REALSIMPLE Lab on Reverberation

Julius O. Smith III and Nelson Lee,

RealSimple Project*
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

January 27, 2009

Abstract

In this REALSIMPLE laboratory, you will implement two different reverberation algorithms within the STK framework. Each reverberation algorithm uses the fundamental element in acoustic modeling with delay, the delay line. The delay line is key in modeling different propagation paths from a source. You will then use Matlab to analyze the results of your reverberation algorithm and develop useful understandings of concepts and terminology applied to reverberation.

Starter code MyJCRev.tar.gz[1] and FDNJot4LP.tar.gz[2] and soundfiles[3] for this assignment may be downloaded from the links in the footnotes. To uncompress the `.tar.gz` files, first save them into your STK `myproj` directory (typically `~/stk/stk/myproj`). Then use the following command in a terminal at the same directory level:

```
tar -zxf filename.tar.tz
```

The directory <`filename`> should then be created.

1 JCReverb

(20 pts)

For starter code, see MyJCRev.tar.gz[4].

The STK module `MyJCRev` will implement a Schroeder-type reverberator.[5] However, the reverberator output is fed into only two delay lines instead of four as shown in the figure, for stereo output. The left and right channels correspond to `OutA` and `OutB`, respectively, in the reverberator diagram. Assume a sampling rate of 22050 Hz.

*Work supported by the Wallenberg Global Learning Network

[1] http://ccrma.stanford.edu/realsimple/reverb/MyJCRev.tar.gz
[2] http://ccrma.stanford.edu/realsimple/reverb/FDNJot4LP.tar.gz
[3] http://ccrma.stanford.edu/realsimple/reverb/sounds/
[4] http://ccrma.stanford.edu/realsimple/reverb/MyJCRev.tar.gz
[5] http://ccrma.stanford.edu/~jos/pasp/Schroeder_Reverberator_called_JCRev.html

Complete the `tick` method in the `MyJCRev.cpp` file. (Since the allpass sections are to be built using the `Rev` modules from the previous homework, you may also need to make changes to the instantiation of the `Rev` objects in the `MyJCRev` constructor) For those following the Realsimple lab path, the previous homework corresponds to the STK module written for the previous `Lattice Ladder` lab.

Turn in your code for the `tick` method. Also plot the impulse response of the reverberator's left channel response on a linear amplitude scale and on a log magnitude (dB) scale.

2 JotReverb

In this section we will study another well known reverberation algorithm, JotReverb. A subsection on the stability of the reverberator follows its implementation.

2.1 JotReverb in STK

(30 pts) For starter code, see FDNJot4LP.tar.gz[6]

The STK module `FDNJot4LP` provided in the starter-code will implement an order 4 single-input, single-output feedback delay network with a one-pole lowpass filter

$$H_i(z) = g_i \frac{1 - a_i}{1 - a_i z^{-1}}, \quad i = 1, 2, 3, 4$$

cascaded with each of the four delay lines.

The basic order 4 SISO FDN has $B^T = [1, 1, 1, 1]$, $C^T = [1, 1, 1, 1]$, and feedback matrix $A = A_4$, where

$$A_N \triangleq g \left(I_N - \frac{2}{N} u_N u_N^T \right)$$

where I_N denotes the $N \times N$ identity matrix and $u_N = [1, \ldots, 1]^T$. (It may be helpful to refer to the third-order example in Figure 1.26[7] of the text.) When $g = 1$, A_N is a so-called *Householder reflection* about the vector u_N in N-space. Note that A_N is orthogonal ($A_N^T A_N = I_N$) when $g = \pm 1$ (as is any real rotation matrix). It is actually more efficient to implement the Householder reflection as written above than to calculate and implement the resulting four-by-four matrix.

Complete the constructor and the `tick` method in `FDNJot4LP.cpp`. Use Jot's formulas to compute g_i and a_i.

Turn in your code for the constructor and the `tick` method. Test `FDNJot4LP.cpp` with an impulse input for the case $g = 1$, $t_{60}(0) = 2.0$, $t_{60}(\pi/T) = 1.0$, and M_i set to four delay line lengths in `MyJCRev.cpp` $(113, 337, 1051, 4799)$. Plot the impulse response on a linear amplitude scale and on a log magnitude (dB) scale.

2.2 Reverberator Stabililty

(10 pts) $t_{60}(\pi/T)$ must be constrained to what range of values for the filters $H_i(z)$ to be usable (stable)?

[6] http://ccrma.stanford.edu/realsimple/reverb/FDNJot4LP.tar.gz

[7] http://ccrma.stanford.edu/~jos/pasp/Single_Input_Single_Output_SISO_FDN.html

3 Energy Decay Curve and Energy Decay Relief

In this section, you will complete the provided Matlab listings for observing the Energy Decay Curve and its generalization, the Energy Decay Relief.

(20 pts) Complete indicated sections in the Matlab files edrAndPLot.m[8] and edcAndPLot.m[9] in order to calculate the Energy Decay Curve (EDC) and Energy Decay Relief (EDR) of a signal. Turn in the new code you wrote, and plot the EDC and EDR for the impulse response of an actual acoustical space and for the impulse responses of the reverberators you built in problems 1 and 2.

Soundfiles[10] for this assignment may be downloaded from the link at the footnote below.

4 A Listening Test

(10 pts) Sonically compare the "real" reverberator and the two artificial reverberators. Listen to their impulse responses and the results of processing an anechoic recording with them, and describe the differences that you notice. (You will need to use the Matlab function `conv` to process the soundfile with the recorded cathedral reverberation. And it will take a while to compute!) Compare the reverberators visually, considering their impulse responses, EDCs, and EDRs.

5 Estimating t60

There are many uses for estimating t_{60}. As shown in a later lab, estimating this parameter is useful for tuning a loop-filter for a string-model using a Digital Waveguide. In the realm of reverberation, it is a useful metric for characterizing different reverberators.

(15 pts) Analyze the order 4 FDN reverberator impulse response from problem 2 by calculating its EDR and spectrogram (in dB units). Using the EDR, estimate the decay times near dc and near half the sampling rate and compare them to the values desired at design time. Do the same, using data from the spectrogram. To measure t_{60} you will need to complete the Matlab code provided estimateT60.m[11], which will fit a straight line to log magnitude data across time frames for different frequency bins.

Turn in the new code you wrote and the measured $t_{60}s$ (four in total).

[8] http://ccrma.stanford.edu/realsimple/reverb/matlab_files/edrAndPLot.m
[9] http://ccrma.stanford.edu/realsimple/reverb/matlab_files/edcAndPLot.m
[10] http://ccrma.stanford.edu/realsimple/reverb/sounds/
[11] http://ccrma.stanford.edu/realsimple/reverb/matlab_files/estimateT60.m