REALSIMPLE STK Lab on Lattice Ladders

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

June 5, 2008

Abstract

This is the REALSIMPLE STK Lattice Ladder Laboratory. It serves as the first of two labs giving students experience in STK programming and a hands-on opportunity for implementing popular, well-known reverberation algorithms.

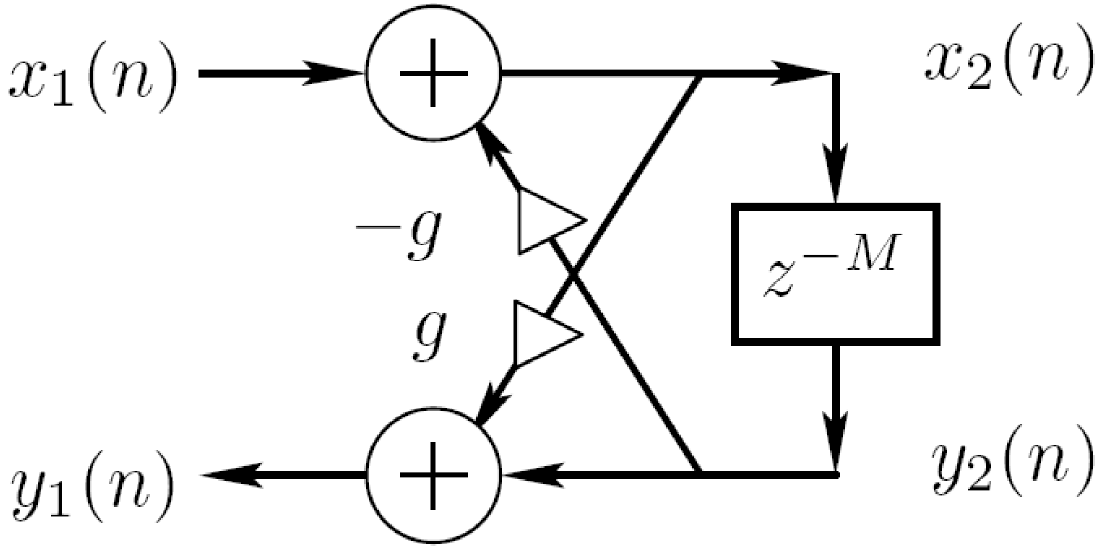

Figure 1: Two-multiply lattice filter.

*Work supported by the Wallenberg Global Learning Network

1 Introduction

In careful observation of the physics of reverberation, the signal that we hear repeated arrives at our ears at different times due to the different paths the sound waves travel. Take for example you are standing in a church: you clap your hands and hear an immediate 'smack' and reverberating copies of the same 'smack' from different parts of the room. The first 'smack' you heard arrived at your ear through a direct path from your hands. The reverberant copies travelled from your hands to perhaps a distant wall, reflecting back to your ear. As Figure 1 shows, x_2 fed straight into the adder corresponds to the direct path of the signal x_1. y_2, which corresponds to the signal x_2 delayed by M samples, corresponds to the same signal but delayed by M samples or in physical terms, traveling a certain longer distance.

2 Exercises

1. (20 pts) Write an STK module called `Rev.cpp` which implements the two-multiply lattice filter shown in Figure 1. The input arguments should be the two coefficients and the delay-line length. Write a small main program to test your module with an impulse input, and verify that you get the same numbers calculated by hand. Turn in your commented program listings and a print-out of your test results for the case $g = 0.5$ and $M = 10$ (include at least 35 output samples). If you need help getting started with the code, observe the examples in the STK directory tree and use them as templates for your own implementation.

2. (20 pts) Write an STK module called `RevLP.cpp` which extends `Rev.cpp` to include the one-pole lowpass filter $y(n) = (1 - a)x(n) + ay(n - 1)$ *in the feedback loop only*. Thus, there should be one new argument a which can be set to 0 to obtain the former case (a useful intermediate test). Turn in your program listings and a print-out of your test results for the case $g = 0.7$, $M = 10$, and $a = 0.5$. With the addition of this lowpass filter, we can model physical phenomena of walls absorbing energy of the signal as it reflects.