# Chapter 7 Practical Exercise

## Creating Your Own Convolution Reverb

Applying a convolution reverb to an audio signal is pretty straightforward. As explained in Chapter 7, all you need is an audio signal and an impulse response (IR) signal. Then you can convolve those together using MATLAB's *conv* or one of the filtering functions. Alternatively, you can transform the signal and the IR into the frequency domain using an FFT and multiply their responses together, transforming thee output back into a playable time domain signal. However, before you can apply any convolutions, you'll need to have an impulse response to use. In this exercise, we describe how to create your own convolution reverb – aka, impulse response.

There are a number of ways to create and capture an impulse response. If you're testing an electronic system, such as a piece of audio hardware or software, you can simply send a short electronic pulse through it, and capture what comes out the other side. You can, for example, capture the impulse response of your sound card by generating a short pulse in MATLAB, Audacity, or the like, and sending it out of your sound card through an audio cable that goes back into an input of your sound card to record it back into software. Of course, your sound card is designed to have a very clean response, so your IR won't be very interesting. An interesting experiment, however, would be to play that pulse through an EQ plugin, set to some particular filtering, before sending it out of your sound card. The pulse you capture on the way back in will be an IR with the response of that EQ filter. If you then take that IR and convolve it with an audio signal, you should get the same effect as if you sent the audio signal through the EQ itself. You can even compare your convolved results with an actual equalized signal in your DAW. You can try this experiment with an artificial reverb plug-in as well.

If you want to capture the impulse response of a physical acoustical space or system, you'll need a more robust method. To capture a decent impulse response, you need to excite all of the audible frequencies in an acoustical space. There are a number of ways to do this, but the most widely used and effective method is using a swept sinusoid signal. A **swept sine** is a pure tone signal whose frequency varies over time, typically sweeping across the audible spectrum over some specified duration. We've discussed that playing a short electronic pulse (impulse) through a system results in an impulse response. Clearly, a swept sine signal is not an impulse, so simply passing it through a system is not going to output an IR in the same way. With a little mathematics, however, we can retrieve the IR. We know that a filtered output signal is achieved by convolving a known input signal with a filter's impulse response (or multiplying them in the frequency domain). In this case, if we capture the output of the swept sine wave played in an acoustic environment (our "filter"), we

This material is based on work supported by the National Science Foundation under CCLI Grant DUE 0717743, Jennifer Burg PI, Jason Romney, Co-PI.

know both the output (what we captured) as well as the input (our swept sine signal). Thus, we can obtain the impulse response by deconvolving the output with the input, as expressed in the definition below.

If

$$\boldsymbol{g} = \boldsymbol{f} \otimes \boldsymbol{h}$$

then

$$\boldsymbol{h} = \ \boldsymbol{g} \ \otimes^{-1} \ \boldsymbol{f}$$

where $\otimes$ is the convolution operator

and $\otimes^{-1}$ is the deconvolution operator.

In this definition, $\boldsymbol{f}$ is the swept sine wave, $\boldsymbol{g}$ is the sound we record when we play the swept wave in a chosen acoustical space, and $\boldsymbol{h}$ is the filter we seek to apply to other signals so that they sound like they are played in the same acoustical space.

Note that we can also perform this operation in the frequency domain. This is accomplished by dividing the output frequency response by the input frequency response, and performing the inverse FFT to get back the time domain IR.

Typically, to get the best result when capturing an IR, you'll want to employ a good, flat loudspeaker to play back the swept sine, as well as precise measurement microphones to capture it. Of course, feel free to use what you have available in order to try this out. (Omnidirectional mics tend to work best.) The impulse response will still be effective to some degree. The first thing you need to do is generate an appropriate swept sine signal to be used as the input to your acoustical system. Make one that sweeps from 20 Hz to 20 kHz logarithmically over a 15 second period. You can easily do this in MATLAB, as shown in Program 1. The FFT of the wave is shown in Figure 1.

This material is based on work supported by the National Science Foundation under CCLI Grant DUE 0717743, Jennifer Burg PI, Jason Romney, Co-PI.

```
function sweep(start, stop, secs, A)
% start:    start frequency
% stop:     stop frequency
% secs:     number of seconds for the sweep
% A:  amplitude
%Run with sweep(20, 20000, 15, 1) to get a one second sweep
%from frequencies 0 to 10,000 Hz at a sampling rate of 44.1 kHz
%At least 10 samples are needed for the last frequency
sr = stop*10;
if sr > 44100
  sr = 44100;
end
N = sr * secs;
f1 = start/sr;
f2 = stop/sr;
a = 2*pi*(f2-f1)/N;
b = 2*pi*f1;
for i = 1:N-1
  sweep(i) = A*sin(a*power(i,2)/2 + b*i);
end;
fftdata = fft(sweep);
magFreqs = abs(fftdata);
plot(magFreqs(1:(sr/2)));
soundsc(sweep, sr);
wavwrite(sweep, sr, 'sweep.wav');
end
```

**Program 1 A swept sine wave in MATLAB**

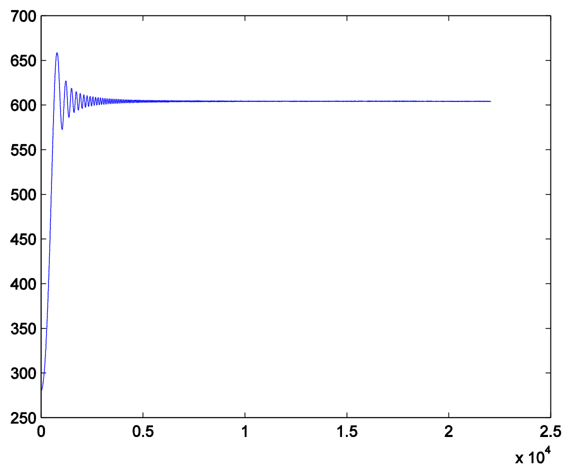

**Figure 1 FFT of swept sine wave**

Store the swept sine to a PCM or uncompressed WAV audio file so you can play it back in the acoustical space you want to capture. Then take your swept sine and whatever playback system you have to this space. Keep in mind that the IR will inherently capture the response of your playback system as well, so if you're using a cheap boom box, your IR

This material is based on work supported by the National Science Foundation under CCLI Grant DUE 0717743, Jennifer Burg PI, Jason Romney, Co-PI.

will also impart a cheap boom box response to any convolved signal. Play the swept sine wave out of your loudspeaker, while simultaneously recording the sound in the room. Try to position your mic in a reverberant area of the room that sounds good to your ears, and don't face it or the loudspeaker directly at each other. Remember that you're trying to capture the sound of the room, not the equipment.

Once you have the captured the swept sine output, you're ready to obtain your IR. (Feel free to try out a number of captures with different mic and loudspeaker positions for your swept sine output.) Using MATLAB, deconvolve the output recording with the original swept sine audio file. MATLAB has a function for this, *deconv*. Save the result as a PCM audio file, also, as shown in Program 2.

```
function out = getIR(sweptWave, recordedWave, sr)
% sweptWave:  filename of swept sine wave
% recordedWave: filename of sound recorded when the
% swept wave was played in an acoustical space
% sr:      sampling rate
swept = wavread(recordedWave);
recorded = wavread(recordedWave);
% deconvolve to get the impulse response,
% which can be used as a filter on
% other another audio signal to make it sound as if
% it is being played in the original acoustical space
[out, r] = deconv(recorded, swept);
plot(out);
soundsc(out, sr);
wavwrite(out, sr, 'IR.wav');
end
```

**Program 2 Using deconvolution to capture an impulse response to use as a filter**

In an audio editor such as Audacity, you'll want to clean up the IR file a bit, trimming out any extra silence before the impulse, and after the impulse decays. Your IR file shouldn't be any longer than the reverb time of the room you captured, likely no more than several seconds long at most. You could try to program some of this cleanup into your MATLAB function, but sometimes it's easier to do it visually looking at the waveform. If you happen to listen to your IR file at this time, you'll probably notice it sounds like a sort of "pop" with an unusual timbre, as mentioned earlier in the chapter. You can now take this IR and load it into a compatible convolution reverb plugin in your DAW, or use MATLAB to convolve it with an audio signal of your choice.

This material is based on work supported by the National Science Foundation under CCLI Grant DUE 0717743, Jennifer Burg PI, Jason Romney, Co-PI.