Introduction to Computer Music

Week 7

Instructor: Prof. Roger B. Dannenberg

Topics Discussed: Samples, Filters, High-pass, Low-pass, Band-pass

Chapter 7

Sampling and Filters

Topics Discussed: Samples, Filters, High-pass, Low-pass, Band-pass

7.1 Sampling Synthesis

Although direct synthesis of sound is an important area of computer music, it can be equally interesting (or even more so!) to take existing sounds (recorded or synthesized) and transform, mutate, deconstruct—in general, mess around with them. There are as many basic sound transformation techniques as there are synthesis techniques, and in this chapter we'll describe a few important ones. For the sake of convenience, we will separate them into time-domain and frequency-domain techniques.

7.1.1 Tape Music / Cut and Paste

The most obvious way to transform a sound is to chop it up, edit it, turn it around, and collage it. These kinds of procedures, which have been used in electronic music since the early days of tape recorders, are done in the time domain. There are a number of sophisticated software tools for manipulating sounds in this way, called *digital sound editors*. These editors allow for the most minute, sample-by-sample changes of a sound. These techniques are used in almost all fields of audio, from avant-garde music to Hollywood soundtracks, from radio station advertising spots to rap.

7.1.2 Time-Domain Restructuring

Composers have experimented a lot with unusual time-domain restructuring of sound. By chopping up waveforms into very small segments and radically reordering them, some noisy and unusual effects can be created. As in collage visual art, the ironic and interesting juxtaposition of very familiar materials can be used to create new works that are perhaps greater than the sum of their constituent parts.

Unique, experimental, and rather strange programs for deconstructing and reconstructing sounds in the time domain are Herbert Brün's SAWDUST see Figure 7.1 and Argeïphontes Lyre (see Figure 7.2, written by the enigmatic Akira Rabelais. Argeïphontes Lyre provides a number of techniques for radical decomposition/recomposition of sounds—techniques that often preclude the user from making specific decisions in favor of larger, more probabilistic decisions.

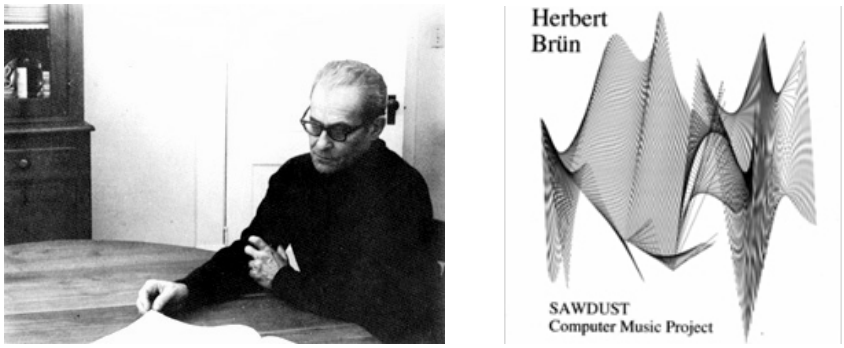

Figure 7.1: Herbert Brün said of his program SAWDUST: "The computer program which I called SAWDUST allows me to work with the smallest parts of waveforms, to link them and to mingle or merge them with one another. Once composed, the links and mixtures are treated, by repetition, as periods, or by various degrees of continuous change, as passing moments of orientation in a process of transformations."

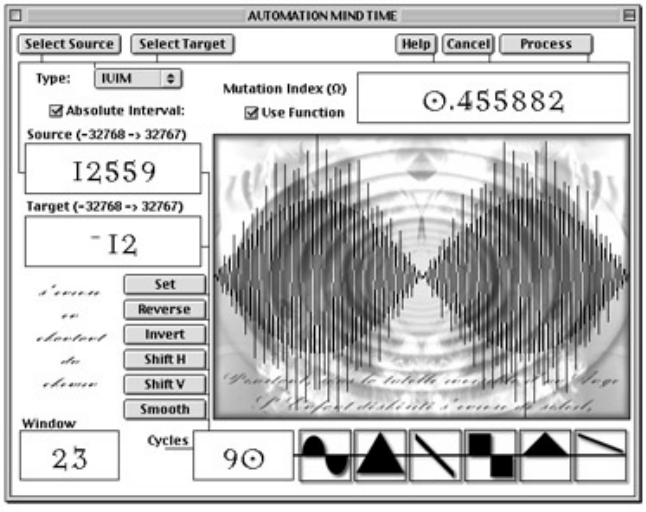

Figure 7.2: Sample GUI from Argeïphontes Lyre, for sound deconstruction. This is time-domain mutation.

7.1.3 Sampling

Sampling refers to taking small bits of sound, often recognizable ones, and recontextualizing them via digital techniques. By digitally sampling, we can easily manipulate the pitch and time characteristics of ordinary sounds and use them in any number of ways.

We've talked a lot about samples and sampling in the preceding chapters. In popular music (especially electronic dance and beat-oriented music), the term *sampling* has acquired a specialized meaning. In this context, a sample refers to a (usually) short excerpt from some previously recorded source, such as a drum loop from a song or some dialog from a film soundtrack, that is used as an element in a new work. A *sampler* is the hardware used to record, store, manipulate, and play back samples. Originally, most samplers were stand-alone pieces of gear. Today sampling tends to be integrated into a studio's computer-based digital audio system.

Sampling was pioneered by rap artists in the mid-1980s, and by the early 1990s it had become a standard studio technique used in virtually all types of music. Issues of copyright violation have plagued many artists working with sample-based music, notably John Oswald of "Plunderphonics" fame and the band Negativland, although the motives of the "offended" parties (generally large record companies) have tended to be more financial than artistic. One result of this is that the phrase "Contains a sample from xxx, used by permission" has become ubiquitous on CD cases and in liner notes.

Although the idea of using excerpts from various sources in a new work is not new (many composers, from Béla Bartók, who used Balkan folk songs, to Charles Ives, who used American popular music folk songs, have done so), digital technology has radically changed the possibilities.

7.1.4 Drum Machines

Drum machines and samplers are close cousins. Many drum machines are just specialized samplers—their samples just happen to be all percussion/drum-oriented. Other drum machines feature electronic or digitally synthesized drum sounds. As with sampling, drum machines started out as stand-alone pieces of hardware but now have largely been integrated into computer-based systems.

7.1.5 DAW Systems

Digital-audio workstations (DAWs) since the 1990s have had the same effect on digital sound creation as desktop publishing software had on the publishing industry in the 1980s: they've brought digital sound creation out of the highly specialized and expensive environments in which it grew up and into people's homes. A DAW usually consists of a computer with some sort of sound card or other hardware for analog and digital input/output; sound recording/editing/playback/multitrack software; and a mixer, amplifier, and other sound equipment traditionally found in a home studio. Even the most modest of DAW systems can provide sixteen or more tracks of CD-quality sound, making it possible for many artists to self-produce and release their work for much less than it would have cost in professional analog recording studios. This ability, in conjunction with similar marketing and publicizing possibilities opened up by the spread of the Internet, has contributed to the explosion of tiny record labels and independent releases we've seen.

In 1989, Digidesign came out with Sound Tools, the first professional tape–less recording system. With the popularization of personal computers, numerous software and hardware manufacturers developed products for computer-based digital audio production. Becoming affordable and practical in the 1990s, personal computer–based production systems have allowed individuals and institutions to make the highest-quality recordings. DAW systems have also revolutionized the professional music, broadcast, multimedia, and film industries with audio systems that are more flexible, more accessible, and more creatively oriented than ever before.

Today, DAWs come in all shapes and sizes on hardware ranging from mobile phones to desktop computeres. Most DAWs can take advantage of an *audio interface*—a piece of hardware that provides two to 16 or more channels of digital and analog audio I/O (inputs and outputs). Many DAW systems also have *control surfaces*—pieces of hardware that look like a standard mixer but are really controllers for parameters in the digital audio system.

7.1.6 Sampling as a Synthesis Technique

Sampling is often used to produce musical tones or notes, and some think of sampling as the opposite of synthesis. In this view, *sampling* is based on recordings while *synthesis* produces sound by mathematical calculation. However, sample-based tone production involves considerable calculation and allows for parametric control, so we will describe it as yet another synthesis technique or algorithm.

The main attraction of sampling is that recordings of actual acoustic instruments generally sound realistic. (You might think samples by definition must *always* sound realistic, but this is an overstatement we will discuss later.) There are three basic problems with samples:

1. Samples of musical tones have definite pitches. You could collect samples for every pitch, but this requires a lot of storage, so most samplers need the capability of changing the pitch of samples through calculation.

2. Samples of musical tones have definite durations. In practice, we need to control the duration of tones, and capturing *every* possible duration is impractical, so we need the capability of changing the duration of samples.

3. Samples have a definite loudness. In practice we need to produce sounds from soft to loud.

Pitch Control

The standard approach to pitch control over samples (here, *sample* means a short recording of a musical tone) is to change the sample rate using interpolation. If we *upsample*, or increase the sample rate, but play at the normal sample rate, the effect will be to slow down and drop the pitch of the original recording. Similarly, if we *downsample*, or lower the sample rate, but play at the normal rate, the effect will be to speed up and raise the pitch of the original recording.

To change sample rates, we interpolate existing samples at new time points. Often, "interpolation" means *linear* interpolation, but with signals, linear interpolation does not result in the desired smooth band-limited signal, which is another way of saying that linear interpolation will distort the signal; it may be a rough approximation to proper reconstruction, but it is not good enough for high-quality audio. Instead, good audio sample interpolation requires a weighted sum taken over a number of samples. The weights follow an interesting curve as shown in Figure 7.3. More samples lead to better quality, but also more computation.

In early samplers, where even specialized hardware had limited performance, interpolation with less than 10 points was probably typical, but details are proprietary. Modern processors are fast enough to achieve high-quality interpolation with many more points (50 to 200 does not seem to be unusual). The specific case of transposing samples *down* and other assumptions can lead to more relaxed requirements.

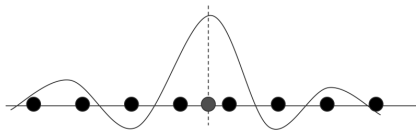

Figure 7.3: To resample a signal, we need to interpolate between existing samples in order to *reconstruct* the signal at a new time. A sequence of these new samples at a narrower or wider spacing creates a representation of the signal at a new sample rate. For each of these new samples (in this figure, the new sample will be at the dotted line), we form a weighted sum of existing samples. The weights are given by the so-called sinc function shown here (the thin wavy line). This weighted sum is essentially a low-pass filter that does in the digital domain what the reconstruction filter does in the analog domain when converting a digital signal to analog.

Another problem with resampling is that when pitch is shifted by a lot, the quality of sound is significantly changed. For example, voices shifted to higher pitches begin so sound like cartoon voices or people breathing helium. This is unacceptable, so usually, there are many samples for a given instrument. For example, one might use four different pitches for each octave. Then, pitch shifts are limited to the range of $1/4$ octave, eliminating the need for extreme pitch shifts.

Duration Control

A simple way to control duration is simply to multiply samples by an envelope to end them as early as needed. However, this could require a considerable amount of memory to store long raw samples in case long notes are required. To save memory, samplers can *loop* or repeat portions of samples to sustain the sound. Usually, there is an initial *attack* portion of the sample at the beginning, and when the sample settles down to a periodic waveform, the sampler can repeat that portion of the sample. To loop samples, one must be very careful to find points where the end of the loop flows smoothly into the beginning of the loop to avoid discontinuities that impart a buzzy sound to the loop. Looping is a painstaking manual process, which is another reason to simply use longer samples as long as the added memory requirements are acceptable.

Loudness Control

A simple way to produce soft and loud versions of samples is to just multiply the sample by a scale factor. There is a difference, however, between amplitude and *loudness* in that many instruments change their sound when they play louder. Imagine a whisper in your ear vs. a person shouting from 100 yards away. Which is louder? Which has higher amplitude?

To produce variations in perceptual loudness, we can use filters to alter the spectrum of the sound. E.g. for a softer sound, we might filter out higher fre-

quencies as well as reduce the amplitude by scaling. Another option is to record samples for different amplitudes and select samples according go the desired loudness level. Amplitude changes can be used to introduce finer adjustments.

Sampling in Nyquist

In Nyquist, the `sampler` function can be used to implement sampling synthesis. The signature of the function is:

`sampler(`*pitch*`,` *modulation*`,` *sample*`)`

The *pitch* parameter gives the desired output pitch, and *modulation* is a frequency deviation function you can use to implement vibrato or other modulation. The output duration is controlled by *modulation*, so if you want zero modulation and a duration of *dur*, use `const(0, dur)` for *modulation*. The *sample* parameter is a list of the form: (*sound pitch loop-start*), where *sound* is the sample (audio), *pitch* is the natural pitch of the sound, and *loop-start* is the time at which the loop starts. Resampling is performed to convert the natural pitch of the sample to the desired pitch computed from the *pitch* parameter and *modulation*. The `sampler` function returns a sound constructed by reading the audio sample from beginning to end and then splicing on copies of the same sound from the loop point (given by *loop-start*) to the end. Currently, only 2-point (linear) interpolation is implemented, so the quality is not high. However, you can use `resample` to upsample your *sound* to a higher sample rate, e.g. 4x and this will improve the resulting sound quality. Note also that the loop point may be fractional, which may help to make clean loops.

Summary

Sampling synthesis combines pre-recorded sounds with signal processing to adjust pitch, duration, and loudness. For creating music that sounds like acoustic instruments, sampling is the currently dominant approach. One of the limitations of sampling is the synthesis of lyrical musical phrases. While samples are great for reproducing single notes, acoustic instruments perform connected phrases, and the transitions between notes in phrases are very difficult to implement with sampling. Lyrical playing by wind instruments and bowed strings is very difficult to synthesize simply by combining 1-note samples. Some sample libraries even contain transitions to address this problem, but the idea of sampling all combinations of articulations, pitches, durations, and loudness levels and their transitions does not seem feasible.

7.2 Filters

The most common way to think about filters is as functions that take in a signal and give back some sort of transformed signal. Usually, what comes out is less than what goes in. That's why the use of filters is sometimes referred to as *subtractive synthesis*.

It probably won't surprise you to learn that *subtractive synthesis* is in many ways the opposite of additive synthesis. In additive synthesis, we start with simple sounds and add them together to form more complex ones. In subtractive synthesis, we start with a complex sound (such as noise or a rich harmonic spectrum) and subtract, or filter out, parts of it. Subtractive synthesis can be thought of as sound sculpting—you start out with a thick chunk of sound containing many

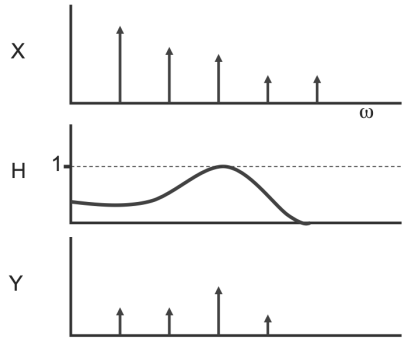

Figure 7.4: Filtering in the frequency domain. The input signal with spectrum X is multiplied by the frequency response of the filter H to obtain the output spectrum Y. The result of filtering is to reduce the amplitude of some frequencies and boost the amplitudes of others. In addition to amplitude changes, filters usually cause phase shifts that are also frequency dependent. Since we are so insensitive to phase, we usually ignore the *phase response* of filters.

possibilities (frequencies), and then you carve out (filter) parts of it. Filters are one of the sound sculptor's most versatile and valued tools.

The action of filters is best explained in the frequency domain. Figure 7.4 explains the action of a filter on the spectrum of a signal. In terms of the magnitude spectrum, filtering is a *multiplication* operation. Essentially, the filter *scales* the amplitude of each sinusoidal component by a frequency-dependent factor. The overall frequency-dependent scaling function is called the *frequency response* of the filter.

A common misconception among students is that since filtering is multiplication in the frequency domain, and since the FFT converts signals to the frequency domain (and the inverse FFT converts back), then filtering must be performed by an FFT-multiply-IFFT sequence. *This is false!* At least simple filters operate in the time domain (details below). Filtering with FFTs is possible but problematic because it is not practical to perform FFTs on long signals.

7.2.1 Four Basic Types of Filters

Figure 7.5 illustrates four basic types of filters: low-pass, high-pass, band-pass, and band-reject. Low-pass and high-pass filters should already be familiar to you—they are exactly like the tone knobs on a car stereo or boombox. A low-pass (also known as high-stop) filter stops, or attenuates, high frequencies while letting through low ones, while a high-pass (low-stop) filter does just the opposite.

Band-Pass and Band-Reject Filters

Band-pass and band-reject filters are basically combinations of low-pass and high-pass filters. A *band-pass* filter lets through only frequencies above a certain point and below another, so there is a *band* of frequencies that get through. A *band-reject* filter is the opposite: it stops a band of frequencies. Band-reject filters are sometimes called *notch filters*, because of the shape of their frequency response.

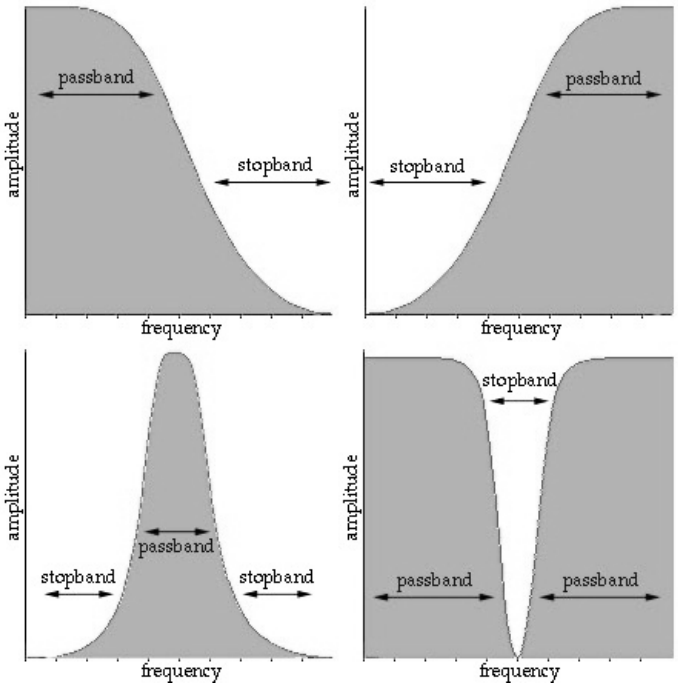

Figure 7.5: Four common filter types (clockwise from upper left): low-pass, high-pass, band-reject, band-pass.

Low-Pass and High-Pass Filters

Low-pass and high-pass filters have a value associated with them called the *cutoff frequency*, which is the frequency where they begin "doing their thing." So far we have been talking about *ideal*, or perfect, filters, which cut off instantly at their cutoff frequency. However, real filters are not perfect, and they can't just stop all frequencies at a certain point. Instead, frequencies die out according to a sort of curve around the corner of their cutoff frequency. Thus, the filters in Figure 3 don't have right angles at the cutoff frequencies—instead they show general, more or less realistic response curves for low-pass and high-pass filters.

7.2.2 Cutoff Frequency

The cutoff frequency of a filter is defined as the point at which the signal is attenuated to 0.707 of its maximum value (which is 1.0). No, the number 0.707 was not just picked out of a hat! It turns out that the power of a signal is determined by squaring the amplitude: $0.707^2 = 0.5$. So when the amplitude of a signal is at 0.707 of its maximum value, it is at half-power. The cutoff frequency of a filter is sometimes called its *half-power point*.

7.2.3 Transition Band

The area between where a filter "turns the corner" and where it "hits the bottom" is called the *transition band*. The steepness of the slope in the transition band is important in defining the sound of a particular filter. If the slope is very steep, the filter is said to be "sharp;" conversely, if the slope is more gradual, the filter is "soft" or "gentle."

Things really get interesting when you start combining low-pass and high-pass filters to form band-pass and band-reject filters. Band-pass and band-reject filters

also have transition bands and slopes, but they have two of them: one on each side. The area in the middle, where frequencies are either passed or stopped, is called the *passband* or the *stopband*. The frequency in the middle of the band is called the *center frequency*, and the width of the band is called the filter's *bandwidth.*

You can plainly see that filters can get pretty complicated, even these simple ones. By varying all these parameters (cutoff frequencies, slopes, bandwidths, etc.), we can create an enormous variety of subtractive synthetic timbres.

7.2.4 A Little More Technical: IIR and FIR Filters

Filters are often talked about as being one of two types: finite impulse response (FIR) and infinite impulse response (IIR). This sounds complicated (and can be!), so we'll just try to give a simple explanation as to the general idea of these kinds of filters.

Finite impulse response filters are those in which delays are used along with some sort of scaling. *Delays* mean that the sound that comes out at a given time uses some of the previous samples. They've been delayed and scaled before they get used.

We've talked about these filters earlier. FIRs tend to be simpler, easy to use, and easier to design than IIRs, and they are very handy for a lot of simple situations. An averaging low-pass filter, in which some number of samples are averaged and output, is an example of an FIR. A key property of FIR filters is that the output goes to zero after the maximum delay used in the filter, so the so-called filter response is finite in length.

Infinite impulse response filters are a little more complicated, because they have an added feature: feedback. You've all probably seen how a microphone and speaker can have feedback: by placing the microphone in front of a speaker, you amplify what comes out and then stick it back into the system, which is amplifying what comes in, creating a sort of infinite amplification loop. Ouch! (If you're Jimi Hendrix, you can control this and make great music out of it.)

Well, IIRs are similar. Because of the feedback path, these filters can be unstable, just like feedback through a microphone, but there are filter design techniques to insure stability, so that's not a problem in practice. The feedback allows IIR filters to have an infinite response time; their output can continue to decay long after the input signal becomes zero, depending on the design.

In general, IIR filters have fewer coefficients (delays and multiplies) meaning lower computation cost for a given task, but FIR filters make it easier to achieve an arbitrary frequency response, and FIR filters can have *linear phase*, which means that all frequencies are delayed by the same amount. This is not the case with IIR filters.

Designing filters is a difficult but key activity in the field of *digital signal processing*, a rich area of study that is well beyond the range of this book. It is interesting to point out that, surprisingly, even though filters change the frequency content of a signal, a lot of the mathematical work done in filter design is done in the time domain, not in the frequency domain. By using things like sample averaging, delays, and feedback, one can create an extraordinarily rich variety of digital filters.

For example, the following is a simple equation for a low-pass filter. This equation just averages the last two samples of a signal (where $x[n]$ is the current sample) to produce a new sample. This equation is said to have a *one-sample delay*. You can see easily that quickly changing (that is, high-frequency) time

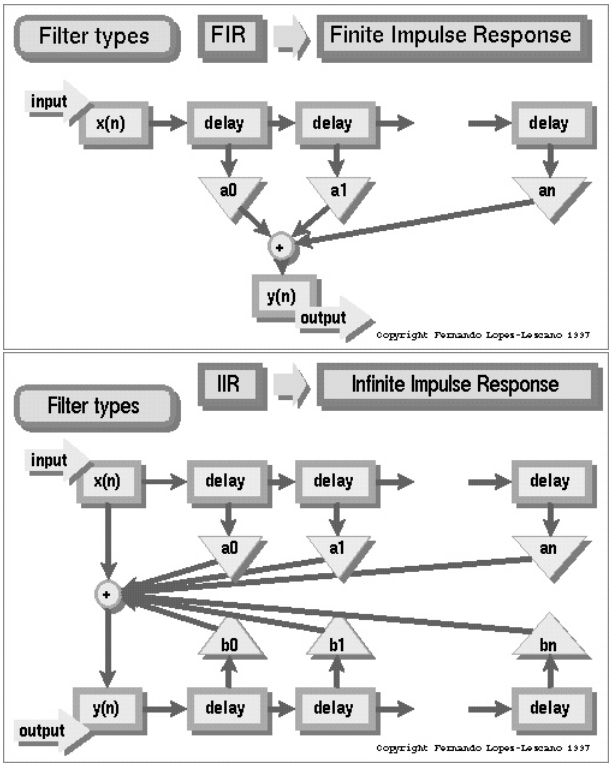

Figure 7.6: *FIR* and *IIR* filters. Filters are usually designed in the time domain, by delaying a signal and then averaging (in a wide variety of ways) the delayed signal and the nondelayed one. These are called *finite impulse response* (FIR) filters, because what comes out uses a finite number of samples, and a sample only has a finite effect.

If we delay, average, and then feed the output of that process back into the signal, we create what are called *infinite impulse response* (IIR) filters. The feedback process actually allows the output to be much longer than the input.

These diagrams are technical lingo for typical filter diagrams for FIR and IIR filters. Note how in the IIR diagram the output of the filter's delay is summed back into the input, causing the infinite response characteristic. That's the main difference between the two filters.

domain values will be "smoothed" (removed) by this equation.

$$x[n] = (x[n] + x[n-1])/2 \tag{7.1}$$

In fact, although it may look simple, this kind of filter design can be quite difficult (although extremely important). How do you know which frequencies you're removing? It's not intuitive, unless you're well schooled in digital signal processing and filter theory and have some background in mathematics. The theory introduces another transform, the Laplace transform and its discrete cousin, the Z-transform, which, along with some remarkable geometry, yields a systematic approach to filter design.

7.2.5 Filters in Nyquist

Fortunately, you do not need to know filter design theory to use filters in Nyquist. Nyquist has an assortment of filters that will work for most purposes. The following expression applies a low-pass filter to `signal` with a cutoff frequency of `cutoff`:

```
lp(signal, cutoff)
```

Note that `cutoff` can also be a signal, allowing you to adjust the cutoff frequency over the course of the signal. Normally, `cutoff` would be a control-rate signal computed as an envelope or `pwl` function. Changing the filter cutoff frequency involves some trig functions, and Nyquist filters perform these updates at the sample rate of the control signal, so if the control signal sample rate is low (typically 1/20 of the audio sample rate), you save a lot of expensive computation. All of these built-in filters are IIR filters. For those able to design their own FIR filters, Nyquist has a convolution function to implement them.

Some other filters in Nyquist include:

- `hp` - a high-pass filter.
- `reson` - a low-pass filter with a resonance (emphasis) at the cutoff frequency and a *bandwidth* parameter to control how narrow is the resonant peak.
- `areson` - the opposite of `reson`; adding the two together recreates the original signal, so `areson` is a sort of high-pass filter with a notch at the cutoff frequency.
- `comb` - a comb filter has a set of harmonically spaced resonances and can be very interesting when applied to noisy signals.
- `eq-lowshelf`, `eq-highshelf` and `eq-band` filters are useful for generally shaping spectra the way you would with a parametric equalizer.
- `lowpass2`, `lowpass4`, `lowpass6`, `lowpass8`, `highpass2`, `highpass4`, `highpass6` and `highpass8` are optimized lowpass and highpass filters with different transition slopes. The digits refer to the *order* of the filter: second order filters have a 12 dB-per-octave slope, fourth have 24 dB-per-octave slopes, etc.

All of these are described in greater detail in the Nyquist Reference Manual.

7.2.6 Filter Summary

Filters are essential building blocks for sound synthesis and manipulation. The effect of a filter in the frequency domain is to multiply the spectrum by a *filter response*. Most filters are actually implemented in the *time domain* using a discrete sample-by-sample process applied to a stream of samples, which eliminates the problems of dealing with finite-sized FFT frames, and is also usually faster than performing FFTs.