Course Name: Digital Signal Processing Design

Course Number and Section: **14:332:447:01**

Dereverberation: Removing Unwanted Echoes and Reverb from Recorded Audio

Part 4 of 6: Multi-Delay

Submitted by: Lance Darragh

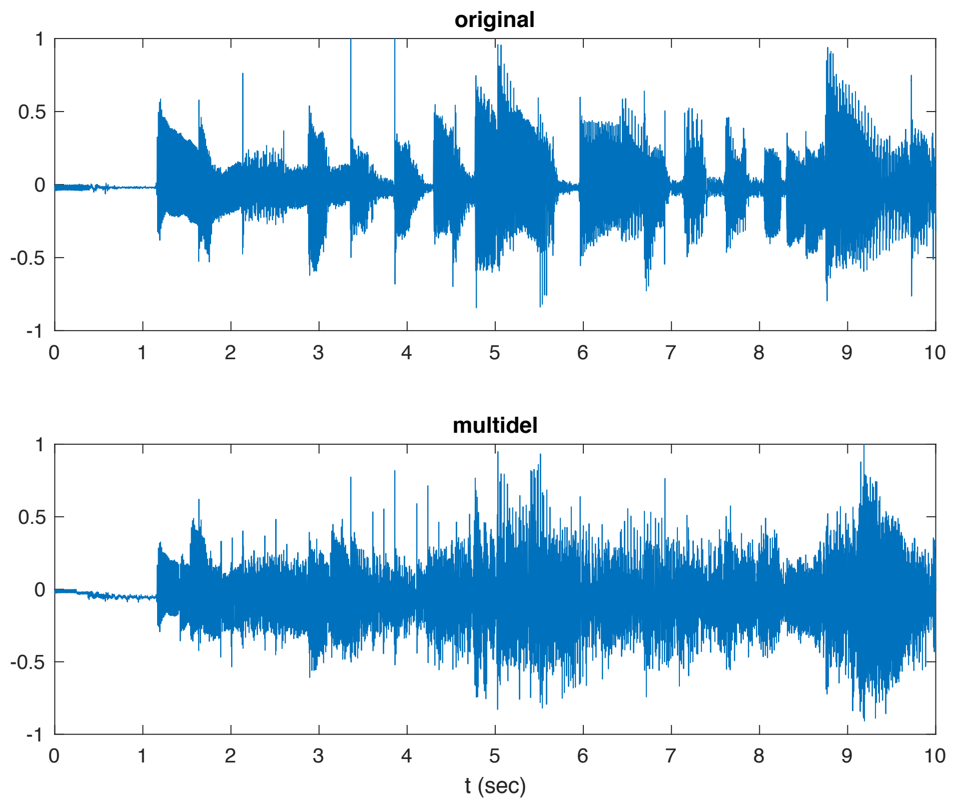
<u>Digital Audio Effects, Multi-Delay Filter</u>

In a real room, the acoustics are characterized by three main classifications of delay times between the sound source and the listener, the direct sound which goes straight from the source to the listener, the early reflections, which reach the listener after bouncing off of one to a few surfaces, and the late reflections, which are the early reflections bouncing off of several more surfaces before reaching the listener. The timing between delays gets progressively smaller, and the sound appears more dense to the listener. By taking the echoes of the previous filter and sending them into another delay unit with a different delay value D_2, we can have these echoes repeated more densely and mimic the more dense portion of the early reflections.

To implement this, will define the following transfer function:

(31)
$$H(z) = b_0 + b_1[\frac{z^{-D_1}}{1 - a_1 z^{-D_1}}] + b_2[\frac{z^{-D_1}}{1 - a_1 z^{-D_1}}][\frac{z^{-D_2}}{1 - a_2 z^{-D_2}}]$$

where a_1 and a_2 are the first and second reflection coefficients. We will let them equal 0.2 and 0.4 respectively, and b_0, b_1, and b_2 also play the role of reflection coefficients, but to scale the clusters of echoes as groups, with b_0 being the strength of the direct sound. For simplicity we let b_0, b_1, and b_2 all equal 1, D_1 = 0.25 msec, and D_2 = 0.125 msec. Appendix A.4 has the algorithm. Again we'll use the circular-buffer implementation and compare it with MATLAB's filter function. The execution time for the circular buffer method is 0.050099 seconds, which is about 50 times faster than MATLAB's built-in filter function, which is 2.495146 seconds. The larger difference is due to the additional data shifts of now using two delay lines.



Appendix A.4: Multi-Delay

```
% Import original audio
[s,fs] = audioread('original.wav');
% Delays 1 and 2
D1 = 0.25*fs; D2 = 0.125*fs;
% Amplitude coefficients
b0 = 1; b1 = 1; b2 = 1;
a1 = 0.2; a2 = 0.4;

% Determine the length of the signal
[N,k] = size(s);
% Internal delay buffer 1 for s(n)
w1 = zeros(1, D1 + 1);
% Internal delay buffer 2 for w1(n)
w2 = zeros(1, D2 + 1);
% Delay buffer 1 index variable
q1 = 1;
% Delay buffer 2 index variable
q2 = 1;
% Delay buffer taps
tap1 = D1 + 1;
tap2 = D2 + 1;
% Loop through input signal
for n = 1:N
    s1 = w1(tap1);
    s2 = w2(tap2);
    y(n) = b0*s(n) + b1*s1 + b2*s2;
    w2(q2) = s1 + a2*s2;
    w1(q1) = s(n) + a1*s1;
    q1 = q1 - 1;                % Backshift index 1
    if q1 < 1                   % Circulate index 1
        q1 = D1 + 1;
    end
    tap1 = tap1 - 1;            % Backshift tap1
    if tap1 < 1                 % Circulate tap1
        tap1 = D1 + 1;
    end
        q2 = q2 - 1;            % Backshift index 2
    if q2 < 1                   % Circulate index 2
        q2 = D2 + 1;
    end
        tap2 = tap2 - 1;        % Backshift tap2
    if tap2 < 1                 % Circulate tap2
        tap2 = D2 + 1;
    end
end

% Normalize y(n)
ymax = max(y);
y = y/ymax;
```

```
% Listen to the results
sound(y,fs);
```

Output the Results

```
audiowrite('Multi-Delay.wav', y, fs)
```

Published with MATLAB® R2017b