in the passband and monotonically decreasing in the stopband, whereas

a Chebyshev filter of type II has a monotonically varying gain in the passband and is equiripple in the stopband. Chebyshev filters involve the Chebyshev polynomials $C_N[x]$. A lowpass Chebyshev filter $H_N(z)$ of type I is defined in such a way that it has the frequency response magnitude

$$|H_N(e^{j\omega})| = \left[\frac{1}{1+\epsilon^2 C_N[\tan(\omega/2)/\tan(\omega_c/2)]^2}\right]^{1/2} \tag{1.8}$$

where N is the filter order, ω_c is the cutoff frequency, and ϵ is a parameter which affects the passband ripples.

Digital Chebyshev filters can be synthesized with the Matlab routines

`[B,A]=cheby1(N,Rp,Wn,'ftype')` and `[B,A]=cheby2(N,Rs,Wn,'ftype')`,

respectively. The minimum order of digital Chebyshev filters required to achieve a given set of specifications can be determined with

`[N,Wn]=cheb1ord(Wp,Ws,Rp,Rs)` or `[N,Wn]=cheb2ord(Wp,Ws,Rp,Rs)`.

Elliptic filters.
Elliptic filters have equiripple frequency responses in both the passband and the stopband. The frequency response magnitude of an elliptic filter of order N is given by

$$|H_N(e^{j\omega})| = \left[\frac{1}{1+\epsilon^2 G_N[\tan(\omega/2)/\tan(\omega_c/2)]^2}\right]^{1/2} \tag{1.9}$$

where $G_N[x]$ is a rational elliptic function of order N, ω_c is the cutoff frequency, and ϵ is a parameter which affects the passband ripples.

Digital elliptic filters can be synthesized with the Matlab routine

`[B,A]=ellip(N,Rp,Rs,Wn,'ftype')`

The required minimum order of a digital elliptic filter for a given set of specifications can be determined with

`[N,Wn]=ellipord(Wp,Ws,Rp,Rs)`.

Elliptic filters have the important property that they achieve a given set of specifications on the frequency response magnitude with the lowest filter order. This can be compared to the optimal FIR filters, which also have equiripple frequency responses. Elliptic filters are therefore the preferred filter type in IIR filter design. However, elliptic filters have a less linear phase response in the passband than the Butterworth or Chebyshev filters. In applications where the phase response is important these filter types should be considered.

Remark 1.1

Observe that all the filters described above have their classical, analogue counterparts. The analog filters are formed in an analogous manner using the building block $F_a(s) = s$ instead of (1.3). Note that $F_a(s)$ has the frequency response $F_a(j\omega) = j\omega$. In analogy with (1.3) the frequency response of $F_a(s)$ has an amplitude which grows from zero at $\omega = 0$ to infinity at infinite ω, and the constant phase $\pi/2$ (90°). From their construction it follows that the digital filters may be obtained from their analogue counterparts by making the substitution

$$s = \frac{z-1}{z+1} \tag{1.10}$$

in the transfer function of the analog filter. The substitution (1.10) is known as the *bilinear* transformation. Historically, the standard digital filter types were first constructed by applying bilinear transformation to the corresponding analog filters, and in textbooks they are usually still introduced in this way. This approach is, however, somewhat unsatisfactory, since it suggests that the digital filters are simply some form of discrete approximation of classical analogue filter types. As we have seen, however, the digital filters can be derived from scratch without a need to refer to analog filters.

Digital highpass, bandpass and bandstop filters.

The lowpass filter prototypes described above can be used to construct other standard filter types by observing that the only differences between lowpass, highpass, bandpass and bandstop filters consist of the locations of the passbands and stopbands. We can therefore transform a lowpass filter to any other standard filter type by introducing a frequency transformation, which takes the passband $-\omega_c \leq \omega \leq \omega_c$ to another location $-\omega_{min} \leq \omega \leq \omega_{max}$ in $[-\pi, \pi]$.

For example, a lowpass filter $H_{LP}(z)$ can be transformed to a highpass filter by defining

$$H_{HP}(z) = H_{LP}(-z) \tag{1.11}$$

Using the fact that $e^{-j\pi} = -1$ we then have

$$H_{HP}(e^{j\omega}) = H_{LP}(-e^{j\omega}) = H_{LP}(e^{j(\omega-\pi)}) = H_{LP}(e^{j(\pi-\omega)})^* \tag{1.12}$$

Hence $H_{HP}(z)$ is a highpass filter with the frequency response magnitude of $H_{LP}(z)$ shifted to the high frequency range, and the frequency response $H_{HP}(e^{j\omega})$ is the mirror image of $H_{LP}(e^{j\omega})$ about $\omega = \pi/2$.

Other frequency transformations, which take a lowpass filter to a bandbass or bandstop filter, are more complicated, and we refer to the literature.

1.2 Realization of IIR filters

Consider the IIR filter in (1.1), which can be written as

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \tag{1.13}$$

In a *direct realization* of the filter, the output is calculated recursively according to (1.13). In practice the direct implementation is, however, applicable only to IIR filters of very low order. For higher order IIR filters, it can be shown that the filter response is extremely sensitive with respect to the coefficients a_k. This is due to the feedback from previous filter outputs. The direct implementation is therefore of very limited use, and is not recommended if $N > 4$.

The solution is to break the filter down into smaller sections, typically to second- and first-order systems. This is achieved by considering the transfer function $H(z)$ associated with (1.13),

$$H(z) = \frac{b_0 z^N + b_1 z^{N-1} + \cdots + b_M z^{N-M}}{z^N + a_1 z^{N-1} + \cdots + a_N} \tag{1.14}$$

A *cascade structure* is obtained by factoring $H(z)$ according to

$$H(z) = H_1(z) \cdot H_2(z) \cdots H_K(z) \tag{1.15}$$

where

$$H_k(z) = \frac{b_{k0} + b_{k1} z^{-1} + b_{k2} z^{-2}}{1 + a_{k1} z^{-1} + a_{k2} z^{-2}} \tag{1.16}$$

and K is the integer part of $N + 1/2$. The factorization (1.15) allows the realization of the filter as a series connection of second-order systems.

A *parallel structure* is obtained by decomposing $H(z)$ using partial fractions according to

$$H(z) = C + H_1(z) + H_2(z) + \cdots + H_K(z) \tag{1.17}$$

where

$$H_k(z) = \frac{b_{k0} + b_{k1} z^{-1}}{1 + a_{k1} z^{-1} + a_{k2} z^{-2}} \tag{1.18}$$

The factorization (1.17) allows the realization of the filter as a parallel connection of second-order systems.

The realizations (1.15) or (1.17) can be implemented on a digital signal processor using only standard second-order building blocks.

When computing IIR filters it is in practice advisable to compute the cascade or parallel filter realizations (1.15) or (1.17) directly, rather than the polynomial coefficients a_k and b_k. Matlab supports the computation of filter realizations using second-order sections (or `sos`). For example, a cascade realization of a lowpass Butterworth filter can be designed by the commands

```
[z,p,k]=butter(N,Wn)
sos=zp2sos(z,p,k)
```

The first command finds a zero-pole-gain filter representation, which is then converted to a cascade realization (1.15). The output `sos` is a K by 6 matrix containing the $K \times 6$ coefficients of (1.15).

1.3 Some special problems in digital filtering

1.3.1 Zero-phase filtering

In real-time applications one has to use causal filters, for which the output $y(n)$ is a function of previous inputs $x(n-k), k = 0, 1, 2, \ldots$ only. An unavoidable property of causal filters is that they introduce phase shift. The best one can then do is to design the filter in such a way that the phase shift depends linearly, or approximately linearly, on frequency in the passband, corresponding to an equal time delay affecting all frequency components.

If one is not restricted to causal filters it is, however, always possible to perform the filtering in such a way that a phase shift is avoided. This is the case when the calculations are performed off line. Also in real-time applications the signals can often be processed in batches of length N, and in such cases the batches do not require causal filtering.

In off-line computations it is straightforward to achieve zero-phase filtering (i.e., a filtering with zero phase shift) as follows. Recall that the frequency response of the filter $H(z)$ is given by $H(e^{j\omega})$, and the phase shift is $\varphi_H(\omega) = \arg(H(e^{j\omega}))$. This means that the phase of the frequency component ω is shifted by the angle $\varphi_H(\omega)$ when the filter is applied to the input forward in time, i.e.,

$$y(n) = h(0)x(n) + h(1)x(n-1) + \cdots + h(n)x(0), n = 0, 1, \ldots, N-1 \tag{1.19}$$

By symmetry, it follows that if the filter is applied in reverse time, i.e., according to

$$y_{rev}(n) = h(0)x(n) + h(1)x(n+1) + \cdots + h(N-1-n)x(N-1), n = N-1, N-2, \ldots, 0 \tag{1.20}$$

the phase of the frequency component ω is shifted by the angle $-\varphi_H(\omega)$, whereas the frequency response magnitude is the same as in the forward implementation. It follows that zero-phase filtering can be achieved by running the filter $H(z)$ twice: first in reverse time, followed by a second application in forward time. We obtain the following two-stage filtering procedure.

1. Given a signal sequence $x(n), n = 0, 1, \ldots, N-1$.
2. Perform reverse-time filtering according to (1.20) to produce the filtered sequence $y_{rev}(n), n = 0, 1, \ldots, N-1$.
3. Perform forward-time filtering of the signal sequence $\{y_{rev}(n)\}$,

$$y(n) = h(0)y_{rev}(n) + h(1)y_{rev}(n-1) + \cdots + h(n)y_{rev}(0), n = 0, 1, \ldots, N-1 \tag{1.21}$$

It can be shown that the filtered signal sequence $\{y(n)\}$ then has the Fourier transform

$$Y(e^{j\omega}) = H(e^{j\omega})H(e^{-j\omega})X(e^{j\omega}) = |H(e^{j\omega})|^2 X(e^{j\omega}) \tag{1.22}$$

i.e., there is zero phase shift and the frequency components of the input signal are simply multiplied by the real, positive, number $|H(e^{j\omega})|^2$.

Zero-phase digital filtering is implemented in the Matlab routine

```
y=filtfilt(B,A,x)
```

1.3.2 Digital resonators

In many applications it is of interest to generate a sinusoidal signal,

$$y_{sin}(n) = \sin(\omega_0 n). \tag{1.23}$$

A sinusoidal signal is needed for example in digital tone generation. It is not practical to use the definition (1.23) directly, as n may become very large and because the computation of the trigonometric function is time-consuming. Therefore, it is preferable to generate a sinusoidal signal by constructing a dynamic model for it by expressing $y_{sin}(n)$ in terms of previous values, $y_{sin}(n-1)$ and $y_{sin}(n-2)$. This can be achieved by using the trigonometric identities

$$\sin(\alpha+\beta) = \sin\alpha\ \cos\beta + \cos\alpha\ \sin\beta \tag{1.24}$$

$$\sin\alpha\ \cos\beta = \frac{1}{2}\sin(\alpha+\beta) + \frac{1}{2}\sin(\alpha-\beta) \tag{1.25}$$

Then we can write

$$\begin{aligned} \sin(\omega_0 n) &= \sin(\omega_0(n-1)+\omega_0) \\ &= \cos\omega_0\ \sin(\omega_0(n-1)) + \sin\omega_0\ \cos(\omega_0(n-1)) \qquad \text{[by (1.24)]} \end{aligned} \tag{1.26}$$

where, by (1.25),

$$\begin{aligned} \sin\omega_0\ \cos(\omega_0(n-1)) &= \frac{1}{2}\sin(\omega_0 n) + \frac{1}{2}\sin(-\omega_0(n-2)) \\ &= \frac{1}{2}\sin(\omega_0 n) - \frac{1}{2}\sin(\omega_0(n-2)) \end{aligned} \tag{1.27}$$

It follows that

$$\sin(\omega_0 n) = 2\cos\omega_0\ \sin(\omega_0(n-1)) - \sin(\omega_0(n-2)) \tag{1.28}$$

Hence the signal $y_{sin}(n) = \sin(\omega_0 n)$ satisfies the difference equation

$$y_{sin}(n) - 2\cos\omega_0\ y_{sin}(n-1) + y_{sin}(n-2) = 0 \tag{1.29}$$

In the same way it can be shown that the signal $\cos(\omega_0 n)$ satisfies the difference equation (1.29). It follows that the general sinusoidal signal

$$y_s(n) = a\sin(\omega_0 n) + b\cos(\omega_0 n) \tag{1.30}$$

can be generated by the difference equation

$$y(n) - 2\cos\omega_0\ y(n-1) + y(n-2) = x(n) \tag{1.31}$$

where the input $x(n)$ is used to set the initial signal values to satisfy (1.30), i.e., $y(0) = y_s(0)$ and $y(1) = y_s(1)$. This is achieved by starting (1.31) from zero initial state ($y(-1) = y(-2) = 0$), and setting $x(0) = y_s(0)$ and $x(1) = -y_s(-1)$.

The system (1.31) is a *digital resonator*, as it resembles the analog resonators used to generate sinusoidal signals. Notice that the factor $2\cos\omega_0$ can be computed in advance, and (1.31) requires only one multiplication and one addition per sample. This is significantly more efficient than using polynomial sine function approximation or a lookup table and interpolation.

The transfer function associated with the difference equation (1.31) is

$$\begin{aligned} H(z) &= \frac{1}{1 - 2\cos\omega_0\ z^{-1} + z^{-2}} = \frac{z^2}{z^2 - 2\cos\omega_0\ z + 1} \\ &= \frac{z^2}{(z - e^{-j\omega_0})(z - e^{+j\omega_0})} \end{aligned} \tag{1.32}$$

and the frequency response $H(e^{j\omega})$ is strongly concentrated to the frequency ω_0, and in fact infinite at this frequency.

By (1.32), the digital resonator has two poles (zeros of the denominator) on the unit circle, and is therefore on the stability boundary. It follows that errors, such as roundoff errors, are not damped out but accumulate over time. Therefore it is a good idea to recalibrate the output $y_s(n)$ from time to time, for example by using (1.30) at certain time instants. Efficient recalibration techniques of digital resonators can be found in the literature.

Difference equation (1.31) is a direct-form filter realization, cf. section 1.2. As mentioned in section 1.2, direct realizations are sensitive to roundoff errors. This is particularly prominent in the case of a digital resonator with poles on the stability boundary. Therefore filter realizations with better numerical behaviour have been developed. A realization of the digital resonator with good numerical properties is the "magic circle" algorithm, defined by the equations

$$\begin{aligned} x_1(n) &= x_1(n-1) + \epsilon x_2(n-1) \\ x_2(n) &= x_2(n-1) - \epsilon x_1(n) \end{aligned} \tag{1.33}$$

where

$$\epsilon = 2\sin(\omega_0/2) \tag{1.34}$$

In order to see that (1.33) generates a sinusoidal signal with frequency ω_0, we introduce the trigonometric identities

$$\cos(\alpha) - \cos(\beta) = 2\sin(\frac{1}{2}(\alpha+\beta))\sin(\frac{1}{2}(\beta-\alpha)) \tag{1.35}$$

$$\sin(\alpha) - \sin(\beta) = 2\cos(\frac{1}{2}(\alpha+\beta))\sin(\frac{1}{2}(\alpha-\beta)) \tag{1.36}$$

Hence

$$\begin{aligned}\cos(\omega_0 n+\phi) - \cos(\omega_0(n-1)+\phi) &= 2\sin(\frac{1}{2}\omega_0(n+n-1)+\phi)\sin(\frac{1}{2}(\omega_0(n-1-n)) \\ &= 2\sin(\omega_0(n-1/2)+\phi))\sin(-\omega_0/2) \\ &= -\epsilon\sin(\omega_0(n-1/2)+\phi))\end{aligned}$$

and

$$\begin{aligned}\sin(\omega_0(n-1/2)+\phi) \; - \; & \sin(\omega_0(n-1-1/2)+\phi) \\ = \; & 2\cos(\frac{1}{2}\omega_0(n-1/2+n-1-1/2) \\ & +\phi)\sin(\frac{1}{2}(\omega_0(n-1/2-n+1/2+1)) \\ = \; & 2\cos(\omega_0(n-1)+\phi)\sin(\omega_0/2) \\ = \; & \epsilon\cos(\omega_0(n-1)+\phi)\end{aligned}$$

The result follows by taking

$$x_1(n) = \sin(\omega_0(n-1/2)+\phi) \tag{1.37}$$

$$x_2(n) = \cos(\omega_0 n+\phi) \tag{1.38}$$

1.3.3 Goertzel filters

Consider the finite sequence

$$\{x(n)\} = \{x(0), x(1), x(2), \ldots, x(N-1)\} \tag{1.39}$$

Recall that the discrete Fourier-transform, given by

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \;\; k = 0, 1, \ldots, N-1 \tag{1.40}$$

gives the frequency components of $\{x(n)\}$ at the frequencies $2\pi k/N, k = 0, 1, \ldots, N-1$. The sequence $\{X(k)\}$ can be efficiently computed using FFT. Suppose, however, that we are interested in the frequency components of $\{x(n)\}$ at a few frequencies only, for example at a single frequency, $2\pi k_0/N$ say. Then it is more efficient to compute the required element $X(k_0)$ directly. This can be done efficiently by exploiting the fact

that each element $X(k)$ can be expressed as the output of a simple filter. This can be seen as follows. By the periodicity of the complex exponential, we have

$$e^{-j2\pi kn/N} = e^{j2\pi k - j2\pi kn/N} = e^{j2\pi k(N-n)/N} = \left(e^{j2\pi k/N}\right)^{N-n} \tag{1.41}$$

Defining the complex number $W_k = e^{j2\pi k/N}$, (1.40) can then be written

$$X(k) = \sum_{n=0}^{N-1} x(n) W_k^{N-n} = W_k x(N-1) + W_k^2 x(N-2) + \cdots + W_k^N x(0) \tag{1.42}$$

But here the sum is recognized as the output at time N of a linear system. More precisely, we observe that the output of the system described by the difference equation

$$y(n) = W_k y(n-1) + x(n) \tag{1.43}$$

is given by

$$y(n) = x(n) + W_k x(n-1) + W_k^2 x(n-2) + \cdots + W_k^n x(0) \tag{1.44}$$

Comparing (1.44) and (1.42) shows that the required transform is obtained as $X(k) = y(N)$, where y is the solution of the difference equation (1.43) with the input sequence $\{x(0), x(1), \ldots, x(N-1), 0\}$, and with the initial condition $y(-1) = 0$.

The complex arithmetic involved in (1.43) can be avoided by manipulating the transfer function associated with (1.43) as follows. We have

$$\begin{aligned} H(z) &= \frac{1}{1 - W_k z^{-1}} \\ &= \frac{1}{1 - W_k z^{-1}} \times \frac{1 - W_k^{-1} z^{-1}}{1 - W_k^{-1} z^{-1}} \\ &= \frac{1 - W_k^{-1} z^{-1}}{1 - (W_k + W_k^{-1}) z^{-1} + z^{-2}} \\ &= \frac{1 - W_k^{-1} z^{-1}}{1 - 2\cos(2\pi k/N) z^{-1} + z^{-2}} \end{aligned} \tag{1.45}$$

where we have used the fact that $W_k + W_k^{-1} = e^{j2\pi k/N} + e^{-j2\pi k/N} = 2\cos(2\pi k/N)$. It follows that the transform $X(k)$ is obtained as

$$X(k) = y_k(N) - W_k^{-1} y_k(N-1) \tag{1.46}$$

where $y_k(n)$ is given by the difference equation

$$y_k(n) - 2\cos(2\pi k/N) y_k(n-1) + y_k(n-2) = x(n), n = 0, 1, \ldots, N \tag{1.47}$$

with the input sequence $\{x(0), x(1), \ldots, x(N-1), 0\}$, and with the initial condition $y_k(-1) = y_k(-2) = 0$.

The filter (1.47) is known as the *Goertzel filter*. Note the obvious similarity between the Goertzel filter and the digital resonator in equation (1.31). Goertzel filters are for example used to detect DTMF tones in digital telephony.

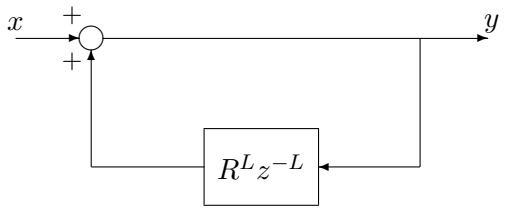

Figure 1.1: Block diagram of comb filter.

1.3.4 Comb filters

Consider the system

$$y(n) = R^L y(n-L) + x(n) \tag{1.48}$$

where $0 < R \leq 1$, and L is a positive integer. The associated transfer function is

$$H(z) = \frac{1}{1 - R^L z^{-L}} \tag{1.49}$$

An intuitive understanding for how the filter works can be obtained by considering the block diagram representation in Figure 1.1. The incoming signal is delayed L time steps, multiplied by R^L, and added to the input. It follows that the filter amplifies a periodic input signal with period L. Hence it also amplifies inputs with the periods which are integer fractions of L, i.e., $L/2, L/3, \ldots$. In other words, the filter amplifies the frequencies (in radians/second) $0, 2\pi/L, 2\pi \times 2/L, 2\pi \times 3/L, \ldots$. Recall that for a discrete-time signal, we need consider frequencies less than π (the Nyquist frequency) only. Hence we have that the comb filter (1.49) amplifies the frequencies

$$0, 2\pi\frac{1}{L}, 2\pi\frac{2}{L}, \ldots, \pi \quad \text{for even } L, \text{ and} \tag{1.50}$$

$$0, 2\pi\frac{1}{L}, 2\pi\frac{2}{L}, \ldots, \pi\frac{L-1}{L} \quad \text{for odd } L. \tag{1.51}$$

The resonance frequencies of the comb filter are reflected in its frequency response. Indeed, the poles z_k (denominator zeros) of the transfer function (1.49) satisfy $z_k^L = R^L$, i.e.

$$z_k = Re^{j2\pi k/L}, \;\; k = 0, 1, 2, \ldots, L-1 \tag{1.52}$$

It follows that when R is close to 1, the magnitude of the frequency response

$$H(e^{j\omega}) = \frac{1}{1 - R^L e^{-j\omega L}} \tag{1.53}$$

attains local maxima at the frequencies $\omega = 2\pi k/L, k = 0, 1, 2, \ldots, L-1$. For large L the magnitude of the frequency response therefore has many local maxima and minima and resembles of comb. See Figure 1.2. A filter with this property is called *comb filter*. The fact that comb filters have resonances at a set of harmonic frequencies makes them useful in a number of applications:

- The comb filter (1.48) can be used as a simple model of echo signals which are delayed by L time instants and are attenuated by the factor R^L before re-entering the loop.
- Starting with any initial sequence $y(n), n = 0, \ldots, L-1$ and a zero input, we have by (1.48) $y(n) = R^L y(n-L)$. Hence, if $R = 1$, the filter describes any periodic signal with period L. For $R < 1$, the signal magnitude is exponentially decreasing by the factor R^L during each period. This property can be used for modelling vibrating systems with an exponentially decaying vibration magnitude. Such a vibrating system contains precisely a set of harmonic frequencies $\omega = 2\pi k/L, k = 1, 2, \ldots$.

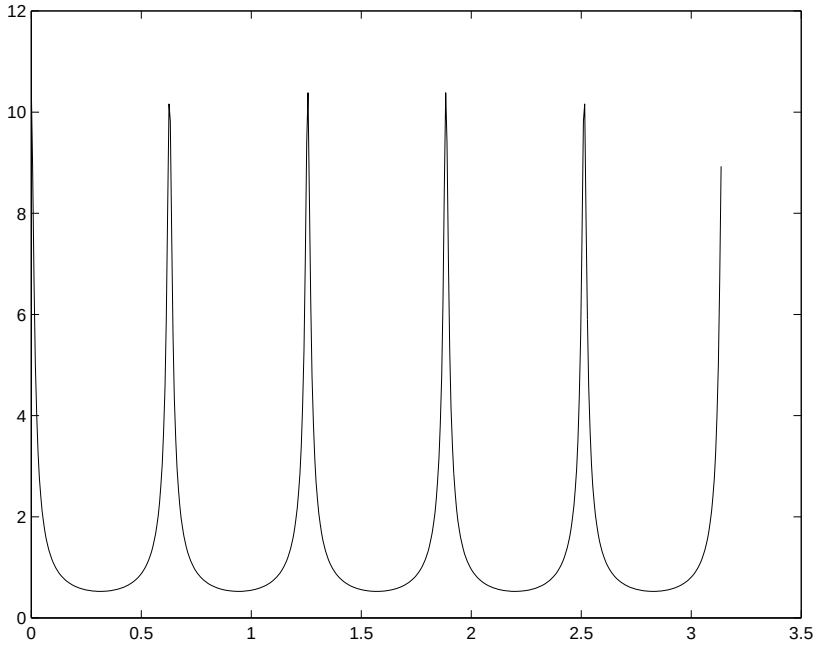

Figure 1.2: Frequency response magnitude of a comb filter with $R = 0.99$ and $L = 10$.

Comb filters with sign inversion.
A modification of the comb filter is obtained by reversing the sign of the delayed signal, i.e.,

$$y(n) = -R^L y(n-L) + x(n) \tag{1.54}$$

The additional sign reversion introduces an additional phase delay corresponding to half a period. Hence the filter (1.54) amplifies inputs with the periods $2L, 2L/3, 2L/5, \ldots$, i.e., it amplifies the frequencies

$$2\pi\frac{1/2}{L}, 2\pi\frac{1+1/2}{L}, 2\pi\frac{2+1/2}{L}, \ldots, \pi\frac{L-1}{L} \quad \text{for even } L, \text{ and} \tag{1.55}$$

$$2\pi\frac{1/2}{L}, 2\pi\frac{1+1/2}{L}, 2\pi\frac{2+1/2}{L}, \ldots, \pi \quad \text{for odd } L. \tag{1.56}$$

The resonant frequencies of the filter (1.54) correspond to the frequencies of a vibrating system which is free at one and and fixed at the other, such as vibrating air in a tube.

Inverse comb filters.
Another kind of comb filter (a.k.a. the *inverse comb filter*) is obtained by taking the inverse of (1.48) (with $R = 1$ and L replaced by M),

$$y(n) = x(n) - x(n - M) \tag{1.57}$$

with the associated transfer function

$$H(z) = 1 - z^{-M} \tag{1.58}$$

In contrast to (1.48), the filter (1.57) eliminates the frequency components $\omega = 2\pi k/M$, $k = 1, 2, \ldots$.

The filters (1.48) and (1.57) can be applied to extract or eliminate periodic components from a signal. Recall that every discrete-time periodic signal with period L contains the frequency components $\omega = 2\pi k/L, k = 0, 1, 2, \ldots$ only, and no other frequencies. It follows that the filter

$$H(z) = \frac{1 - z^{-M}}{1 - R^L z^{-L}} \tag{1.59}$$

eliminates a component with period M and lets a component with period L pass. This has been applied to separate periodic components caused by solar and lunar effects in astrophysical signals.

1.3.5 Allpass filters

An *allpass filter* $H_{AP}(z)$ is a filter whose frequency response has constant magnitude, i.e.,

$$|H_{AP}(e^{j\omega})| = K \tag{1.60}$$

In an allpass filter all frequencies are passed with equal weight, and only the phase is affected. Allpass filters can therefore be used in order to introduce a given amount of phase shift without affecting the relative magnitudes of the frequency components. For example, the resonant frequencies of the comb filters (1.48) are restricted to integer

multiples of frequencies which can be expressed as $2\pi/L$, where L is an integer. In order to fine-tune the comb filter to have resonant frequencies which are integer multiples of any fundamental frequency, which cannot be expressed as $\frac{2\pi}{L} \times k$, an allpass filter which provides the required additional phase shift without affecting the magnitude can be introduced into the loop.

In order to see how an allpass filter can be obtained, consider the general first-order IIR filter

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 + az^{-1}} \tag{1.61}$$

Its frequency response is given by

$$H(e^{j\omega}) = \frac{b_0 + b_1 e^{-j\omega}}{1 + ae^{-j\omega}} \tag{1.62}$$

Now consider how the allpass property $|H(e^{j\omega})| = K$, for all ω, can be achieved. As

$$|H(e^{j\omega})| = \frac{|b_0 + b_1 e^{-j\omega}|}{|1 + ae^{-j\omega}|} \tag{1.63}$$

one obvious way is to let

$$b_0 + b_1 e^{-j\omega} = K\Big(1 + ae^{-j\omega}\Big) \tag{1.64}$$

But this implies that $H(z) = K(1 + az^{-1})/(1 + az^{-1}) = K$, which is trivial. An alternative way to achieve an allpass is to require that the numerator is a constant times the complex conjugate of the denominator,

$$b_0 + b_1 e^{-j\omega} = K\Big(1 + ae^{-j\omega}\Big)^* = K\Big(1 + ae^{j\omega}\Big) \tag{1.65}$$

This is achieved if $H(z) = K(1 + az)/(1 + az^{-1})$, which is, however, a noncausal filter. Recalling that a time delay does not affect the frequency response magnitude, we can, however, achieve the allpass property with a causal filter by requiring that

$$b_0 + b_1 e^{-j\omega} = Ke^{-j\omega}\Big(1 + ae^{-j\omega}\Big)^* = K\Big(e^{-j\omega} + a\Big) \tag{1.66}$$

corresponding to the filter

$$H(z) = K\frac{a + z^{-1}}{1 + az^{-1}} \tag{1.67}$$

This is the general form of a causal first-order allpass filter.

Let's also determine the phase shift given by the filter (1.67). We have

$$\begin{aligned} H(e^{j\omega}) &= K\frac{a + e^{-j\omega}}{1 + ae^{-j\omega}} \\ &= K\frac{ae^{j\omega/2} + e^{-j\omega/2}}{e^{j\omega/2} + ae^{-j\omega/2}} \\ &= K\frac{ae^{j\omega/2} + e^{-j\omega/2}}{e^{j\omega/2} + ae^{-j\omega/2}} \times \frac{ae^{j\omega/2} + e^{-j\omega/2}}{ae^{j\omega/2} + e^{-j\omega/2}} \\ &= \frac{K}{1 + 2a\cos(\omega/2) + a^2} \times (ae^{j\omega/2} + e^{-j\omega/2})^2 \end{aligned} \tag{1.68}$$

Notice that the square of a complex number $z = re^{j\varphi}$ with the magnitude r and the argument φ is given by $z^2 = r^2 e^{j2\varphi}$, and therefore it has the magnitude r^2 and the argument 2φ. Hence the phase shift of $H(z)$, i.e., the argument of the complex number $H(e^{j\omega})$, is $2\times$(the argument of $ae^{j\omega/2} + e^{-j\omega/2}$). But

$$ae^{j\omega/2} + e^{-j\omega/2} = (a+1)\cos(\omega/2) + j(a-1)\sin(\omega/2) \tag{1.69}$$

It follows that the phase shift $\varphi_H(\omega) = \arg(H(e^{j\omega}))$ satisfies

$$\tan(\varphi_H(\omega)/2) = \frac{(a-1)\sin(\omega/2)}{(1+a)\cos(\omega/2)} = -\frac{1-a}{1+a}\tan(\omega/2) \tag{1.70}$$

or

$$\varphi_H(\omega) = -2\arctan\left(\frac{1-a}{1+a}\tan(\omega/2)\right) \tag{1.71}$$

Hence the phase shift is a nonlinear function of the frequency ω. However, using the fact that $\tan x \approx x$ for small x, we have that for small ω and $\varphi_H(\omega)$, (1.70) gives

$$\varphi_H(\omega) \approx -\delta\omega, \;\; \delta = \frac{1-a}{1+a} \tag{1.72}$$

Hence the phase shift is approximately linear for small ω, with the phase delay $\delta = (1-a)/(1+a)$. The first-order allpass filter is useful for achieving a desired phase delay δ without affecting the magnitude. Solving for a in terms of δ gives the filter parameter to be used to achieve the phase delay,

$$a = \frac{1-\delta}{1+\delta} \tag{1.73}$$

Figure 1.3 shows the phase shift of the first-order allpass filter (1.67) for some values of the filter parameter a.

The derivation of the first-order allpass filter can be generalized to higher-order filters. Hence a general Nth order allpass filter has the transfer function

$$H_{AP}(z) = \frac{a_N + a_{N-1}z^{-1} + \cdots + a_1 z^{-N+1} + z^{-N}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} \tag{1.74}$$

In most applications using allpass filters, the first-order filter (1.67) is, however, quite sufficient.

1.4 Digital filter case studies

1.4.1 DTMF tone generation and detection

One specific application of the digital resonator and the Goertzel algorithm is in generating and detecting dual-tone multifrequency (DTMF) tones in digital telephony. See section 8.19 in Ifeachor–Jervis for details.

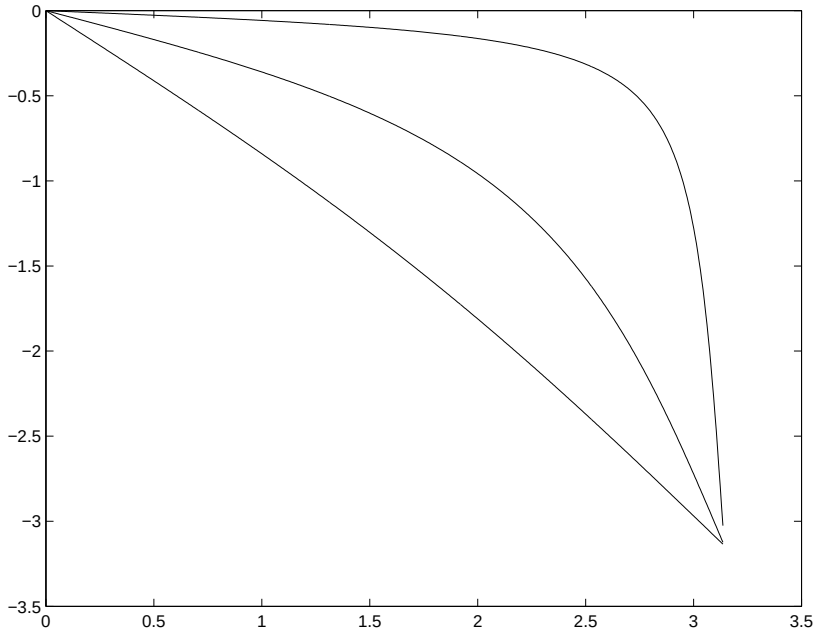

Figure 1.3: Phase shift of the first-order allpass filter (1.67) for $a = 0.1, 0.5$ and 0.9 (from below).

1.4.2 Plucked-string filters for synthetic music

Digital IIR filters are very useful for generating synthetic sound, such as synthetic music, speech signals, bird song etc. The idea is to construct a dynamic system which models the physical system generating the sound.

The simplest example of this kind is a simple vibrating system, such as a string or vibrating air in a tube. A vibrating string generates a periodic acoustic signal consisting of a fundamental frequency ω_0 and its harmonics $k\omega_0, k = 2, 3, \ldots$. We have seen in section 1.3.4 that such a signal can be described by a comb filter. In particular, the comb filter (1.48) with a specified initial sequence $y(n), n = 0, \ldots, L-1$ and zero input describes an exponentially decaying periodic signal with the period L, corresponding to the fundamental frequency $\omega_0 = 2\pi/L$.

The filter (1.48) does not, however, alone give a realistic model of the acoustic signal produced by a vibrating string. This is due to the fact that according to (1.48), all frequency components decay with the same exponental rate defined by R. In a real string, however, higher frequencies decay faster than lower ones. Hence the frequency contents of a vibrating string change with time, so that higher frequencies which are present in the beginning fade away whiler lower frequency components begin to dominate in the end. And that's a very important characteristic which makes the string sound as it does!

The filter can be equipped with the desired property by introducing a low-pass

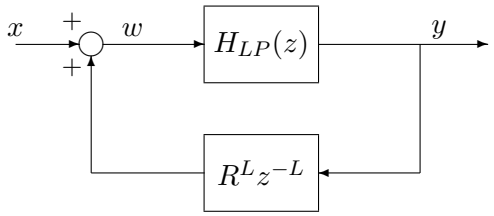

Figure 1.4: Block diagram of the plucked-string filter.

filter into the loop, cf. Figure 1.4. The effect of the low-pass filter is to attenuate higher frequencies, which therefore will decay faster than lower ones. It turns out that it is sufficient with a very simple low-pass filter, with a frequency response magnitude which is a monotonically decreasing function of the frequency. The standard method is to use the simple FIR filter

$$H_{LP}(z) = \frac{1}{2}\left(1 + z^{-1}\right) \tag{1.75}$$

Introduction of the filter (1.75) has a significant effect on the output signal, which is now a reasonably realistic approximation of the acoustic signal produced by a vibrating string. The filter in Figure 1.4 is called *plucked-string filter* because it models the acoustic signal produced by a plucked string. The plucked-string filter is one of the most commonly used building blocks in computer music. Notice that the transfer function of the plucked-string filter is given by

$$H_{PS}(z) = \frac{(1 + z^{-1})/2}{1 - R^L(z^{-L} + z^{-L-1})/2} \tag{1.76}$$

Apart from affecting the magnitudes of the frequency components, the filter (1.75) also introduces an additional phase shift in the loop. The filter (1.75) has the frequency response

$$H_{LP}(e^{j\omega}) = \frac{1}{2}\left(1 + e^{-j\omega}\right) = e^{-j\omega/2}\cos(\omega/2) \tag{1.77}$$

Thus, the phase shift is linear, introducing a time delay equal to 1/2, i.e., one-half sample. The effective loop delay of the system in Figure 1.4 is therefore changed to $L + 1/2$ time instants, and the filter amplifies the frequencies $2\pi k/(L + 1/2), k = 0, 1, 2, \ldots$.

Tunable plucked-string filter.
As we have seen the plucked-string filter has the loop delay $L + 1/2$, which restricts

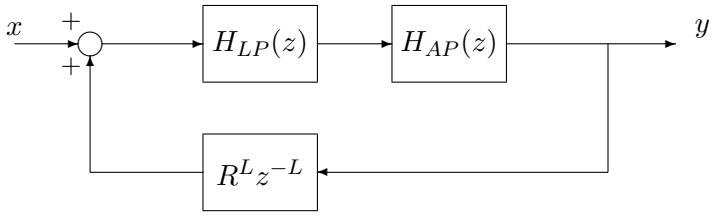

Figure 1.5: Block diagram of a tunable plucked-string filter.

the fundamental frequency to have the form $2\pi/(L + 1/2)$, where L is an integer. For realistic sampling frequencies, the frequencies which can be produced in this way are too far apart for the needs of synthetic music production. For example, at the frequency corresponding to $L = 50$, only the fundamental frequencies $2\pi/49.5, 2\pi/50.5, 2\pi/51.5$ can be constructed, corresponding approximately to a 2% resolution.

In order to construct arbitrary frequencies, we should introduce additional phase shift in the loop. A linear phase FIR filter, for example, is not useful, because such a filter will introduce a phase shift of the form $-n\omega$ or $-(n + 1/2)\omega$, and this we can already achieve with the delay L and the low-pass filter (1.75). What is needed is an additional phase shift of the form $-\delta\omega$, where δ can be any number between 0 and 1.

We observe that the first-order allpass filter introduced in section 1.3.5 has precisely the desired properties: its phase shift is approximately linear for small frequencies (cf. equation (1.72)), and its frequency response magnitude is constant. Therefore, the allpass filter can be designed in such a way that only the required phase shift is achieved, without affecting the attenuation of the various frequency components. This results in the *tunable plucked-string filter* shown in Figure 1.5. Here, $H_{AP}(z)$ is the first-order allpass filter (1.67), which should have the frequency response magnitude $K = 1$.

The tunable plucked-string filter in Figure 1.5 has the loop delay $L+1/2+\delta$, giving rise to the fundamental frequency $2\pi/(L + 1/2 + \delta)$. The filter therefore amplifies the frequencies $2\pi k/(L + 1/2 + \delta), k = 0, 1, 2, \ldots$. As the allpass filter can be designed to give δ any value between 0 and 1, it is possible to tune the filter for any fundamental frequency.

Finally, we may observe that the phase shift of the allpass filter is only approximately linear, and in reality higher frequencies will be delayed more than lower ones, cf. Figure 1.3. This does not seem to have any serious effect on the produced sound, however. The filter can be made still more realistic by adding nonlinear distortion. For a readable account of the plucked-string filter and references to the literature, we refer to K. Steiglitz: A Digital Signal Processing Primer (Addison-Wesley, 1995).