Department of Electrical & Computer Engineering
McGill University

ECSE-490 DSP Laboratory
Experiment 1

Implementing Audio Effects in MATLAB and Simulink

1.1 Purpose

In this experiment, you will implement a few basic audio effect algorithms in order to become familiar with the programming tools you will use during the class, namely MATLAB and Simulink. Students are encouraged to use Simulink for implementation of the systems. The output of the simulation can then be processed in MATLAB to visualize and or listen to the results.

1.2 Reverberation

In any natural environment, the sounds we perceive are always dependent on the listening conditions, and in particular on the environment. It is well known that the same signal heard in a big cathedral or in a small room will not"sound" the same. The room in which the listening experience takes place shapes the way any sound is perceived. Physically, the phenomenon of *reverberation* takes place, in which different copies of an audio signal reach the hear with different delays and different amplitudes, after taking different paths and having bounced against walls and surrounding objects. Each room (with a fixed configuration of furniture and occupants) can be "measured" in terms of how it shapes the perception of sound. For a fixed position of the sound source and of the microphone, the different delayed and attenuated versions of the original sound can be thought of as being produced by a linear time invariant system. More precisely, the perceived signal at the location of the microphone is[1]

$$y[n] = \sum_i \alpha_i x[n - D_i] = h[n] * x[n], \tag{1.1}$$

where $h[n] = \sum_i \alpha_i \delta[n - D_i]$. One can measure the impulse response of a room by simply producing an impulsive audio signal and recording the signal captured by the microphone.

[1]We assume sampled signals in this experiment.

In practice, simulated reverberations are not carried out with impulse responses measured from actual rooms. The main reason for that is the fact that room impulse responses tend to be very long, and represent a heavy computational load to implement the convolution in Eq. (1.1). The following sections will introduce basic building blocks to construct a synthetic reverberation tool.

1.2.1 FIR comb filter

An FIR comb filter with delay M is represented by an impulse response as follows

$$h[n] = \delta[n] + G\delta[n - M],$$

where G is a scalar gain. Such a comb filter can model a single echo with a time delay corresponding to a delay of M sampling intervals, and an attenuation given by G.

1.2.2 IIR reverberator model

Figure 1.1 shows the system block diagram of a basic IIR reverberator, with two parameters, a feedback gain G and a delay M. For reasonably sized rooms, M is usually chosen so that it corresponds to a time delay between 25 and 100 ms. This reverberator generates multiple echoes.

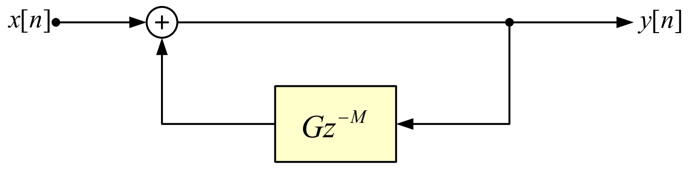


Fig. 1.1: Block diagram of an IIR reverberator building block.

- Compute the impulse response of a reverberator like the one in Fig. 1.1. Explain why it can simulate a reverberating room.
- Compute the frequency response of the IIR reverberator. Use MATLAB to plot the frequency response for sample values. What are the conditions for stability?
- Implement an IIR reverberator building block both in MATLAB and Simulink, as well as a series of resonators in parallel with different parameters.
- Especially in the case of speech signals, such a reverberator can sound metallic. This can be improved by including a simple FIR filter in the feedback loop. Implement this and listen to the difference.

1.2.3 Allpass reverberator model

Another system that can give more thickness to a sound and simulate a reverberant room is an allpass filer. This filter modifies the phase response, while leaving the amplitude response unchanged. An allpass filter has the same coefficients in the numerator and denominator, but in reversed order, e.g.

$$A(z) = \frac{a + bz^{-1}}{b + az^{-1}},$$

for any real a and b. An implementation of a first-order allpass filter is shown in Fig. 1.2.

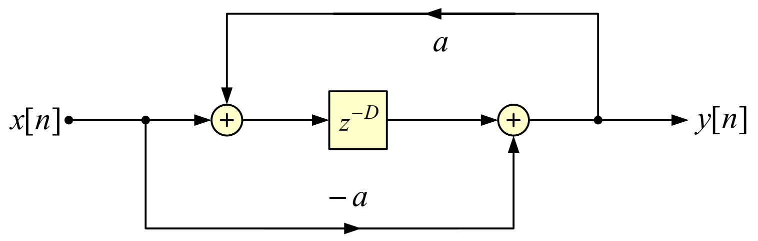


Fig. 1.2: Implementation of first-order all-pass transfer function.

- Check that the model is allpass for a random choice of parameters, by plotting its frequency response. Also plot its impulse response. Comment on the effect of this impulse response on the signal and how it can approximate a reverberator. Check that the implementation in Fig. 1.2 yields an all-pass filter. What are the conditions on a to ensure stability?
- Cascade a few allpass reverberators with the IIR-based system above and try to adjust the parameters so that you can simulate different types of rooms. You might want to also use an FIR filter to model so-called "early-reflections", in series with the above reverberator. In this case, the IIR/Allpass reverberator output should be delayed so that the early reflections reach the output before the IIR/allpass reverberated signals.

1.3 Ring Modulation

Ring modulation consists in modulating a signal so that the resulting signal is still in the audio range, but appears to have modified frequency content. In general, the modulation consists in a multiplication by a sine wave of frequency f_0, so that in the output, the frequencies that appear are the sum and difference of the original frequencies and f_0. Ring modulation is implemented by a multiplication in the time domain by a sinusoid of frequency f_0.

- Implement a ring modulator. Test it on a speech signal. By adjusting the value of f_0, find a choice which makes voices sound robotic.

1.4 Vibrato, Flanging and Chorus

The effects of vibrato, flanging and chorus are closely related in that they depend on the implementation of fractional delays.

1.4.1 Varying fractional delay implementation

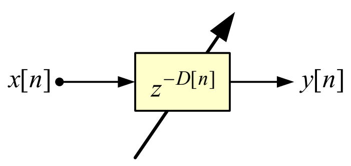
The three effects studied in this section rely on a time-varying delay element. The input-output relationship defining this element is given in time by

$$y[n] = x[n - D[n]],$$

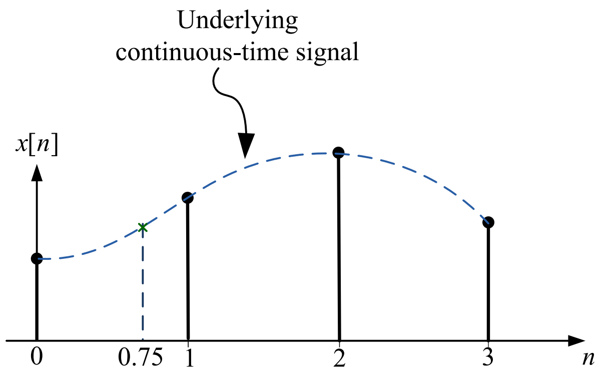
where the dependence of the delay D on the time index has been made explicit. In order to enable smooth variations of the delay $D[n]$, one often has to consider non-integer values of D. For non-integer delays, the values of a discrete-time signal between the sampling instants are assumed to be interpolated between the samples. More precisely, the value of $x[n]$ at a non-integer time t is obtained by using ideal bandlimited reconstruction of the underlying analog signal,

$$x(t) = \sum_{k=-\infty}^{\infty} x[k] \frac{\sin \pi(t-k)}{\pi(t-k)}.$$

Then the interpolated value is $x(n - D[n])$. Fig. 1.3 shows a block diagram representation of a variable delay and illustrates the interpolation operation in the time domain.



(a) Block diagram



(b) Interpolation

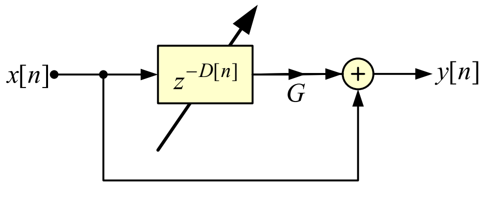
Fig. 1.3: (a) Block-diagram representation of a variable delay; (b) Illustration of interpolation in the time domain, where one wants to acquire the signal value at time 0.75.

In practice, we approximate the interpolation using an infinite number of sinc($\cdot$) functions with a simpler interpolation function. For instance we can use linear interpolation or polynomial interpolation between the available sampling points.

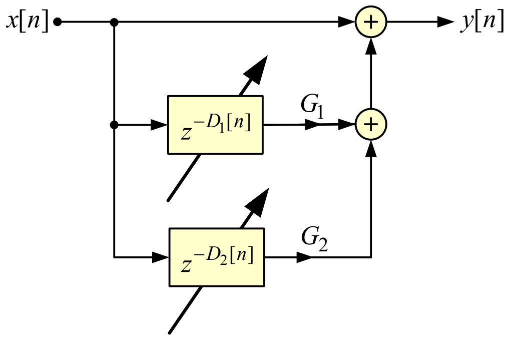
1.4.2 Implementing the effects

Vibrato is an effect implemented directly by the structure in Fig. 1.3, where the delay $D[n]$ is periodically varied around an average value by a low-frequency oscillator (typically of frequency 5–15 Hz). This results in a periodic variation of the pitch (perceived frequency) of the signal. Typical average delay values are between 5 and 10 ms.

Flanging is obtained by using an FIR comb filter where the delay is made variable, as illustrated in Fig. 1.4. The delay is again varied in a periodic way around some small value (less than 15 ms), with a low frequency oscillator (around 1 Hz). The resulting signal will be affected by a very distinctive periodic phase distortion.



(a) Flanging



(b) Chorus

Fig. 1.4: Block diagram for (a) flanging and (b) chorus effects.

Chorus is an effect that simulates the presence of several sources playing in imperfect unison. It is implemented by combining the original signal with several copies delayed by a randomly varied number of samples. Typical delays in the variable delay lines are 10 to 25 ms, and they are modified by small, slowly varying random variations. Fig. 1.4 shows a possible implementation.

- Implement all three effects in MATLAB.

1.5 A Few Hints

- You will be processing a lot of data. Make sure that you set your Simulink parameters to process discrete-time data. In addition the blocks in Signal Processing Blockset can be used to speed up simulations as they can process data in blocks.
- In order to load the audio files, use the MATLAB commands `auread` or `wavread`, depending on the file type. Note that some of your files are in stereo. They will give you a matrix with two columns. Convert the output to mono (average the stereo channels). In order to play a sound from MATLAB, use the command `sound`. Bring headphones to the lab in order to listen to samples. Simulink contains an output to WAV format block, that can also be directly output to the computer's line out.

References

[1] A. .V. Oppenheim, R. W. Schafer, with J. R. Buck, *Discrete-Time Signal Processing*, 2nd edition, Prentice-Hall, 1996 (ISBN 978-0-13-754920-7).

[2] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*, 4th edition, Prentice-Hall, 2007, (ISBN 978-0-13-187374-2).