# Filter Design

Filters are the basis for many types of audio effects. In this chapter, we'll look at the basics of designing digital filters. We will show how many of the most important digital filters can be derived from the simplest possible designs. Several types of filters are commonly used in audio effects for manipulating the frequency content of the signal. These filters, in their ideal form, are depicted in Figure 3.1. They can be summarized as follows.

*Low pass, high pass and band pass filters*: the first class of filters aims to totally eliminate certain frequency ranges from the signal. The *low pass filter* passes low frequencies below some *cut-off frequency* *ωc* while eliminating all frequencies above the cut-off. The *high pass filter* is a mirror image of this. It passes all frequences above the cut-off frequency *ωc* and eliminates all frequencies below it. *Band pass filters* pass a range of frequencies. They are defined by the *center frequency ωc* and the *bandwidth B* which specify the location and width of the *passband*, respectively. Frequencies above and below this passband are blocked. The inverse of the band pass filter is the *band stop filter*, which blocks frequencies in its *stop band* while passing frequencies above and below.

*Shelving, peaking and notch filters*: this class of filters do not aim to entirely eliminate any frequencies, but rather aim to adjust the relative gain of specific portions of the frequency spectrum. *Shelving filters* come in two forms, *low shelving* and *high shelving* filters. They are analogous to low pass and high pass filters, but instead of eliminating high or low frequency, they provide a boost or cut on the shelf region and leave all other frequency content unaffected. *Peaking* and *notch filters* are analogous to band pass and band stop filters in that they affect only a narrow range of the spectrum. They apply a boost or cut to a specific frequency band while leaving the area around it unchanged. Like band pass and band stop filters, peaking and notch filters are often specified by a *center frequency* and *bandwidth*. The center frequency defines the location of maximum gain in a peaking filter, or minimum gain in a notch filter. The bandwidth defines the size of the region around the center frequency. The gain is also typically a parameter in peaking and notch filter design. These will be explained in detail later in the chapter.

In Chapter 4, we will see how these filters are used in audio effects like equalizers. But let us first study their general behavior. In practice, we cannot achieve the perfectly sharp cut-off between pass band and stop band shown in Figure 3.1, and in fact a perfect sharp cut-off would not be aesthetically desirable. Rather, in audio contexts, we usually desire a smooth transition between regions. As we will see, the *order* of a filter (a rough measure of its complexity) also determines how sharp the cut-off is between pass band and stop band. Audio effects most commonly use low order filters, which have the benefits of simpler design and less susceptibility to error.

In the following sections, we will begin with a very simple prototype filter and gradually generalize it to more complex designs. We will show that all the common classes of filter can be constructed by a series of operations on this simple prototype, and that we can alter the frequency spectrum of a signal almost arbitrarily just by applying combinations of such filters.

## Filter construction and transformation

In this section, we look at how a simple filter can be constructed, and how it can be extended or transformed into other designs. We keep most of the discussion regarding transformations quite general, so that they apply to high order as well as low order designs. There is a fair amount of maths in this section, and we don’t skip the details. However, readers can skip ahead to the next section on Popular IIR Filter Design, if just interested in how these transformations are used to generate the various filter designs from Figure 3.1.

### Simple, prototype low pass filter

Consider averaging every two consecutive samples. In the time domain, the output is given as;



This equation produces a low pass filter. To see why, let's consider its response at very low and very high frequencies. For a very low frequency input, the signal hardly changes from sample to sample. In fact, if the input signal has frequency 0, *x*[*n*]= *A*cos(2**0*n*/*fs*), then it is constant. This is known as DC input, since direct current electrical signals have this quality. For DC input, the output of this filter is identical to the input.

Now suppose we have a very high frequency signal, at half the sampling frequency. So *x*[*n*]=*A*cos(2**( *fs*/2*)n*/*fs*)= *A*cos(*n*). Thus, the signal switches sign from sample to sample, and the output of this filter is zero.

Let’s look at this filter in the frequency domain. Recall from Chapter 1 that we can express this in the *Z* domain as . Its transfer function, written in positive powers of *z*, is

,

and the square magnitude of this transfer function is

.

Hence this acts as a low pass filter, allowing low frequencies to pass through to the output, but removing high frequency content. You can easily see that for *f*=0, ***z*=1)=1 and for *f*= *fs*/2, *z*=-1)=0. But what happens halfway, at *f*= *fs*/4? Here, = 2*f*/*fs*= **/2 and we see that |***z=j*)|2= 1/2.

We call this the cut-off frequency of our low pass filter. Generally, the cut-off frequency is where the frequency response makes the transition between two values. There are various ways a cut-off frequency may be formally defined, but one of the most effective, and the one we will use in this chapter, is that frequency at which the square magnitude is halfway between its low and high value (in this case, halfway between 0 and 1).

### High order prototype low pass filter

Sometimes we want a sharp roll-off at the cut-off frequencies. Let’s return to our simple low pass filter. It has one zero at *z=*-1 and one pole at *z=*0. Replace the pole at 0 by *N* points given by:



where *n* assumes ranges over the values between –/2 and /2, . So our *N*th order prototype filter is



These filters have the same cut-off frequency and same behaviour at DC and Nyquist as the first order filter. But now the poles result in a very sharp transition at =/2. In fact, the square magnitude is now



The pole zero plot and square magnitude are shown in Figure 3.2 for a fourth order prototype low pass filter.

We now have a high order filter, composed of first order sections. In what follows, we will show how each of those first order sections may be transformed to create other types of high order filters.

### Changing the gain at the cut-off frequency

Consider another transformation, given by

.

This has a few important properties. Every value inside the unit circle is mapped to a value inside the unit circle, those outside are mapped to values outside the unit circle, and those on the unit circle stay on the unit circle. So we can view this transfer function as preserving stability. Furthermore, it maps *z*=1 to 1 and *z*=-1 to -1.

Suppose that at some frequency *G* we have

.

Then from

.

We now use *F* to map this frequency G to /2, so that the square magnitude of *H*(*F*(*z*)) at /2 is the same as the square magnitude of *H*(*z*) at G. That is, we need the transformation

.

From and we have,

.

Thus, to change the gain at the cut-off frequency /2, we replace each first order section,  from our prototype filter with the following;

.

### Shifting the cut-off frequency

So far, our cut-off frequency has been set to **/2. We would like to be able to set the cut-off frequency to any value between 0 and **. This would allow us to change the location that provides the transition between where frequencies are passed and blocked.

Consider another transformation, given by

,

where as before,

.

This transfer function will map some new cut-off frequency *c* to the cut-off frequency of our prototype filter, **/2, as shown in Figure 3.3.

By putting into and solving for ,

.

Let us consider any first order section (the transfer function can be written as first order polynomials), . To change the cut-off frequency, we replace *H*(*z*) with,

,

where ** is defined as in .

### Creating a shelving filter

We can construct a low shelving filter by transforming our prototype filter, such that the square magnitude response is transformed from *H*2 to (*G*2-1)*H*2­+1, depicted in Figure 3.4. This transformation changes the extreme square magnitudes 0 and 1 of a low pass design to 1 and *G*2.

Recall that the poles will push up the magnitude response for nearby frequencies, and the zeros will pull it down. For the low shelving filter, we want to keep the poles on the imaginary axis, giving a sharp cut-off frequency. But now we shift the zeros so that, for each first order section *H*(*z*=1)=*g*=*G*1/*N* and *H*(*z*=-1)=1. That is, if the first order section of a prototype low pass filter is written as



Then the first order section of the low shelving filter becomes,



So . The pole zero plot and square magnitude response for a fourth order low shelving filter are shown in Figure 3.5. The poles still produce a sharp transition at the cut-off frequency, and the filter reaches its minimum at **= But the zeros are all inside the unit circle, far from **=**, so that the square magnitude response does not drop to zero. What the pole zero plot does not show, however, is the constant *k*, which is used to normalize the filter so that the response ranges from *G* to 1.

### Inverting the magnitude response

For a high pass filter, we want |***z*=1)|2=0, |*z*=-1)|2=1 and |*z*=*c*)|2=G2. So we simply apply the transformation *FHP*(*z*)=-*z.* This transformation is shown in Figure 3.6.



### Simple low pass to band pass transformation

Now we want to create a band pass filter from our simple filter with centre frequency /2. We would like to transform the frequency range 0 to  to the frequency range – to If this transformation is applied to the input before a low pass filter is applied, then our low pass filter becomes a band pass filter, as shown in Figure 3.7. To do this, consider a transfer function *F*(*z*) with the following constraints;

.

*F* will move the lower and upper cut-off frequencies to ±/2, where our prototype low pass filter has its cut-off frequency, and it will move the centre frequency to 0, where our prototype low pass filter has gain equal to 1.

We can solve Eq. to arrive at

.

This transfer function has second order polynomials in the numerator and the denominator. So our first order section has now become a second order section.

## Popular IIR Filter Design

It is possible to design FIR (finite impulse response) filters, where the output is dependent only on current and previous inputs, and not on previous outputs, to design the filters whose ideal forms were depicted in Figure 3.1. However, FIR designs either give poor approximations to the ideal forms, or must be very high order and hence introduce significant delay and computational issues. Thus, preferred designs are usually IIR (infinite impulse response) filters, where output is fed back to the input. This section will use the transformations from the previous section to construct such filters.

We now have everything we need to construct many of the most popular audio filters. We first construct a high order prototype filter with prescribed gain at /2. Then we show how we can use the transformations just described to turn this prototype filter into the filters given in Figure 3.1. In each case, we’ll give an example of the lowest order where cut-off frequency or bandwidth is defined using Eq. .

### Low pass

As mentioned, a low pass filter has a transfer function *HLP* with magnitude 1 at frequency 0 and magnitude 0 at frequency **=** (*f*=*fs*/2). That is, the magnitude of the lowest frequencies are unaffected and the highest frequencies are eliminated. At some cut-off frequency *c*, it has magnitude *Gc*, and this represents the transition where frequencies below *c* are considered passed and above this are rejected.

In order to generate a high order low pass filter*, w*e first generate our high order prototype filter. Then by applying these transformations to each first order section in our high order prototype filter, we can generate a high order low pass filter.

We first use Eq. to transform each first order section in the *N*thorder prototype filter such that each first order section has magnitude *Gc*1/*N*, rather than (1/2)1/(2*N*), at **/2 (that is, the whole filter has square magnitude *Gc* 2, rather than 1/2 at **/2). Then we apply Eq. to shift the cut-off frequency so that each first order section has magnitude *Gc*1/*N* at *c*, rather than at /2*.*

Consider a simple first order case where we define the gain at the cut-off frequency such that the square magnitude is the average of the two extremes, *Gc* 2=(02+12)/2=1/2.

We start with

.

At **/2, this filter has square magnitude 1/2. That is, this definition of gain at the cut-off is the same definition used in the prototype filter. So there is no need to change the gain at the cut-off frequency.

Now we shift the cut-off frequency from **/2 to *c*,

.

This simplifies to,

.

Figure 3.8 shows a pole zero plot and square magnitude response of a 1st order and a 4th order low pass filter with cut-off frequency of **/4. In both cases, the zeros are at -1. Notice that, for the 4th order design, the poles are placed on the arc of a circle so as to pull down the magnitude sharply at *c*.

### High pass

A high pass filter has a transfer function *HHP* with magnitude 0 at frequency **0, magnitude 1 at frequency **=**, and magnitude *Gc* at some cut-off frequency *c*.

To generate a high pass filter with cut-off frequency c, we begin with the high order prototype filter. Then we transform each first order section, such that it has magnitude *Gc*1/*N* at **=/2. Then we invert the magnitude response of each of these sections, and then shift the cut-off frequency of each section from /2 to *c*.

Let us again consider the first order case where *G*2=1/2. There is no need to change the gain at the cut-off frequency, so we begin with Eq. . Then we invert the magnitude response using Eq. , giving

.

And now we shift the cut-off frequency,

,

which reduces to

.

Figure 3.9 shows a pole zero plot and square magnitude response of a 1st order and a 4th order low pass filter with cut-off frequency of **/4. These are identical to the low pass filters, except now the zeros have been moved from -1 to +1, giving maximal attenuation at the low frequencies.

### Low shelf

A low shelving filter has a transfer function *HLS* with magnitude *G* at frequency **0 (representing the low shelf), magnitude 1 at frequency **=**, and magnitude *Gc* at some cut-off frequency *c*.

As before, we start with our high order prototype filter. We will change the gain at the cut-off frequency to some value *g*. We will transform the range of square magnitudes of the prototype filter to the range of square magnitudes of the shelving filter. That is, we want the extreme square magnitudes, 0 and 1, of the low pass filter, 0 and 1, to map to the extreme square magnitudes of the low pass filter, 1 and *G* 2, with *g*2 mapping to *Gc* 2. Thus,

.

So for each first order section of an *N*thorder prototype filter,we apply Eq. and to change the gain at the cut-off frequency to this new value *g*1/*N.* Then we transform each section to a shelving filter using Eq. , and then use Eq. to shift the cut-off frequency to *c*.

Consider again our example first order filter with the gain at the cut-off frequency defined such that the square magnitude is the average of the two extremes *Gc*2=(*G*2+12)/2. Then Eq. simplifies to

,

and as with the low pass filter, this choice of cut-off frequency is the same as that used in the prototype filter.

Now we create the shelf,

.

And finally, we shift the cut-off frequency,

,

which reduces to



In Figure 3.10, we’ve given a pole zero plot and square magnitude response of a 1st order and a 4th order low shelving filter with cut-off frequency of **/4. By placing the poles slightly closer to 1 than the zeros, it has the effect of pushing up the low frequencies. If the poles and zeros were further apart, then the shelf would be higher (larger value of *G*). If the zeros were placed closer to 1 than the poles, then the shelf would be at a value *G*<1.

### High Shelf

A high shelving filter has a transfer function *HHS* with magnitude 1 at frequency **0, magnitude *G* at frequency **=**(representing the high shelf), and magnitude *Gc* at some cut-off frequency *c*.

As before, we will change the square magnitude at the cut-off frequency of each first order section of an *N*thorder prototype filter,from (1/2)1/*N* to *g*2/*N*, where *g*2 is defined as in . Then we transform each section to a shelving filter, then invert the magnitude response and then shift the cut-off frequency to *c*.

For our first order filter, the procedure is the same as with the low shelf, giving Eq. after we transform this to a shelving filter. Inverting the magnitude response then gives,



And after shifting the cut-off frequency, we arrive at,



First and fourth order high shelving filters are depicted in Figure 3.11. Compared with the low shelving filter, the pole and zero positions have been switched. Note that the pole zero plots are also equivalent to a low shelving filter with *G*<1, only the constant term would be different.

### Gain at bandwidth

The band pass, band stop, peaking and notch filters have additional parameters that relate to bandwidth. We previously specified that the centre frequency is where the filter reaches its maximum or minimum value, and cut-off frequency is where the square magnitude is half way between its two extremes, but what about bandwidth? Well, the gain at bandwidth is defined similar to the gain at the cut-off frequency. That is, *GB* is defined using the arithmetic mean of the extremes of the square magnitude response, *GB*2=(1+*G*2)/2. So



This is thus the simplest definition, and one we will use in examples that show how to generate simple low order filters. Note, however, that we will introduce an alternate definition later when we discuss parametric equalizers in Chapter 4.

### Band pass filters

For a band pass filter, *HBP*(=0)=0, *HBP*(=/2)=0, *HBP*(=*c*)=1, and for the upper and lower cut-off frequencies, |*HBP*(=*l*)|= *|HBP*(=*u*)|=*Gc*. Bandwidth is defined as *B=u*-*l*. So this filter is designed to only pass a range of frequencies around the cut-off frequency, and suppress all other content.

To design a high order band pass filter, we consider each first order section of our prototype filter. We change the gain at the cut-off frequency to *Gc* using Eq. , then shift the cut-off frequency to the bandwidth with Eq. , where *c* in Eq. is replaced with *B.* Finally we transform this to a band pass filter using Eq. .

For our first order filter with the gain at the cut-off frequencies defined such that the square magnitude is the average of the two extremes *Gc*2=(02+12)/2=1/2, again there is no need to change the gain at the cut-off frequency of the prototype low pass filter.

So we now shift the center frequency of the prototype filter to *B*.



Then transform to a band pass filter with bandwidth *B* and center frequency *c,* which gives

.

Band pass filters are shown in Figure 3.12. Note that these are twice the order of the previous examples (2nd and 8th, as opposed to 1st and 4th) since the band pass transformation doubles the order of the filter. Now there are zeros at both 1 and -1, but poles surround the center frequency, allowing for a sharp transition from attenuating to passing frequency content. If the bandwidth is increased, then the distance of the poles from the center frequency would also increase.

### Band stop filters

For a band stop filter, *HBP*(=0)=1, *HBP*(=/2)=1, *HBP*(=*c*)=0, and bandwidth is *B=u*-*l* where |*HBP*(=*l*)|= *|HBP*(=*u*)|=*Gc*. So this filter is designed to suppress a range of frequencies around the center frequency, and pass all other content.

To design a band stop filter, we again start with the prototype filter. For each first order section, we change the gain at the cut-off frequency. Then invert the magnitude response, then shift the cut-off frequency, then transform this to a band pass filter. This is similar to the design of a high order band pass filter, except that by inverting the magnitude response using Eq. , the resultant filter has band stop behavior.

For our first order filter with the gain at the upper and lower cut-off frequencies given as *Gc*2=1/2, there is no need to change the gain at the cut-off frequency. So, starting with the prototype, we invert the magnitude response, and then shift the cut-off frequency, as in the high pass filter, giving

.

Then transform to a band pass filter with bandwidth *B.*



The pole zero plots of the band stop filters in Figure 3.13 are very similar to the band pass filters in Figure 3.12. But now the zeros have been moved from +1 and -1 to cos*c*+*j*sin*c* and cos*c*-*j*sin*c*. This has the effect of the poles and zeros cancelling out far from the center frequency (leaving the magnitudes unaffected), and having the zeros dominate at the center frequency (attenuating the magnitude to zero)

### Peaking and notch filters

For a peaking or notch filter, *HPN*(=0)=1, *HPN*(=/2)=1, *HPN*(=*c*)=*G*, and for the upper and lower cut-off frequencies, |*HPN*(=*l*)|= *|HPN*(=*u*)|=*Gc* [13]. Bandwidth is defined as *B=u*-*l*. When *G* is greater than 1, this filter provides a boost around *c*and is known as a peakingfilter. When *G* is less than 1, this filter attenuates the frequency content near *c*and is known as a notchfilter. Clearly, if *G=*0, the filter completely removes frequency content near *c* and is a band stop filter.

To design a high order peaking or notch filter, we consider each first order section of our prototype filter. We use Eq. and to change the gain at the cut-off frequency. Next we transform this to a shelving filter using Eq. . Then we shift the cut-off frequency to the bandwidth with Eq. , where *c* in Eq. is replaced with *B.* Finally we transform this to a band pass filter using Eq. .

For our first order filter with the magnitude of the transfer function at bandwidth defined as we defined the magnitude at the cut-off frequency for the shelving filter, we can follow the same steps as were taken with the shelving filter to give Eq. , except now the center frequency is replaced by the bandwidth *B.*



Then transform to a band pass filter with bandwidth *B.*

.

The pole zero and square magnitude plots for first and fourth order designs are shown in Figure 3.14.

To summarise, when we use the simple definition of bandwidth or cut-off frequency, the standard filters mentioned above all have relatively straightforward forms for first order designs (where bandwidth is not specified), and second order designs (for band pass, band stop and peaking/notch filters). These are given in Table 1.

Table . Transfer functions of common first and second order filters, and their equivalent forms based on simpler filters.

|  |  |  |
| --- | --- | --- |
|  | Transfer function | Equivalence |
| 1st order low pass (*HLP*) |  |  |
| 1st order high pass (*HHP*) |  | 1-*HLP*(*z*) |
| 1st order low shelf (*HLS*) |  | *GHLP*(*z*) +*HHP*(*z*) |
| 1st order high shelf (*HHS*) |  | *HLP*(*z*) +*GHHP*(*z*) |
| 2nd order band pass (*HBS*) |  | - |
| 2nd order band stop (HBS) |  | 1-*HBP*(*z*) |
| Peaking or notch filter (*HPN*) |  | *GHBP*(*z*) + *HBS*(*z*) |

## The allpass filter

In addition to the filter types discussed so far, there is one more type that commonly appears in audio effects. This is the *allpass filter*, and it passes all frequencies with no amplification or attenuation. That is, the magnitude of the gain is 1 at all frequencies. But if the gain doesn't change, what does the filter do? The answer lies in the *phase* of the signal. The allpass filter introduces a frequency-dependent phase shift which is useful for many effects, especially the *phaser* (Chapter 4). Also, a simple delay is a type of allpass filter, since it changes the phase but not the magnitude of each frequency component. Figure 3.15 is a plot of the magnitude and phase response of another possible allpass filter.

The allpass filter may be given as

.

We can easily check that this has the allpass property,

.

If we look at any first order section from this high order filter, it can be written in the form,

,

and we see that for a pole at *c*, there is a corresponding zero at 1/*c*\*.

A digital allpass filter can be created rather simply with a single sample delay and two gain blocks. Figure 3.16 shows the block diagram; note that the feedforward and feedback gain are identical in magnitude but opposite in sign.

This is only a first order allpass filter, as in the following equation;

.

Its simply a version of Eq. with real valued coefficients.

More complicated, higher order versions with multiple delay blocks must be constructed in order to create an arbitrary phase response. See Chapter 4 for how this is accomplished with the phaser.

## Applications of filter fundamentals

### Exponential moving average filter

Let’s now take a look at one simple, but very important filter. The exponential moving average filter, also known as a smoothing filter or one-pole moving average filter, is given by

,

where is the amount of decay between adjacent samples. For instance, if  is 0.75, the value of each sample in the output signal is three quarters of the previous output and one quarter of the new input. The higher the value of x, the slower the decay. This filter is often used to smooth the effect of processing a signal, or to derive a smoothly changing estimate of signal level. As we will see in Chapter 6, it features prominently in dynamics processing.

The transfer function is

.

which gives one zero at zero, one pole at and gain of (1- We can also find the square magnitude response and, after a bit of manipulation, the phase response,

,

where  is normalized radial frequency.

The impulse response of this filter is:

.

Now consider the step function, . The step response of this filter is:

.

The time constant  is defined as the time it takes for this system to reach 1-1/*e* of its final value, i.e., . Thus from , we have

.

This is an important filter, since it can be used to very simply smooth a signal. Generally, one gives the time constant finds  from , and implements the filter in the time domain using . Note that this particular filter is primarily used for smoothing a signal in the time domain, and not explicitly for modifying frequency content. It simply provides a smooth output that can still quickly respond to sudden changes in the input.

### Loudspeaker Crossovers

Low pass, high pass and band pass filters are commonly used in *crossover networks* (or just *crossovers*)for loudspeakers. Larger hi-fi speakers almost always use more than one speaker driver to cover the entire audio frequency range. *Woofers* are large drivers (4”-15” diameter) used for the lower frequencies, and *tweeters* are smaller cone or dome-shaped drivers (0.5-2” diameter) used for the higher frequencies. Some speakers also include a third *midrange* driver to cover the frequencies between woofer and tweeter. Others use a single free-standing *subwoofer* to cover the lowest bass frequencies for the entire audio system; since the human ear cannot localize low bass frequencies, a single large subwoofer can be used to cover the low frequency content for a stereo or surround-sound system, which allows the remaining speakers to be smaller and less expensive.

For a speaker to work properly, the same signal cannot be sent to every driver. Woofers reproduce high frequencies poorly and with significant distortion, and sending bass frequencies to a tweeter can cause mechanical damage. The role of the crossover is to divide the audio signal into two or three frequency ranges which are sent to each driver. In a speaker with a woofer and tweeter (*2-way*), the crossover consists of a low pass filter for the woofer and a high pass filter for the tweeter (Figure 3.17a). The two filters have the same cut-off frequency (typically around 1-3kHz), which means that any given input frequency should go either to the woofer or to the tweeter, but not both. Of course, practical filters do not have perfectly sharp cut-offs, so there will always be a gradual transition between the two drivers. The gradual transition is a desirable property in practice, and very few crossovers use filters higher than fourth order. Second order filters are most commonly used in crossovers, and the occasional first order design can also be found.

When woofer, midrange and tweeter are used (a *3-way* speaker), there are two crossover points to consider: the transition between the woofer and the midrange and between the midrange and tweeter. In practice, a 3-way crossover looks like two 2-way crossovers in series (Figure 3.17b). A low pass filter is used for the woofer and a high pass filter for the rest of the signal. The output of the high pass filter is then split again into midrange and tweeter components. (An equivalent way to understand the operation is that the woofer has a low pass filter, the midrange a band pass filter, and the tweeter a high pass filter.) The woofer-midrange crossover frequency is typically in the range 200Hz-1kHz, and the midrange-tweeter crossover in the range 2-4kHz.

In the early days of loudspeakers, crossovers were built using high-power capacitors and inductors that could operate directly on the output of an audio power amplifier. This is still a common practice, but increasingly, self-powered speakers such as studio monitors will perform the crossover filtering on the line-level audio signal and use a separate power amplifier for each driver; this process is known as *bi-amplification* for a 2-way speaker or *tri-amplification* for a 3-way speaker. Crossovers at line level can generally have lower distortion and tighter component tolerances, and they can be more sophisticated in correcting slight imperfections in the frequency responses of the drivers.

Good crossover design can be as important as the quality of the drivers in determining the sound of a finished loudspeaker. Many subtle design variations are possible, which are discussed in detail in [14]. Crossovers are easily implemented digitally using the basic filters presented in this chapter, the main practical constraint being that there must be at least one DAC channel and one amplifier for every driver (e.g. 6 channels for a stereo 3-way speaker).

## Problems

1. Design a second order peaking filter for a 48kHz signal with center frequency 6kh and gain of 6dB, and bandwidth of 1kHz. Assume bandwidth is defined as when the square magnitude is the average of the two extremes, *GB*2=(1+*G*2)/2, where *G* is the linear gain.

2. Consider the second order filter having transfer function . Show that it is an all pass filter, that is, show that |*H*(z)|=1. Find the zeros of the filter as a function of the poles.

3. Consider again the exponential moving average filter. Find the cut-off frequency, if it is defined as where the square magnitude is halfway between the two extremes, |*H*(*z=*1)|2 and |*H*(*z=*-1)|2

4. Consider the moving average window . What is the transfer function, impulse response and step response of this filter? Define the time constant as was done for the exponential moving average, the time it takes for this system to reach 1-1/*e* of its final value, i.e., .

If the time constant of the moving average window is set equal to the time constant of the exponential moving average, what is the relationship between *W* and ?

5. An out of band shelving filter has unity gain at centre frequency c, bandwidth *B* and magnitude *G* at frequency 0 and at *fs*/2. It may be constructed by convertingthe prototype to a shelving filter, reverse the filter, shift the cutoff frequency to the bandwidth, then transform this to a bandpass filter. A second order design could also be constructed by manipulating other filter designs. Give the transfer function for a second order out of band shelving filter.

Figure .1. Ideal filters.



Figure .2. Pole zero plot (top) and square magnitude response for a fourth order prototype low pass filter.



Figure .3. Shifting the centre frequency of a low pass filter.

Figure .4. Shelving filter transformation.





Figure .5. Pole zero plot (top) and square magnitude response for a fourth order prototype low shelving filter (bottom). Compared to our prototype filter, it moves the zeros towards the pole positions on the imaginary axis.



Figure .6. Reversing the z domain to turn a low pass into a high pass filter.



Figure .7. Transforming a low pass filter into a band pass filter.



Figure .8. Pole zero plot for a 1st order (top left) and 4th order (top right) low pass filter with center frequency c=/4. On bottom, square magnitude response for the first order (solid line) and fourth order (dashdot line) filters.



Figure .9. Pole zero plot for a 1st order (top left) and 4th order (top right) high pass filter with center frequency c=/4. On bottom, square magnitude response for the first order (dashdot line) and fourth order (solid line) filters.



Figure .10. Pole zero plot for a 1st order (top left) and 4th order (top right) low shelving filter with center frequency c=/4 and gain at center frequency *G*=2. On bottom, square magnitude response for the first order (dashdot line) and fourth order (solid line) filters.



Figure .11. Pole zero plot for a 1st order (top left) and 4th order (top right) high shelving filter with c=/4 and *G*=2. On bottom, square magnitude response for the first order (dashdot line) and fourth order (solid line) filters.



Figure .12. Pole zero plot for a 2nd order (top left) and 8th order (top right) band pass filter with c=/4 and *B*=/8. On bottom, square magnitude response for the first order (dashdot line) and fourth order (solid line) filters.



Figure .13. Pole zero plot for a 2nd order (top left) and 8th order (top right) band stop filter with c=/4 and *B*=/8. On bottom, square magnitude response for the first order (dashdot line) and fourth order (solid line) filters.



Figure .14. Pole zero plot for a 2nd order (top left) and 8th order (top right) peaking filter with c=/4, *B*=/8 and *G*=2. On bottom, square magnitude response for the first order (dashdot line) and fourth order (solid line) filters.



Figure 3.15. The magnitude and frequency response for a particular allpass filter (first order, cut-off frequency of c=/4). All frequencies have unity gain but a different phase shift.

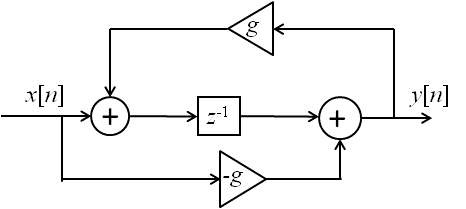


Figure 3.16. The block diagram for the first order digital allpass filter.



Figure .17. A loudspeaker crossover separates the incoming audio signal into several different frequency bands so that each loudspeaker is provided with content within a preferred frequency range. 2-way (a) and 3-way (b) loudspeaker crossovers are depicted.

# Filter Effects

In a sense, any audio effect could be considered a *filter* in that every effect performs mathematical operations on the input signal that produce some change in the content. This chapter, however, is concerned with effects based on the canonical types of filters defined in Chapter 3: low pass, high pass, band pass, peaking/notch, shelving and allpass filters. The effects that fall into this category are equalizers, a broad class of effects that adjust the frequency balance of the input audio signal; wah-wah, a musical effect typically used on guitar which is based on a peaking filter; and the phaser, an effect based on allpass filters that produces a sweeping, spacious sound similar to the flanger (Chapter 2).

## Equalization

Equalization (EQ for short) is one of the most common audio effects. It is the process of adjusting the relative strength of different frequency bands within a signal. The name “equalization” comes from the desire to obtain a flat (equal) frequency response from an audio system by compensating for non-ideal equipment or room acoustics. Peaks or troughs in the frequency response of a system are often described as ‘coloration’ of the sound, and equalization can be used to remove this coloration [15-17].

Equalization covers a broad class of effects, ranging from simple tone controls to sophisticated graphic and parametric equalizers. All EQ effects are based on *filters*, and most equalizers consist of multiple subfilters, each of which affects a single frequency band. In the following, we will go through the steps of designing an equalizer out of such subfilters and discuss the properties these subfilters should have.

### Theory

Equalizers range in complexity from simple two-knob tone controls to complex multi-band graphic and parametric equalizers, but at a fundamental level, all designs are based on the same collection of filter structures. Equalizer design therefore depends heavily on choosing the correct parameters for each filter, keeping in mind which parameters should be fixed by the design and which should be user-adjustable.

#### Two-Knob Tone Controls

Most stereo systems feature tone controls, which provide a simple and quick way to adjust the sound to suit the listener’s taste and compensate for the frequency response of the room. Tone controls are the simplest and possibly most common equalization system. A basic version consists of two knobs, typically labeled ‘bass’ and ‘treble’. These knobs are used to control the gain of low and high frequencies, respectively, through the use of shelving filters. Figure 4.1a shows a block diagram of a typical implementation.

Recall from 0 that shelving filters come in low and high shelf configurations. A low shelving filter has adjustable gain at low frequencies and unity gain at high frequencies; it is used in the bass control. The reverse is true for the high shelving filter, used in the treble control. Tone controls commonly use first order shelving filters, which produce a gradual 6dB/octave transition between low and high frequencies (Figure 4.2).

We can use the procedure described in Chapter 3 to design the shelving filters. However, now we would like the gain at the cut-off frequency, *Gc*, to be , where *G* is the gain of the shelf. That way, a low shelf and high shelf with the same cut-off frequency *c* and gain *G* will produce a constant gain *G* across all frequencies. The transfer function for the low shelving filter is as follows:

.

And for the high shelving filter:

.

In each case, the maximum gain of the shelf, G, is adjustable. Where G > 1, the bass (low pass case) or treble (high pass case) will be boosted; where G < 1, it will be cut. Typical tone controls have an adjustment range of +/-12dB, corresponding to approximately 0.25 < G < 4 for each filter. The cut-off frequency ωc is usually fixed. The values for bass and treble vary by manufacturer, but typically the bass control uses a lower ωc than the treble control. Cut-off frequencies for the bass control might range from 100Hz to 300Hz, and the treble control from 3kHz to 10kHz.

#### Three-Knob Tone Controls

In two-knob tone controls, the midrange frequencies (between bass and treble) are usually left unchanged. On some units, in addition to control of the bass and treble, there may be a “midrange” or “mid” control. This control is usually implemented as a peaking or notch filter. Its transfer function is derived in Chapter 3, where again, we define the gain at bandwidth to be :

.

The peaking or notch filter is depicted in Figure 4.3.

The knob on the midrange control affects the gain G at center frequency, which generally takes the same range of values as the bass and treble controls (e.g. 0.25 < G < 4). The center frequency ωc is generally fixed to be midway between the bass and treble controls, and the bandwidth B is chosen so that the midrange control affects the frequencies which are left unadjusted by the bass and treble controls.

#### Presence Control

Some systems, including many guitar amplifiers and mixing consoles, will have a ‘presence’ knob or button in addition to the tone controls mentioned above. This controls a peaking filter in the upper midrange frequencies. The transfer function for the peaking filter is identical to the midrange control described in the previous section, but the center frequency and the *quality factor* or *Q* (defined as center frequency over bandwidth) are higher, such that the presence control affects roughly the 2-6 kHz frequency band, where the human ear is very sensitive. The presence control is intended to simulate the effect of an instrument being physically present in the same room as the listener, and it can help bring out an instrument in a mix without changing its overall level.

#### Loudness Control

Many stereo amplifiers feature a “loudness” control, either as a knob or an on-off button. In this context, loudness has a different meaning than volume or amplitude (which refer to the total sound pressure level, or SPL, produced by the audio system). The sensitivity of human hearing is heavily dependent on frequency: for the same SPL, midrange frequencies will be perceived to be louder than very low or very high frequencies.

Figure 4.4 shows a plot of equal loudness contours for human hearing. Each curve indicates the SPL required to maintain the same perceived loudness (equivalent to 90 Phon, 80 Phon… 10 Phon, where 1 phon = 1 dBSPL at a frequency of 1 kHz) across all frequencies. Notice that the curves become flatter as overall SPL increases. This means that especially at low listening levels, bass and treble need to be boosted with respect to the midrange to be perceived as equally loud.

The loudness control is therefore designed to boost the low and high frequencies of the input signal. Two shelving filters (one for bass, one for treble) can be used for this purpose. When the loudness control is implemented as a continuous knob, its setting controls the gain of the low- and high-frequency shelves, with the midrange gain fixed at 1. When it is implemented as a button, switches between flat frequency response (no boost) and an inverse of the A-Weighting loudness curve, which is a rough approximation of the equal loudness contour near 40 phon. Thus it is used to boost that frequency content that we perceive to have additional attenuation at low listening levels.

#### Graphic Equalizers

The graphic equalizer is a tool for precisely adjusting the gain of multiple frequency regions. In contrast to the simple 2- or 3-knob tone control, a graphic equalizer can provide up to 30 controls for manipulating the frequency response. Structurally, a graphic EQ is simply a set of filters, each with a fixed center frequency. The only user control is the amount of boost or cut in each frequency band, which is often controlled with vertical sliders. The term “graphic” refers to the fact that the position of the sliders’ knobs can be understood as a graph of the equalizer’s magnitude response versus frequency. In other words, the position of the sliders resembles the frequency response itself, which makes the graphic equalizer intuitive to use despite the number of controls.

The basic unit of the graphic equalizer is the band. A band is a region in frequency defined by a center frequency*c* and a bandwidth B or *quality factor* Q. Recall that these three terms are related by *Q*=*c*/*B*.

The gain of each band is controllable by the user. Typical gains range from -12dB to +12dB (0.25 < G < 4), with 0dB (G = 1) meaning no change or “flat”. The same peaking or notch filter found in the midrange tone control, Eq. , can be used to create a single band of a graphic equalizer.

The center frequency ωc and bandwidth B are not user-adjustable. To choose the right values for these parameters, we must consider the relationship between the bands.

##### Bands in a Graphic Equalizer

The bands in a graphic equalizer are usually distributed logarithmically in frequency, to match human perception. Let us denote the normalized lower and upper cut-off frequencies of the *i*th band with *l*,i and *u*,i, respectively. As before, bandwidth is the difference between the upper and lower cut-off frequencies, *Bi*=*u,i**l*,*i*.

The frequency bands are adjacent [18], so the upper cut-off of band i will be the lower cut-off of band i+1, *u*,*i* =*l*,*i*+1.That is, input audio frequencies below this cut-off will be primarily affected by the gain control for band i, where input frequencies above it will be primarily affected by the gain control for band i+1.

The logarithmic distribution of the frequency bands can be specified using a fixed ratio *R* between each band, so*l*,*i*+1 = *R·**l*,*i*, *u*,*i*+1 = *R·**u,i*, or *Bi*+1 = *R·Bi*. We also consider the geometric mean of the two cut-off frequencies, , where you can see that we have the same relationship for the distribution of these values, *M*,*i*+1 =*R·**M*,*i*.

Two common designs are octave and 1/3-octave graphic equalizers. An octave is a musical interval defined by a doubling in frequency, so octave graphic equalizers will have the ratio R = 2 between each band. In a 1/3-octave design, each octave contains three bands, which implies R3 = 2 or *R*~1.26. So starting at 100 Hz, an octave spacing would have geometric mean frequencies at 200 Hz, 400 Hz, 800 Hz... and a 1/3-octave spacing would have filters centered at 126 Hz, 159 Hz, 200 Hz, etc.

The number of bands is determined by their spacing and the requirement to cover the entire audible spectrum. Octave graphic equalizers usually have 10 bands, ranging from 31Hz at the lowest to 16kHz at the highest. 1/3-octave designs have 30 bands ranging from 25Hz to 20kHz. These frequencies, shown in Table 2, are standardized by the ISO (International Standards Organization) [19].

The bandwidth can be easily related to the geometric mean of the cut-off frequencies,

.

Note that the geometric mean of the cut-off frequencies of a filter, *M*,*i* is not usually the true center frequency where the filter reaches its maximum or minimum value, *c*,*i.* From Chapter 3 and a bit of trigonometry, we can find a relationship between the upper and lower cut-off frequencies and the center frequency of a band pass, band stop, peaking or notch filter,

.

However, the geometric mean is usually quite close to the center frequency. Thus, the bandwidth scales roughly proportionally with the center frequency. Thus higher bands will have a larger bandwidth than lower ones. Since *Q* = *c*/*B*, this is another way of saying that the *Q* factor is nearly constant for each band in a graphic equalizer. From , we can estimate *Q* as

.

So for an octave (10-band) equalizer,  since R = 2. For a third-octave (30-band) equalizer, we find  since .

Ideally, the subfilter for the *i*th band has magnitude of the desired gain *Gi* inside the band and gain at bandwidth , so that at the cut-off frequency, when *Gi* = *Gi*+1 = *G*, we have

.

The individual filters are high order low-shelving filters, as derived previously.

We base our design on the lower cut-off frequencies, so that we express other parameters as

.

So the design of a graphic equalizer is as follows;

1. Choose the distribution of filters, i.e., set *R* for octaves, one third octaves,…
2. Choose the first lower cut-off frequency.
3. From this, find bandwidth and center frequency.
4. For a given gain, generate the peaking/notch filter.
5. Find next lower cut-off frequency.
6. Repeat steps 3-5 until the whole frequency range is covered.

This procedure could equally have been performed with band pass filters rather than peaking/notch filters. However, then the filters would have been arranged in parallel, not series.

Table . The ISO standard for octave and 1/3 octave frequency bands.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Octave bands | | | 1/3 Octave bands | | |
| Lower frequency *fl* (Hz) | Geometric mean frequency *fM* (Hz) | Upper frequency *fu* (Hz) | Lower frequency *fl* (Hz) | Geometric mean frequency *fM* (Hz) | Upper frequency *fu* (Hz) |
| 22 | 31.5 | 44 | 22.4 | 25 | 28.2 |
| 28.2 | 31.5 | 35.5 |
| 35.5 | 40 | 44.7 |
| 44 | 63 | 88 | 44.7 | 50 | 56.2 |
| 56.2 | 63 | 70.8 |
| 70.8 | 80 | 89.1 |
| 88 | 125 | 177 | 89.1 | 100 | 112 |
| 112 | 125 | 141 |
| 141 | 160 | 178 |
| 177 | 250 | 355 | 178 | 200 | 224 |
| 224 | 250 | 282 |
| 282 | 315 | 355 |
| 355 | 500 | 710 | 355 | 400 | 447 |
| 447 | 500 | 562 |
| 562 | 630 | 708 |
| 710 | 1,000 | 1,420 | 708 | 800 | 891 |
| 891 | 1,000 | 1,122 |
| 1,122 | 1,250 | 1,413 |
| 1,420 | 2,000 | 2,840 | 1,413 | 1,600 | 1,778 |
| 1,778 | 2,000 | 2,239 |
| 2,239 | 2,500 | 2,818 |
| 2,840 | 4,000 | 5,680 | 2,818 | 3,150 | 3,548 |
| 3,548 | 4,000 | 4,467 |
| 4,467 | 5,000 | 5,623 |
| 5,680 | 8,000 | 11,360 | 5,623 | 6,300 | 7,079 |
| 7,079 | 8,000 | 8,913 |
| 8,913 | 10,000 | 11,220 |
| 11,360 | 16,000 | 22,720 | 11,220 | 12,500 | 14,130 |
| 14,130 | 16,000 | 17,780 |
| 17,780 | 20,000 | 22,390 |

#### Parametric Equalizers

The parametric equalizer is perhaps the most powerful and flexible of the equalizer types. Where each band of a graphic equalizer is adjustable only in gain, each band in a parametric equalizer has three adjustments: gain, center frequency and *Q* (or bandwidth). Most parametric equalizer bands are implemented with the same peaking/notch filter we have previously seen, Eq. . Notice that the three controls relate directly to the parameters *G*, *c* and *B* (where *B* = *c* /*Q*):

Since each band is more complex on a parametric equalizer compared to a graphic equalizer, a complete parametric equalizer unit will typically have fewer bands (an EQ section on a mixing console might have 1 or 2 bands; a dedicated rack-mount unit 4 or 6 bands).

In addition to the gain, center frequency and *Q* controls, some bands may have a switch to act as a shelving filter rather than the typical peaking/notch filter. In shelving mode (Figure 4.2), gain G is applied to all frequencies below the center frequency (low shelving filter, sometimes used in the lowest-frequency band) or to all frequencies above the center frequency (high shelving filter, sometimes used in the highest-frequency band). These shelving filters are similar to those found in the basic tone control, but where the tone control usually uses first order shelving filters, the parametric equalizer uses either first or second order filters.

**Who invented the parametric equaliser?**

Early filters included bass and treble controls without adjustable center frequency, bandwidth, and cut or boost. So sound engineers could only make broad, overall changes to a signal. When graphic equalizers arrived, engineers were still limited to the constraints imposed by the number and location of bands.

In the 1960s Harold Seidel of Western Electric and Bell Telephone devised a tunable parametric filter. Daniel Flickinger then introduced an important tunable equalizer in early 1971. His circuit allowed arbitrary selection of frequency and cut/boost level in three overlapping bands over the entire audio spectrum.

In 1966, Burgess Macneal and George Massenburg began work on a new recording console for International Telecomm Incorporated. During the building of the console, Macneal and Massenburg, who was still a teenager, conceptualized an idea for a sweep-tunable EQ that would avoid inductors and switches. Soon after, Bob Meushaw, a friend of Massenburg, built a three-band, frequency adjustable, fixed-Q equalizer.

When asked who invented the parametric equalizer, Massenburg stated “four people could possibly lay claim to the modern concept: Bob Meushaw, Burgess Macneal, Daniel Flickinger, and myself...Our (Bob's, Burgess' and my) sweep-tunable EQ was borne, more or less, out of an idea that Burgess and I had around 1966 or 1967 for an EQ … By 1969 I was spending all of my time designing circuitry sufficient to get to an elegant user interface: we perceived this as three controls adjusting, independently, the parameters for each of three bands for a recording console.... I remember agonizing over the topology for the EQ for months, and asking everyone I knew for help. I wrote and delivered the AES paper on Parametrics at the Los Angeles show in 1972 [4] … it's the first mention of ‘Parametric’ associated with sweep-tunable EQ... what I'm proudest of is less in designing devices alone, and more in exploring the ever-expanding applications and uses of gear, and then applying that knowledge to designs.”

George Massenburg went on to win many awards for both his technical contributions to recording technology, and his critically acclaimed recordings. And today, the parametric equalizer is pervasive in audio signal processing, and a fundamental tool in digital filtering techniques as well.

**Summary**

Every equalization effect, from the simplest tone control to the most complex parametric EQ, is based on the same set of filter types. The main differences between them are the number of individual subfilters and the controls presented to the user. First and second order filters are most commonly used in all equalizers, as these provide sufficient flexibility while minimizing complexity and artifacts sometimes found in higher-order filters. The next section discusses the practical implementation of the required filters for each type of equalizer.

### Implementation

#### General Notes

All types of equalizers discussed in this chapter were originally analog effects, and their digital implementation follows the analog designs as closely as possible. IIR (infinite impulse response) filters are used throughout. Partly this is to emulate analog models, but processing delay is also a concern whenever any effect is used in live performance. While the maximum allowable total delay in an audio system may be a matter of discussion, it is safe to require each individual device to have the lowest possible processing delay, to allow cascading of several devices. This suggests using minimum-phase IIR filters instead of linear phase FIR filters, which typically exhibit higher latency for the same performance.

While some simple equalization effects such as the presence control can be implemented with a single filter section, most effects require two or more independent filters (referred to here as subfilters) to produce the overall result. In the case of the graphic equalizer, thirty or more subfilters might be used, and the way they are connected to one another becomes extremely important. Two topologies are commonly used: parallel connections, where audio is processed independently through several subfilters and the results added together, and cascade (or series) connections, where the output of one subfilter becomes the input to the next one. The optimal choice for each type of equalizer is discussed in the following sections.

#### Tone Control Architecture

The two-knob tone control is implemented with a low shelving filter for the bass and a high shelving filter for the treble. First order filters are used to create smooth (6dB/octave) transitions between affected and unaffected frequency regions. When a third midrange knob is used, it is often created from a second order peaking/notch filter. Cascade connections are most commonly used for these controls, as shown in Figure 4.1(b).

See Chapter 3 for details of implementing each subfilter. Calculating the output of a first order IIR filter requires not only the current input sample x[n], but also the previous input x[n-1] and the previous output y[n-1]. The second order peaking/notch filter also requires values for x[n-2] and y[n-2]. Therefore, a complete three-knob tone control will require 11 stored previous values.

#### Calculating Filter Coefficients

The coefficients for each filter in the tone control depend on the settings of the three knobs. As seen in Eq.s to , calculating the coefficients requires a complex series of operations including trigonometric functions and floating-point division, all of which are more computationally expensive than multiplication. It is therefore advisable to only recalculate the filter coefficients when the knob positions change. The calculated coefficients can then be stored and recalled each time they are needed to process new audio samples.

#### Presence and Loudness Architecture

The presence control can be implemented with a single peaking/notch filter with a center frequency around 4kHz and a Q of approximately 1. This corresponds to a bandwidth of 4kHz, thus covering the 2-6kHz frequency range. The presence knob adjusts the gain of this band; as with all peaking/notch filters, the gain outside of the band is 1.

The loudness control is architecturally nearly identical to the two-knob tone control. A *low* shelving and *high* shelving filter connected in cascade mode (Figure 4.1a). Their cut-off frequencies are fixed by design (100 Hz in the bass and 8 kHz in the treble are typical values), just as the cut-off frequencies are fixed in the tone control. However, a single knob or button controls the passband gain of both filters. The midrange gain remains at 1.

#### Graphic Equalizer Architecture

The graphic equalizer can be considered a generalization of the basic tone control with more bands; however, it is often implemented differently. The bass and treble controls in a basic tone control are connected in *cascade*, with each control affecting part of the frequency range and leaving the rest unaffected. A cascade of *peaking/notch* filters can be used in the graphic equalizer, but it involves the series connection of up to 31 subfilters, which can have unexpected harmful effects.

The discussion in this chapter has focused mostly on the *magnitude response* of each filter, but *phase distortion* needs to be considered as well. Every IIR filter will have a nonlinear phase response with respect to frequency, and when filters are connected in cascade, the phase response of each section will add. In some cases, the combined phase shifts can produce audible artifacts which color the sound. FIR filters, which are less commonly used in equalization, can have linear phase, avoiding this particular form of distortion. However, FIR filters typically have a longer *group delay* for the same filter performance, so cascading many FIR filters in series might produce an audible delay between input and output.

Instead of using a cascade of peaking/notch filters, graphic equalizers often use a collection of band pass filtersarranged in *parallel*, as shown in Figure 4.5. The audio input is split and sent to the input of every band pass filter. Each filter allows only a small frequency band to pass through. The center frequencies and bandwidths are configured so that if all the outputs were added together, the original signal would be reconstructed. The controls on the graphic equalizer are then implemented by changing the gain of each band pass filter output before the signals are summed together.

Parallel connection of band pass filters avoids the accumulating phase errors (and, potentially, quantization noise) found in the cascade. It also has a secondary benefit: the coefficients of each band pass filter depend only on their center frequency and Q and do not change with the setting of the gain sliders. Coefficients can therefore be calculated once when the effect is initialized, given information on the sample rate. Changing gain only involves changing a single number that multiplies the output of the band pass filter.

#### Parametric Equalizer Architecture

A single section of a parametric equalizer is created from a second order peaking/notch filter (or in certain cases, a second order shelving filter). When multiple sections are used, they are always connected in cascade so that the effects of each subfilter are cumulative. Even though the parametric equalizer is among the most complex for the user to control, its implementation is no more complex than a simple tone control. Also like the tone control, filter coefficients should be recalculated every time the user changes any knob, but for reasons of efficiency should not be recalculated every audio sample.

#### Code example

Implementing a parametric equalizer includes two main tasks: first, the coefficients of the IIR filter must be calculated whenever the user changes the center frequency, gain or bandwidth; and second, the filter must be applied to each audio sample. This code shows the calculation of coefficients. For efficiency, it is run only when the user updates the controls:

void ParametricEQFilter::makeParametric(const double discreteFrequency,

const double Q,

const double gainFactor) noexcept

{

/\* Limit the bandwidth so we don't get a nonsense result from tan(B/2) \*/

const double bandwidth = jmin(discreteFrequency / Q, M\_PI \* 0.99);

const double two\_cos\_wc = -2.0\*cos(discreteFrequency);

const double tan\_half\_bw = tan(bandwidth / 2.0);

const double g\_tan\_half\_bw = gainFactor \* tan\_half\_bw;

/\* setCoefficients() takes arguments: b0, b1, b2, a0, a1, a2

\* It will normalise the filter according to the value of a0

\* to allow standard time-domain implementations

\*/

setCoefficients (1.0 + g\_tan\_half\_bw, /\* b0 \*/

two\_cos\_wc, /\* b1 \*/

1.0 - g\_tan\_half\_bw, /\* b2 \*/

1.0 + tan\_half\_bw, /\* a0 \*/

two\_cos\_wc, /\* a1 \*/

1.0 - tan\_half\_bw /\* a2 \*/);

}

This code is based on the Juce IIRFilter class, which implements a generic two-pole, two-zero IIR filter. The ParametricEQFilter object is a subclass of IIRFilter which implements the coefficients for a parametric equalizer. setCoefficients() saves the values of the six coefficients so they can later be used to calculate output samples. jmin() is a simple macro provided by Juce to return the minimum of two numbers.

The following code applies the filter to a block of audio samples for a single channel:

int numSamples; // Indicates how many audio samples to process

float \*channelData; // Array of audio samples, length numSamples

float coefficients[6]; // Previously calculated filter coefficients

float x1, x2, y1, y2; // Previous values of the input and output

for (int i = 0; i < numSamples; ++i)

{

const float in = channelData[i];

float out = coefficients[0] \* in /\* b0 \*/

+ coefficients[1] \* x1 /\* b1 \*/

+ coefficients[2] \* x2 /\* b2 \*/

- coefficients[4] \* y1 /\* a1 \*/

- coefficients[5] \* y2; /\* a2 \*/

x2 = x1;

x1 = in;

y2 = y1;

y1 = out;

channelData[i] = out;

}

Here, coefficients is an array containing the values b0, b1, b2, a0, a1 and a2, as previously calculated and stored using setCoefficients(). Notice that a0 is not used in this calculation. When the coefficients are set, they are normalized so a0 = 1, eliminating the need for it in further calculations. The variables x1, x2, y1 and y2 hold the last two inputs and outputs, respectively. For example, if in represents x[n], then x1 represents x[n-1] and x2 represents x[n-2].

### Applications

#### Graphic Equalizer Application

Graphic equalization is more commonly found in live performance and recording studios than in most home stereo systems. One common use of graphic equalization is to “tune” a room, adjusting the equalizer to roughly compensate for resonances in the room or imperfections in the frequency response of the speakers [20]. The goal is to achieve a desired frequency response, flattening out extremes, reducing coloration in the sound and achieving greater sonic consistency among performance venues. However, graphic equalizers are occasionally found in consumer stereo systems and even in digital music player software, where they can be used as a more flexible form of tone control for adjusting the sound to taste.

#### Parametric Equalizer Application

Parametric equalizers allow the operator to add peaks or notches at arbitrary locations in the audio spectrum. Adding a peak can be useful to help an instrument be heard in a complex mix (see also the *presence control* earlier in the chapter), or to deliberately add coloration to an instrument’s sound by boosting or reducing a particular frequency range [21]. Notches can be used to attenuate unwanted sounds, including removing power line hum (50Hz or 60Hz and sometimes their harmonics) and reducing feedback. To remove artifacts without affecting the rest of the sound, a narrow bandwidth would be used. To deliberately add coloration to an instrument’s sound by reducing a particular frequency range, a wider bandwidth might be used.

Wah-wah

The sound of the *wah-wah* effect resembles its name: wah-wah is a filter-based effect which imparts a speech-like quality to the input sound similar to a voice saying the syllable “wah”. Wah-wah is most commonly known as a guitar effect which was popularized by Jimi Hendrix, Eric Clapton and others in the late 1960s. However, its origins go back to the early days of jazz, when trumpet and trombone players achieved a similar sound using mutes.

The wah-wah audio effect uses a *band pass* or *peaking* filter whose center frequency is changed by a foot pedal. In some pedals, the mix between original and filtered signal can be controlled by a separate knob, as shown in Figure 4.6 [22].

### Theory

#### Basis in Speech

In speech, formants are the peaks in the frequency spectrum when a human voice utters a sound, and are due to resonances in the vocal tract. For many vowel sounds, at least three formants can be easily identified. Humans listen for and assign meaning to the relative spacing of the first three formants of the human vocal tract.

Formants are distinct from the fundamental frequency (or pitch) of the voice, which is the frequency at which the vocal folds vibrate. We hear and notice the fundamental frequency, but in most languages its exact location is not important when assigning meaning to vocal sounds. We also notice relative shifts of the fundamental frequency, but these are often associated with emotional states, such as when someone's voice goes up in pitch when under stress. But the relative positioning of formants, especially the first two formants, represents important information in how we interpret vowel sounds.

The wah-wah effect gives a voice-like quality to an input signal by simulating the formants found in speech [22]. The first formant of the [u] vowel is roughly located around 300Hz, and the first two formants of the [a] vowel are located at approximately 750Hz and 1200Hz. Therefore, the wah-wah simulates the transitions between vowels by adjusting the center frequency of its filter in roughly this range (the exact range depends on the manufacturer and model of pedal, but a range between 400Hz and 1200Hz is typical). Wah-wah could be considered a simple form of speech synthesizer, though not close enough to be truly mistaken for a vowel sound.

#### Basic Wah-Wah

Figure 4.6 shows a block diagram of the wah-wah effect. A single second order filter is typically used, with several possible variations; peaking, band pass or resonant low pass filters. As discussed previously, a peaking filter will boost the midrange frequencies while leaving all other frequencies with a gain of 1, and a band pass filter will boost the midrange frequencies and gradually roll off the low and high frequencies to 0. A resonant low pass filter will pass the low frequencies with a gain of 1, create a peak with magnitude *Gc* in the midrange around the cut-off frequency *c*, and gradually roll off the high frequencies to 0. A second order resonant low pass filter is given below,



where *c*=tan(*c*/2). The magnitude response of this filter is shown in Figure 4.7.

The serendipitous invention of the wah-wah pedal

The first wah-wah pedal is attributed to Brad Plunkett in 1966, who worked at Warwick Electronics Inc., which owned Thomas Organ Company. Warwick Electronics acquired the Vox name due to the brand name's popularity and association with the Beatles. Their subsidiary, Thomas Organ Company, needed a modified design for the Vox amplifier, which had a midrange boost, so that it would be less expensive to manufacture.

In a 2005 interview [2], Brad Plunkett said, I “came up with a circuit that would allow me to move this midrange boost … As it turned out, it sounded absolutely marvelous while you were moving it. It was okay when it was standing still, but the real effect was when you were moving it and getting a continuous change in harmonic content. We turned that on in the lab and played the guitar through it... I turned the potentiometer and he played a couple licks on the guitar, and we went crazy.

A couple of years later... somebody said to me one time, ‘You know Brad, I think that thing you invented changed music.’”

The gain and Q of the filter are generally fixed by design, but the center frequency is adjustable under the control of a foot pedal. As mentioned, the centre frequency commonly takes a range of around 400-1200Hz. Above and below this range, the effect loses its vocal quality. A mix control is sometimes used to vary the intensity of the effect by mixing between the filtered and unfiltered signals. In the case of a peaking filter, this function could be equivalently implemented by changing the gain at the center frequency.

#### Auto-Wah

For the standard wah-wah, as used commonly on guitar pedals, the player manually controls the center frequency. In the auto-wah effect, the center frequency is controlled automatically. Two variations of auto-wah are commonly found. In the first, the center frequency sweeps back and forth following a low-frequency oscillator (LFO) with an adjustable frequency, typically around 1-2 Hz. The range of filter center frequencies is similar to the basic wah-wah, but it is often adjustable by the user.

In the second auto-wah variation, the center frequency is automatically adjusted according to the amplitude of the input signal. This arrangement is sometimes known as an envelope follower, and when used on a guitar, it produces a moving resonance on every note. Louder signals push the center frequency upwards (towards the “ah” sound). The sensitivity of this process is typically a user-adjustable parameter, as are the attack time and release time of the envelope. More details on envelope calculation can be found in Chapter 6.

#### Tremolo-Wah

If the LFO-controlled auto-wah is combined with a periodic change in amplitude (tremolo; 0), the result is an effect known as tremolo-wah. The effect is generally equivalent to placing a tremolo and auto-wah effect in series:

.

where y[n] is the output of the entire tremolo-wah, w[n] is the output of the auto-wah section and g[n] is a time-varying gain factor, and ** is normalised frequency 2*f*/*fs* as usual. g[n] may also be calculated on a logarithmic scale since the human sensitivity of loudness follows a logarithmic relation. The same LFO can be used to control both amplitude and wah center frequency, but often the two move independently with different frequencies, or with the same frequency but out of phase.

**Other Variations**

The standard wah-wah pedal has only one resonant peak, in contrast to the two or three identifiable formants in most vowel sounds. By adding additional resonances, even more vocal-like sounds from a wah-wah effect are possible. Some pedals use a second peaking filter circuit whose center frequency that moves around in a different manner than the main filter, for example following the second formant in the “oo” and “ah” vowels. This produces an effect much closer to human speech.

Implementation

Wah-wah and wacka-wacka

The wah-wah effect is incredibly expressive. Its associated with whole genres of music, and it can be heard on many of the most influential funk, soul, jazz and rock recordings over the past 50 years.

Jimi Hendrix would sometimes use the wah-wah effect while leaving the pedal in a particular location, creating a unique filter effect that did not change over time. However, in ‘Voodoo Child (slight return)', Hendrix muted the strummed strings while rocking the pedal, creating a percussive effect. The sweeping of the wah-wah pedal is more dramatic in the louder versus and the chorus, emphasizing the song’s blues styling.

The ‘wacka-wacka’ sound that Hendrix created soon became a trademark of a whole subgenre of 1970s funk and soul. Melvin ‘Wah-Wah Watson’ Ragin, a highly respected Motown session musician, is renowned for his use of the wah-wah pedal, especially on The Temptations ‘Papa Was A Rolling Stone’. This distinctive ‘wacka-wacka’ funk style of soon became a feature of urban black crime dramas, such as in Isaac Hayes’ ‘Theme from Shaft,’ Bobby Womack’s score to ‘Across 110th Street’ and Curtis Mayfield’s ‘Superfly.’

Another unusual use of the wah-wah pedal can be heard on the Pink Floyd song ‘Echoes.’ Here, screaming sounds were created by plugging in the pedal back to front, that is, the amplifier was connected to the input and he guitar was connected to the pedal’s output.

Of course, use of wah pedals is not reserved just to guitar. Bass players have used wah-wah pedals on well-known recordings (Michael Henderson playing with Miles Davis, Cliff Burton of Metallica, …). John Medeski and Garth Hudson use the pedals with Clavinets. Rick Wright employed a wah-wah pedal on a Wurlitzer electric piano on the Pink Floyd song ‘Money,’ and Dick Sims used it with a Hammond organ. Miles Davis’s ensembles used it to great extent, both on trumpet and on electric pianos. The wah-wah is frequently used by electric violinists, such as Boyd Tinsley of the Dave Matthews Band. Wah wah pedals applied to amplified saxophone also feature on albums by Frank Zappa and David Bowie.

#### Filter Design

Like equalization, wah-wah nearly always uses second order IIR filters. Two of the filter types used in the wah-wah are the same as those found in the various types of equalizer. The peaking/notch filter is used in the parametric EQ, and a passable wah-wah effect can be obtained from a parametric EQ by choosing large gain (up to 12dB) and high Q (values from 2-10 are typical) and varying the center frequency. The band pass filter, used in some wah-wah implementations, is also found in the graphic EQ. Again a high Q is often used. The design and implementation of these filters are the same as in the equalizers.

Most analog wah-wah pedals for guitar use band pass filters. Analog synthesizers typically use resonant low pass filters to achieve similar effects. Resonant low pass filter coefficients can be calculated similarly to more commonly-used second order low pass filters, but substituting a higher Q. Values from 2-10 might be found in the wah-wah, compared to 0.71 in a standard Butterworth low pass filter. The resonant low pass filter creates a peak in the frequency response at the cut-off frequency which is responsible for the “wah” effect.

As with the equalizers, the filter coefficients should be recalculated if and only if the center frequency (or cut-off frequency) has changed. Changing the mix control in the basic wah-wah (Figure 4.6) does not require recalculating the filter. In any variation of the auto-wah, the center frequency changes each sample and continuous recalculation of the coefficients is inevitable. However, the computational load can be reduced by recalculating the coefficients less frequently than each audio sample. For example, at a sample rate of 44.1kHz, recalculating the coefficients every 16 samples will still update the center frequency over 2700 times per second, improving efficiency without any significant difference in audio quality.

#### Low-frequency oscillator

In one variant of the auto-wah, a low-frequency oscillator (LFO) controls the center frequency of the filter. Typical parameters include LFO frequency, LFO waveform, minimum frequency and sweep width. Not all parameters will be user-adjustable; some will be fixed by design. LFO frequency (fLFO) is the number of cycles per second the center frequency oscillates; typical values range from 0.2Hz to 5Hz. LFO waveform controls the shape of the center frequency variation; sinusoidal waveforms are most common in the auto-wah. Minimum frequency (fmin) sets the lowest center frequency for the filter, typically no less than around 250Hz and often higher to maintain the vocal effect. Sweep width (W), expressed in Hz, is the difference between the minimum and maximum center frequencies across an entire oscillation. The center frequency of the filter over time with sinusoidal LFO and sample rate fs can be written as:

.

Further considerations on LFO implementation can be found in Chapter 2.

#### Envelope Follower

The envelope follower variant of the auto-wah scales the center frequency of the filter proportionally to the level of the input signal. The instantaneous value of each sample is a poor measure of a signal’s level, so a level detector must be used to calculate a local average value. Level detectors based on the exponential moving average are discussed in detail in Chapter 6, and the same types of level detectors used in the compressor can be used in the envelope follower wah. A common level detector has a variable attack time *A*and release time **R; its operation is given by:

,

where  and xL[n] is the input signal. Attack time and release time are often user-adjustable parameters. The other parameters in the envelope follower wah are minimum frequency and sweep width, which work analogously to the LFO case. We can therefore write the center frequency fc as a function of the level detector value yL[n]:

.

yL[n] is taken to always be positive, and assuming the input signal is scaled to have a maximum value of 1, the center frequency will reach a maximum value of fmin + W, just as with the LFO auto-wah.

#### Analog Emulation

Musicians often become attached to particular brands of wah-wah pedal to achieve their signature sound. Even when the gain, center frequency and Q of a digital filter are tuned identically to the analog case, the sound may still be subtly different. Part of the distinct sound of many analog wah pedals comes from nonlinear distortion introduced by the electronic components, especially the iron-core inductors used in the resonant filters. Nonlinear distortion, discussed in Chapter 7, adds new harmonic and intermodulation frequency components to the output signal which were not present in the input. Precise replication of these effects requires detailed numerical simulation of the behavior of each circuit element, which is beyond the scope of this text. References for further reading on analog modeling are given in [23].

Phaser

The *phase shifter* (or *phaser*) creates a series of *notches* in the audio spectrum where sound at particular frequencies is attenuated or eliminated. The flanger (Chapter 2) also produces its characteristic sound from notches, and in fact the flanger can be considered a special case of phasing. However, where the flanger is based on delays, the phaser uses *allpass filters* to create phase shifts in the input signal. When the allpass-filtered signal is mixed with the original, notches result from destructive interference. Where the flanger always generates evenly-spaced notches, the phaser can be designed to arbitrarily control the location of each notch, as well as their number and their width. Like the flanger, though, the phaser’s characteristic sound comes from the sweeping motion of the notches over time.

### Theory

#### Basic Phaser

The notches needed for the phaser are most often implemented using allpass filters. Allpass filters, whose design is discussed in Chapter 3, pass all frequencies with no change in magnitude, but they introduce a frequency-dependent phase lag. The output of the allpass filter is then added to the original signal, as in Figure 4.8. The relative level of the filtered signal can be adjusted by a *depth* (or *mix*) control.

Mixing the original and filtered signals creates notches in the frequency response according to the principle of constructive and destructive interference, just as in the flanger. At certain frequencies, the allpass filters will introduce a phase shift of 180 degrees or an odd multiple thereof (540, 900, etc.). This is equivalent to inverting the input, and when the original and filtered signals are added together with equal weight (*depth* = 1), they will cancel completely, resulting in a notch at that frequencies. Frequencies near the notch, which experience nearly 180 degrees of phase shift, will also be attenuated.

The pure delay used in the flanger can also be considered an allpass filter with *linear phase*: a delay produces no change in magnitude for any frequency, but it produces a phase lag proportional to the input frequency. Therefore, the phase response of a pure delay hits odd multiples of 180 degrees (-180, -540, -900, -1260; the negative value indicates the output phase *lags* the input) at evenly spaced frequencies, and the notches will also appear at evenly spaced frequencies.

IIR allpass filters (Chapter 3), like all IIR filters, are not linear phase, so the phase shift they create is not a linear function of frequency. By changing the order, *Q* and center frequency of the filter, many variations in notch location and width are possible. The number of notches is determined by the number of times the phase crosses an odd multiple of 180 degrees, which in turn is determined by the order of the filter. However, another useful property of allpass filters is that a cascade (series connection) of several allpass filters is itself an allpass filter. Phaser effects thus often consist of a several simpler allpass filters in series, where the total phase lag is the sum of each filter’s phase response.

#### Low-frequency oscillator

Like the chorus and flanger effects, a low-frequency oscillator (LFO) produces a periodic change in notch location over time. It does this by changing the center frequencies of the allpass filters. But where the chorus and flanger commonly use sinusoidal or triangular LFO waveforms, the phaser often changes the notch frequencies in an *exponential* pattern over time. This more closely corresponds to human hearing, where perceived pitch is an exponential function of frequency. Example LFO waveforms are discussed in the Implementation section.

#### Phaser with Feedback

Like the flanger, some phasers incorporate feedback (sometimes called regeneration) between the allpass filter output and input, as shown in Figure 4.9. Like other effects incorporating feedback, the feedback gain of the phaser must be strictly less than 1 to maintain stable operation. The effect of feedback is to increase the Q of the allpass filters, making the phase transitions more steep and therefore making the notches sharper.

#### Stereo Phaser

Just as with stereo flangers and chorus units, a stereo phaser can be created from two monophonic phasers with different filter settings, creating notches at different frequencies. Typically, the notches are controlled by two low-frequency oscillators in quadrature phase, where the output of one oscillator trails the other by 90°.

An optional addition is to selectively mix the outputs of each filter, which can create additional notches (Figure 4.10). The phaser affect can even be created acoustically with no mixing at all: each *feed-across* gain is set to zero and each output is sent to a separate speaker. In this *spatial phaser* arrangement, notches exist at different points in the room where the signal from each speaker cancels out in the air. Moving around the room will change the location of the notches.

### Implementation

#### Allpass Filter Calculation

The phaser uses first order or second order IIR allpass filters whose center frequencies vary over time, as shown in Figure 4.11a. A first order allpass has 0° phase lag at 0Hz and a total phase lag of 180° at high frequencies. By itself, a single first order section is not enough to create a moving notch in the phaser. A second order allpass has a total phase of 360° at high frequencies. Since the phase lag increases monotonically with frequency, a single second order section produces a single notch at the frequency with 180° phase lag (which is the cut-off frequency of the allpass filter) as shown in Figure 4.12a. To achieve more notches, multiple allpass sections are placed in series.

A common analog phaser design uses four first order allpass filter sections in series (or, equivalently, two second order sections). This produces 720° of total phase shift and therefore two notches (at 180° and 540°). In a typical design, all four first order allpass filters might have the same center frequency (that is, the frequency at which the phase lag is 90°). This does not mean that both notches fall at the same frequency. Rather, the total phase lag is the sum of the contributions of each filter, and the notches occur where this sum reaches 180° and 540°. Therefore, in this four-section design, the notches fall where the phase shift of each individual filter is 45° or 135° (Figure 4.12b).

If six first order allpass sections are used instead, the total phase lag is 1080° and three notches will result (180°, 540°, 900°). Again, each section can be tuned to an identical center frequency, but the notches will occur in different locations than in the four-section design (Figure 4.12c), this time occurring where each filter contributes 30°, 90° or 150° of phase lag.

The transfer functions for an allpass filter were derived in Chapter 3. We can rewrite these transfer functions, replacing the arbitrary constants by terms relating to how the filter modifies the phase. A first order allpass filter may be given as,



The cut-off frequency *c* is where the phase response reaches 90o.



For a second order design, we can write the allpass filter [10] as



Here, the cut-off frequency *c* is where the phase response reaches 180o and the bandwidth *B* is the difference between the two frequencies where the phase response reaches 90o and 270o. Note that both Eq. and conform with the format of any allpass filter given in Chapter 3.

#### Alternate Implementation

An alternate approach to phaser design uses a set of notch filters (Figure 4.11b). in place of the allpass sections (Figure 4.11a). The notch filters can be implemented as a cascade of second order IIR sections, similar to a parametric equalizer with high Q and gain 0 at the center frequency of each section. This implementation does not require the filter output to be mixed with the direct sound since the notches are created directly, however a mix can still be used to vary the intensity of the effect. Like the allpass implementation, the center frequency of the notch filters is varied with a low-frequency oscillator.

#### LFO Waveform

Just as with the flanger, the sound of the phaser results from how the notch frequencies change over time. In contrast to sinusoidal low-frequency oscillators commonly found in other effects, exponential motion of the notch frequencies is often used in the phaser. Specifically, a waveform that is triangular in the log-frequency domain can be used:

,

where fmin is the minimum center frequency, **LFO=fLFO /*fs* is the normalised LFO frequency and W = fmax/fmin is defined as the ratio of the maximum to minimum frequency. Note that this definition of W differs from the wah-wah effect described earlier in the chapter. Here we define the triangle waveform to take a range of values between 0 and 1 over a complete oscillation:

.

#### Analog and Digital Implementations

Filters implemented digitally (discrete time) behave slightly differently than their analog (continuous time) counterparts, due to the fact that frequency is unbounded in the continuous-time domain but restricted to the [0, 2π) range in discrete time. Techniques exist to convert continuous-time prototype filters into discrete-time equivalents, including the *bilinear transform* with pre-warping. However, even with these methods, the response of the continuous and discrete filters diverge near the Nyquist frequency.

Figure 4.13 shows the differences between the continuous and discrete filter phase responses and the resulting effect on phaser notch location. For an allpass center frequency f*c* = 3 kHz and four second order sections, the highest notch is 2.5 semitones (15%) lower in discrete time than it would be in an analog implementation. When f*c* = 10 kHz, the top notch is 12.8 semitones (52%) lower, over an octave of difference. Fortunately, the range of center frequencies used in the phaser rarely extends much beyond 1 to 2 kHz (though individual notch locations may be higher), so the differences present only a minor concern when emulating analog phasers.

#### Common Parameters

Depth (Mix/Level) - The depth controls the amount of allpass filtered signal that is added to the output. At a depth of zero, only the original signal appears at the output. At a depth of 1, the mix between original and filtered signals is equal, producing the deepest notches. Some phasers will use “depth” to refer to what this book labels “sweep width”, the frequency range in which the notches move, so it is important to know the convention used by any particular phaser unit.

Sweep Width (Range) - This parameter controls the frequency range across which the notches sweep. Possible variations include fixing the minimum frequency location and using the sweep width to change the maximum frequency, or offering separate controls for minimum and maximum frequency.

Feedback/Regeneration - This control adjusts the feedback gain between output and input of the allpass filter section (Figure 4.9). The value must be strictly less than 1 to avoid instability. Using feedback can produce sharper, more pronounced notches.

LFO frequency - Sometimes labeled *speed* or *rate*, this changes the rate at which the notches move up and down in frequency. Its effect and use is similar to the flanger and chorus. Like those effects, the control sets how many times per second the notches sweep across their range. The actual speed at which the notches move (in Hz per second) will also depend on the sweep width and LFO waveform.

#### Code example

The following C++ code fragment, adapted from the code that accompanies this book, implements a phaser with feedback, a user-selectable number of allpass sections, and adjustable LFO waveform:

int numSamples; // Indicates how many audio samples to process

float \*channelData; // Array of audio samples, length numSamples

float ph; // Current phase of the LFO (0-1)

float lastFilterOutput; // Output of the filter last sample, for implementing feedback

OnePoleAllpassFilter \*\*allpassFilters; // Objects handling a first order allpass filter

float inverseSampleRate; // Defined as 1.0/(sample rate)

int sc; // Sample count, used to decide when to update the coefficients

float depth\_; // Depth of the phaser effect (0-1)

float feedback\_; // Amount of feedback (>= 0, < 1)

float lfoFrequency\_; // Frequency of the LFO

float baseFrequency\_; // Lowest point in the sweep of the allpass center frequency

float sweepWidth\_; // Width of the LFO (in Hz)

int waveform\_; // Identifier of what type of waveform to use (sine, triangle, ...)

int filtersPerChannel\_; // How many allpass filters are used

int filterUpdateInterval\_; // How often to update the allpass coefficients

for (int sample = 0; sample < numSamples; ++sample)

{

float out = channelData[sample];

// If feedback is enabled, include the feedback from the last sample in the

// input of the allpass filter chain. This is actually not accurate to how

// analog phasers work because there is a sample of delay between output and

// input, which adds a further phase shift of up to 180 degrees at half the

// sampling frequency. To truly model an analog phaser with feedback involves

// modelling a delay-free loop, which is beyond the scope of this example.

if(feedback\_ != 0.0)

out += feedback\_ \* lastFilterOutput;

for(int j = 0; j < filtersPerChannel\_; ++j)

{

// First, update the current allpass filter coefficients depending on the

// parameter settings and the LFO phase

// Recalculating the filter coefficients is much more expensive than calculating

// a sample. Only update the coefficients at a fraction of the sample rate; since

// the LFO moves slowly, the difference won't generally be audible.

if(sc % filterUpdateInterval\_ == 0)

{

allpassFilters[j]->makeAllpass(inverseSampleRate,

baseFrequency\_ + sweepWidth\_\*

lfo(ph, waveform\_));

}

out = allpassFilters[j]->processSingleSampleRaw(out);

}

lastFilterOutput = out;

// Add the allpass signal to the output, though maintaining constant level

// depth = 0 --> input only ; depth = 1 --> evenly balanced input and output

channelData[sample] = (1.0f-0.5f\*depth\_)\*channelData[sample] + 0.5f\*depth\_\*out;

// Update the LFO phase, keeping it in the range 0-1

ph += lfoFrequency\_\*inverseSampleRate;

if(ph >= 1.0)

ph -= 1.0;

sc++;

}

At the core of this code is an array C++ objects allpassFilters, which each implement a single first order allpass filter. Even if every filter has the same coefficients, we need to maintain separate objects for each filter since the filter must keep track of previous input and output samples in order to calculate its output. The code above passes each sample through each filter in succession, eventually arriving at the sample out which is mixed with the original input channelData[sample] to produce the phaser effect. The depth\_ parameter controls the relative balance of these two signals, with depth\_ = 1 producing an even mix between them and therefore the most pronounced phaser effect.

As in the earlier parametric equalizer example, it is not efficient to recalculate the filter coefficients at every single audio sample, and the LFO will change value slowly enough that recalculating once every few samples will be sufficient. The variable sc stores how many samples have elapsed since the coefficients were last recalculated, and filterUpdateInterval\_ indicates how often they should be recalculated. A value of 16 or 32 would strike an appropriate balance between efficiency and smoothness of effect.

The function lfo() implements one of several LFO waveforms depending on the value of the waveform\_ variable. waveform\_ will take one of several predefined values, for example 0 corresponding to a sine, 1 to a triangle wave, 2 to a square wave or 3 to a sawtooth wave.

The expression allpassFilters[j]->processSingleSampleRaw(out) runs the following code for each allpass filter object:

float OnePoleAllpassFilter::processSingleSampleRaw (const float sampleToProcess)

{

// Process one sample, storing the last input and output

y1 = (b0 \* sampleToProcess) + (b1 \* x1) + (a1 \* y1);

x1 = sampleToProcess;

return y1;

}

Here, x1 and y1 keep track of the last input and output x[n-1] and y[n-1] respectively. b0, b1 and a1 are coefficients of the filter, as calculated by the makeAllpass() function in the previous code block.

## Problems

1. Which one of the three diagrams in Figure 4.14 would *not* produce a working graphic equaliser, and why?

2. Suppose a graphic equalizer is implemented with six filters having an octave spacing between filters. The first (the one with lowest center frequency) filter has a lower cut-off frequency at 375Hz. Find the lower cut-off frequency, upper cut-off frequency, bandwidth and center frequency of the fourth filter.

3. i) Draw a magnitude response plot showing the frequency response of each band in a *10-band, 1-octave* graphic EQ with a lowest centre frequency of 30Hz. Assume the controls are all set to flat.

ii) Draw a block diagram of the implementation of this filter.

4. i) How do graphic equalisers differ from parametric EQs and basic tone controls?

ii) What are the primary controls in a graphic EQ?

iii) What are the primary controls in a parametric equaliser, and what do they do?

5. (difficult) In a peaking or notch filter, as used for parametric equalization, find a formula for the lower cutoff frequency as a function of bandwidth and center frequency.

6. Explain why *wah-wah* could be considered a special case of parametric equalisation. (What is the main effect of moving the pedal on a wah-wah box?)

7. Define 3 of the main parameters of a phaser; Depth, Sweep Depth, and Speed, and describe the effect of varying their settings.

8. i) How does a flanger differ from a phaser in implementation?

ii) One of the diagramsin Figure 4.15 is the frequency response of a flanger, the other is the frequency response of a phaser. Identify which is which, and explain why.

9. Explain how a phaser may be implemented using allpass filters to create notches in the frequency spectrum.



Figure .1. (a) Bass and treble tone controls implemented as a low shelving filter and high shelving filter placed in series. (b) Three tone controls, including a peaking/notch filter to adjust the midrange.



Figure .2. Frequency response of a low shelving filter (left) and a high shelving filter (right), for varying the gain (top) or varying the center frequency (bottom).



Figure .3. Magnitude responses for a peaking or notch filter when adjusting (a) gain, (b) center frequency and (c) bandwidth.



Figure .4. Equal loudness contours by phon (90 phon- top curve, 10 phon- bottom curve), as given in the ISO 226 standard. Each curve represents the Sound Pressure Level (SPL) required for which a listener will perceive a constant loudness when presented with pure steady tones across the frequency range.



Figure .5. A diagram of a graphic equalizer, implemented with band pass filters placed in parallel, with *N* bands of control.



Figure .6. General wah-wah effect block diagram.



Figure .7. Magnitude response of a resonant low pass filter, with magnitude 12 dB (*Gc* ~ 20) at *c*=**/2.



Figure 4.8. A phaser, also known as a phase shifter.



**Figure 4.9. A phaser with feedback. The feedback gain can be positive or negative, but must have a**

magnitude strictly less than 1.



Figure 4.10. A generalized diagram of a stereo phaser with feed-across gains.



Figure 4.11. Two approaches to implementing a phaser; phasing with time-varying allpass filters and optional feedback (a) and phasing with notch filters (b).



Figure .12. Allpass filter phase and phaser notch locations for different numbers of second order allpass filters. Center frequency is 1kHz and sampling frequency is 44.1kHz. a, b, c and d represent 1, 2, 3 and 4 second order sections, respectively.



Figure .13. Continuous-time (analog) and discrete-time (digital) allpass filters produce different notch locations when used in the phaser, particularly at high frequencies. Both plots depict the phase response and notch locations for four second order allpass sections and 44.1kHz sampling frequency. On left, center frequency is 3kHz and on right, center frequency is 10kHz.



Figure .. One of these three does *not* represent a graphic equalizer.



Figure .. One of these is the frequency response of a flanger, the other is the frequency response of a phaser.

# References

[1] G. Milner, *Perfecting Sound Forever: The Story of Recorded Music*: Granta Books, 2010.

[2] M. Vdovin, "Artist Interview: Brad Plunkett," *Universal Audio WebZine,* vol. 3, October 2005.

[3] M. Lewisohn, *The Beatles Recording Sessions*. New York: Harmony Books (Crown Publishers), 1989.

[4] G. Massenburg, "Parametric Equalization," in *42nd AES Convention*, 1972.

[5] G. Martin and W. Pearson, *Summer of Love: The Making of Sgt Pepper*. London: Pan Books, 1994.

[6] J. O. Smith, *Physical Audio Signal Processing*: W3K Publishing, 2010.

[7] J. O. Smith, *Spectral Audio Signal Processing*: W3K Publishing 2011.

[8] A. V. Oppenheim and R. W. Schafer, *Discrete-time signal processing*, 3rd ed.: Prentice Hall, 2009.

[9] S. J. Orfanidis, *Introduction to Signal Processing* Orfanidis (prev. Prentice Hall), 2010.

[10] U. Zolzer, *Digital Audio Signal Processing*, 2nd ed.: John Wiley and Sons, Ltd., 2008.

[11] O. Niemitalo. (2001, Oct.). Polynomial Interpolators for High-Quality Resampling of Oversampled Audio. Available: <http://yehar.com/blog/wp-content/uploads/2009/08/deip.pdf>

[12] A. M. Stark*, et al.*, "Real-Time Beat-Synchronous Audio Effects," presented at the New Interfaces for Musical Expression (NIME), New York, NY, 2007.

[13] J. D. Reiss, "Design of audio parametric equalizer filters directly in the digital domain," *IEEE Transactions on Audio, Speech and Language Processing,* vol. 19, pp. 1843-48, August 2011.

[14] D. Self, *The Design of Active Crossover*. Oxford: Focal Press, 2011.

[15] A. Case, *Mix Smart: professional techniques for the home studio*: CRC Press, 2012.

[16] R. Izhaki, *Mixing Audio: Concepts, Practices and Tools*: Focal Press, 2008.

[17] E. Perez Gonzalez and J. D. Reiss, "Automatic equalization of multi-channel audio using cross-adaptive methods," presented at the 127th AES Convention, New York, 2009.

[18] M. Holters and U. Zölzer, "Graphic Equalizer Design Using Higher-Order Recursive Filters," in *Digital Audio Effects (DAFx)*, Montreal, 2006.

[19] ISO, "ISO 266, Acoustics– Preferred frequencies for measurements," 1975.

[20] L. Bregitzer, *Secrets of Recording: Professional Tips, Tools & Techniques*: Focal Press, 2009.

[21] Z. Ma*, et al.*, "Implementation of an Intelligent Equalization Tool Using Yule-Walker for Music Mixing and Mastering," in *134th AES Convention*, Rome, 2013.

[22] A. Loscos and T. Aussenac, "The wahwactor: a voice controlled wah-wah pedal," in *New interfaces for musical expression (NIME)*, 2005, pp. 172–175.

[23] R. G. Keen. (1999, Sept.). Technology of Wah Pedals. Available: [www.geofex.com/article\_folders/wahpedl/wahped.htm](http://www.geofex.com/article_folders/wahpedl/wahped.htm)

[24] R. Parncutt, "A Perceptual Model of Pulse Salience and Metrical Accent in Musical Rhythms," *Music Perception,* vol. 11, Summer 1994.

[25] H. Fastl and E. Zwicker, *Psychoacoustics: facts and models*, 3rd ed.: Springer, 2007.

[26] S. Rosen, "Temporal Information in Speech: Acoustic, Auditory and Linguistic Aspects," *Philosophical Transactions of the Royal Society B,* vol. 336, June 1992.

[27] B. Lachaise and L. Daudet, "Inverting dynamics compression with minimal side information," in *Digital Audio Effects Workshop (DAFx)*, Helsinki, 2008.

[28] D. Giannoulis*, et al.*, "A Tutorial on Digital Dynamic Range Compressor Design," *J. Audio Eng. Soc.,* vol. 60, June 2012.

[29] B. Rudolf, "1176 Revision History," *Mix Magazine Online,* June 1 2000.

[30] D. Barchiesi and J. D. Reiss, "Reverse Engineering the Mix," *Journal of the Audio Engineering Society,* vol. 58, pp. 563-576, July 2010.

[31] R. J. Cassidy, "Level Detection Tunings and Techniques for the Dynamic Range Compression of Audio Signals," in *117th AES Convention*, 2004.

[32] J. S. Abel and D. P. Berners, "On peak-detecting and rms feedback and feedforward compressors," in *115th AES Convention*, 2003.

[33] P. Dutilleux*, et al.*, "Nonlinear Processing, Chap. 4," in *Dafx:Digital Audio Effects*, U. Zoelzer, Ed., 2nd ed: Wiley, John & Sons, 2011, p. 554.

[34] J. Bitzer and D. Schmidt, "Parameter Estimation of Dynamic Range Compressors: Models, Procedures and Test Signals," presented at the 120th AES Convention, 2006.

[35] L. Lu, "A digital realization of audio dynamic range control," in *Fourth International Conference on Signal Processing Proceedings (IEEE ICSP)*, 1998, pp. 1424 - 1427.

[36] Sonnox, "Dynamics plug-in manual," *Sonnox Oxford Plug-ins,* April 1st 2007.

[37] M. Zaunschirm*, et al.*, "A sub-band approach to musical transient modification," *Computer Music Journal,* vol. 36, Summer 2012.

[38] M. Senior, *Mixing Secrets for the small studio*: Focal Press, 2011.

[39] D. Gibson, *The Art of Mixing*, 2nd ed.: artistpro.com LLC, 2005.

[40] M. J. Terrell*, et al.*, "Automatic Noise Gate Settings for Drum Recordings Containing Bleed from Secondary Sources," *EURASIP Journal on Advances in Signal Processing,* vol. 2010, pp. 1-9, 2010.

[41] S. Gorlow and J. D. Reiss, "Model-Based Inversion of Dynamic Range Compression," *IEEE Trans. Audio Speech Lang. Proc.,* vol. 21, 2013.

[42] B. De Man and J. D. Reiss, "An Intelligent Multiband Distortion Effect," presented at the AES 53rd International Conference on Semantic Audio, London, UK, 2014.

[43] D. Yeh*, et al.*, "Simulation of the diode limiter in guitar distortion circuits by numerical solution of ordinary differential equations," presented at the Digital Audio Effects (DAFX), 2007.

[44] V. Välimäki*, et al.*, "Virtual Analog Effects, Chap. 12," in *Dafx:Digital Audio Effects*, 2nd ed: Wiley, John & Sons, 2011.

[45] D. Yeh*, et al.*, "Numerical methods for simulation of guitar distortion circuits," *Computer Music Journal,* vol. 32, pp. 23-42, 2008.

[46] J. Pakarinen and D. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music Journal,* vol. 33, pp. 85-100, 2009.

[47] J. Pakarinen and M. Karjalainen, "Enhanced wave digital triode model for real-time tube amplifier emulation," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 18, pp. 738-746, 2010.

[48] W. M. J. Leach, "SPICE models for vacuum-tube amplifiers," *Journal of the Audio Engineering Society,* vol. 43, pp. 117-126, 1995.

[49] H. Shapiro and C. Glebbeek, *Jimi Hendrix: Electric Gypsy (New and Improved ed.)*: St. Martin's Press, 1995.

[50] J. L. Flanagan and R. M. Golden, "Phase Vocoder," *Bell System Technical Journal,* pp. 1493-1509, Nov. 1966.

[51] M. Portnoff, "Implementation of the digital phase vocoder using the fast Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 24, pp. 243- 248, 1976.

[52] J. Laroche and M. Dolson, "Improved phase vocoder timescale modification of audio," *IEEE Trans. Speech and Audio Processing,* vol. 7, pp. 323 -332, 1999.

[53] M. Dolson, "The phase vocoder: A tutorial " *Computer Music Journal,* vol. 10, pp. 14-27, 1986.

[54] J. Vilkamo*, et al.*, "Directional Audio Coding: Virtual microphone-based synthesis and subjective evaluation," *Journal of the Audio Engineering Society,* vol. 57, 2009.

[55] D. Beheng, "Sound Perception," Deutche Welle, Radio Training Centre (DWRTC), Cologne 2002.

[56] B. Bernfeld, "Attempts for Better Understanding of the Directional Stereophonic Listening Mechanism," presented at the 44th Audio Engineering Society Convention, 1973.

[57] V. Pulkki, "Virtual Sound Source Positioning Using Vector Base Amplitude Panning," *J. Audio Eng. Soc.,* vol. 45, 1997.

[58] D. G. Malham and A. Myatt, "3-d sound spatialization using ambisonic techniques," *Computer Music Journal,* vol. 19, pp. 58-70, 1995.

[59] J. S. Bamford, "An Analysis of Ambisonics Sound Systems of First and Second Order," MSc, University of Waterloo, Waterloo, Ont., Canada, 1995.

[60] M. A. Poletti, "Three-Dimensional Surround Sound Systems Based on Spherical Harmonics," *J. Audio Eng. Soc.,* vol. 53, November 2005

[61] E. G. Williams, *Fourier Acoustics: Sound Radiation and Nearfield Acoustical Holography*. London: Academic Press, 1999.

[62] P. M. Morse and K. U. Ingard, *Theoretical Acoustics*. New York: McGraw-Hill, 1968.

[63] M. Gerzon, "Design of Ambisonic Decoders for Multispeaker Surround Sound," in *58th Audio Egineering Society Convention*, 1977.

[64] E. N. G. Verheijen, "Sound Reproduction by Wave Field Synthesis," PhD, Delft University of Technology, 1998.

[65] J. Blauert, *Spatial Hearing: The Psychophysics of Human Sound Localization*: MIT Press, 1997.

[66] F. Rumsey, *Spatial Audio*: Focal Press, 2001.

[67] J. W. Strutt, "On our perception of sound direction," *Philos. Mag.,* vol. 13, pp. 214-232, 1907.

[68] V. R. Algazi*, et al.*, "Motion-Tracked Binaural Sound," *J. Aud. Eng. Soc.,* vol. 52, November 2004.

[69] C. P. Brown and R. O. Duda, "A Structural Model for Binaural Sound Synthesis," *IEEE Trans. Speech and Audio Processing,* vol. 6, Sept. 1998.

[70] R. O. Duda, "3-D Audio for HCI," 2000.

[71] V. Pulkki, "Spatial Sound Generation and Perception by Amplitude Panning Techniques," Electrical and Communications Engineering, Helsinki University of Technology, Helsinki, Finland, 2001.

[72] D. d. Vries*, et al.*, "The wave field synthesis concept applied to sound reinforcement: restrictions and solutions," presented at the 96th Convention of the Audio Engineering Society, Amsterdam, 1994.

[73] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," presented at the 108th AES Convention, 2000.

[74] A. Pierce, *Acoustics: an introduction to its physical principles and applications*: American Institute of Physics, 1989.

[75] J. O. Smith, "Doppler simulation and the Leslie," in *5th Intl. Conf. on Digital Audio Effects (DafX)*, Hamburg, 2002.

[76] B. G. Quinn, "Doppler speed and range estimation using frequency and amplitude estimates," *J. Acoust. Soc. Am.,* vol. 98, pp. 2560-2566, 1995.

[77] A. R. Gondeck, "Doppler time mapping," *J. Acoust. Soc. Am.,* vol. 73, pp. 1863-1864, 1983.

[78] D. Rocchesso, "Fractionally addressed delay lines," *IEEE Transactions on Speech and Audio Processing,* vol. 8, November 2000.

[79] L. L. Beranek, "Analysis of Sabine and Eyring equations and their application to concert hall audience and chair absorption," *J. Acoust. Soc. Am.,* vol. 120, Sept. 2006.

[80] D. Howard and J. Angus, *Acoustics and Psychoacoustics*, 4th ed.: Focal Press, 2009.

[81] D. Davis and E. Patronis, *Sound System Engineering*, Third ed.: Elsevier, 2006.

[82] D. Davis, "A More Accurate Way Of Calculating Critical Distance," Altec TL-207, 1971.

[83] M. R. Schroeder and B. F. Logan, "Colorless Artificial Reverberation," *Journal of the Audio Engineering Society,* vol. 9, 1961.

[84] M. R. Schroeder, "Natural Sounding Artificial Reverberation," *Journal of the Audio Engineering Society,* vol. 10, 1962.

[85] J. A. Moorer, " About this reverberation business," *Computer Music Journal,* vol. 3, pp. 13-18, 1979.

[86] J. Allen and D. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Am.,* vol. 65, pp. 943–950, 1979.

[87] E. Lehmann and A. Johansson, "Prediction of energy decay in room impulse responses simulated with an image-source model," *Journal of the Acoustical Society of America,* vol. 124, July 2008.

[88] P. Peterson, "Simulating the response of multiple microphones to a single acoustic source in a reverberant room," *Journal of the Acoustical Society of America,* vol. 80, November 1986.

[89] W. G. Gardner, "Efficient Convolution without Input-Output Delay," in *97th Convention of the Audio Engineering Society*, San Francisco, CA, 1994.

[90] D. M. Huber and R. E. Runstein, *Modern Recording Techniques*, seventh ed.: Focal Press, 2010.

[91] D. K. Wise, "Concept, Design, and Implementation of a General Dynamic Parametric Equalizer," *Journal of the Audio Engineering Society,* vol. 57, pp. 16-28, Jan. 2009.

[92] P. Pestana, "Automatic Mixing Systems Using Adaptive Audio Effects," PhD, Universidade Catolica Portuguesa, Lisbon, Portugal, 2013.

[93] P. D. Pestana and J. D. Reiss, "Intelligent Audio Production Strategies Informed by Best Practices," presented at the AES 53rd International Conference on Semantic Audio, London, UK, 2014.

[94] B. Stroustrup, *The C++ Programming Language*: Addison Wesley, 2000.

[95] A. Koenig and B. E. Moo, *Accelerated C++*: Addison Wesley, 2000.

[96] S. B. Lippman*, et al.*, *C++ Primer*, 5th ed.: Addison Wesley, 2012.

[97] R. Boulanger*, et al.*, *The Audio Programming Book*: MIT Press, 2010.

[98] M. Walmsley, *Multi-Threaded Programming in C++*: Springer 1999.