DSP Project:
Audio Click Removal
Using Linear Prediction

Malcolm Knapp and Raihan Bashir
12/8/04

Introduction:

Cleaning audio signals is one of the most widely used applications of digital signal processing. There are many different types of audio signal processing but the one this paper will focus on is click removal. Removing a click is an interesting problem because the energy of the click exists across all frequencies. Thus signals corrupted with clicks cannot be cleaned up solely through the use of filters. To completely remove the click the processing has to take place only in the time domain. Therefore, the only way to remove the click is to replace it with another signal. In this project Linear Prediction was used to generate the replacement waveform and to allow replacement with no discontinuities.

Theory:

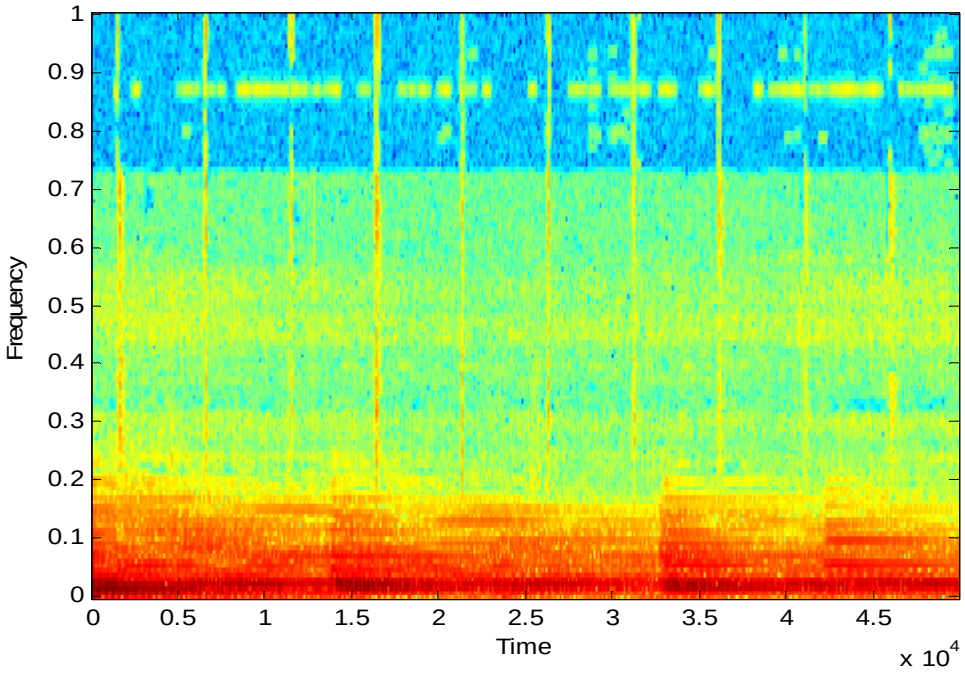
Click removal consists of two problems. The first is finding the click and replacing it with something else. The second is making sure that there are no discontinuities between original signal and the inserted signal. Simple click removal can deal with the first problem but is unable to deal with the second. Linear prediction, however, solve both of these problems.

Linear Prediction Coefficient (LPC) starts with the assumption passing a random signal though a specific filter would produce the signal of interest. Linear Prediction Coefficient (LPC) filtering constructs an all pole filter using prior values. Thus LPC filtering can predict what signal should look like where the click is. The key though is that if a signal is run through the inverse of this filter it will make the signal look like a random signal. This means the output of the inverse filter will look like white noise. One of the characteristics of white noise is that each point is unrelated to the points around it. The signal is reconstructed by running the whitened signal back through the original LPC filter. Thus, if one part of the signal replaces another part and the signal is run back through the filter the interface between the original signal and the inserted signal will be interpreted as if it was noise as well and reconstructed with no discontinuity.

Problem Specification:

The sound track we chose to filter with linear prediction has a periodic click in the background. The main purpose of the project is to device an algorithm to isolate the clicks in an iterative fashion, use linear prediction model to remove the clicks and retrieve click free signal so that the sound quality and musical continuity of the signal is the least affected. In diagram 1.1 we can notice the lines that represent the presence of rather periodic clicks by taking spectrogram of the original signal.

Diagram 1.1 Specgram analysis of the origainal signal



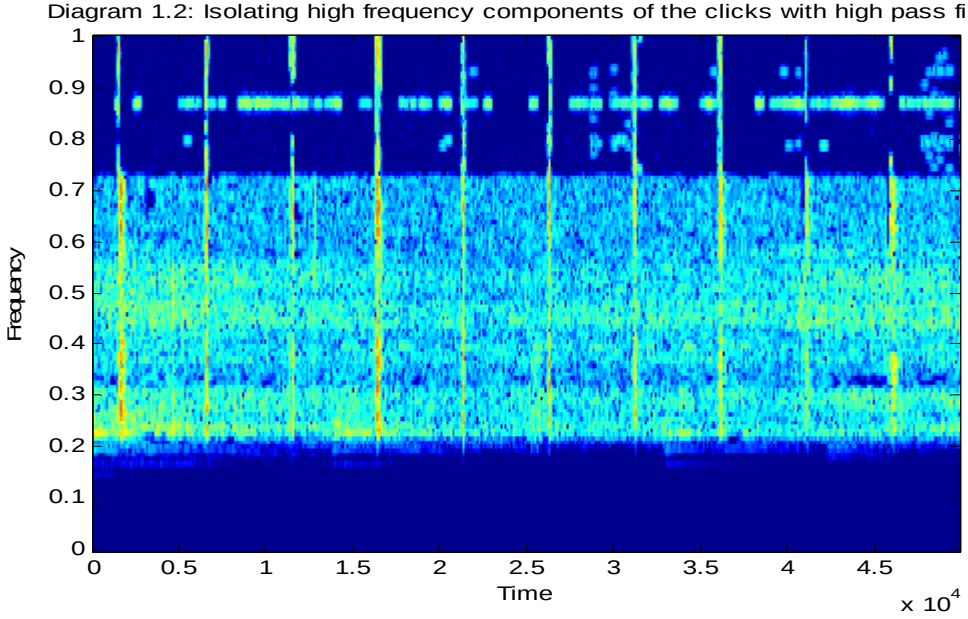
Solution Method:

From Diagram 1.1 we can notice that in terms of energy spectrum, large share of the energy associated with the clicks exist in frequency domain greater than 0.2. Based on this finding, our first step was to use a high pass filter with the following properties:

[b,a]=cheby1(8,3,5000/22050,'high');

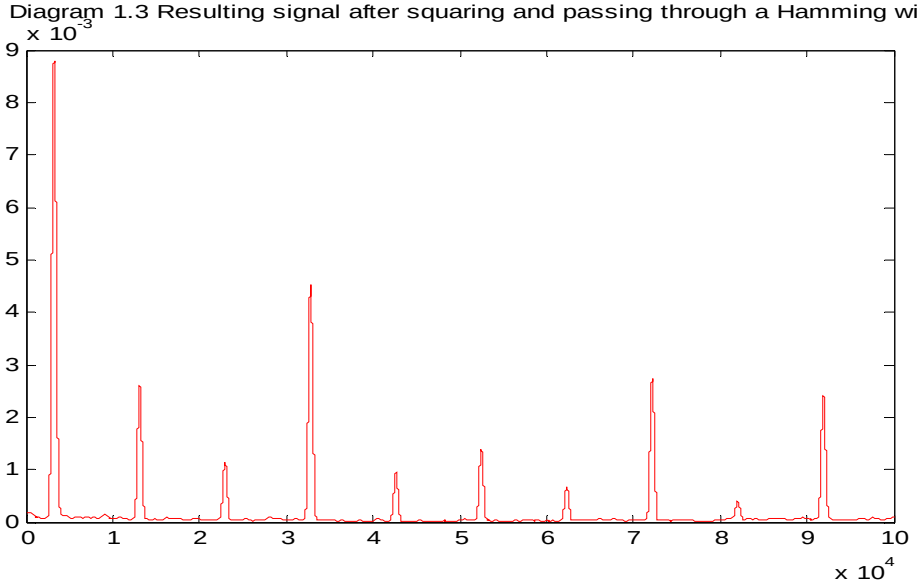
The resulting plot in the diagram 1.2 validates that we are successful in separating major energy components (high frequency) for each click.

Diagram 1.2: Isolating high frequency components of the clicks with high pass filter

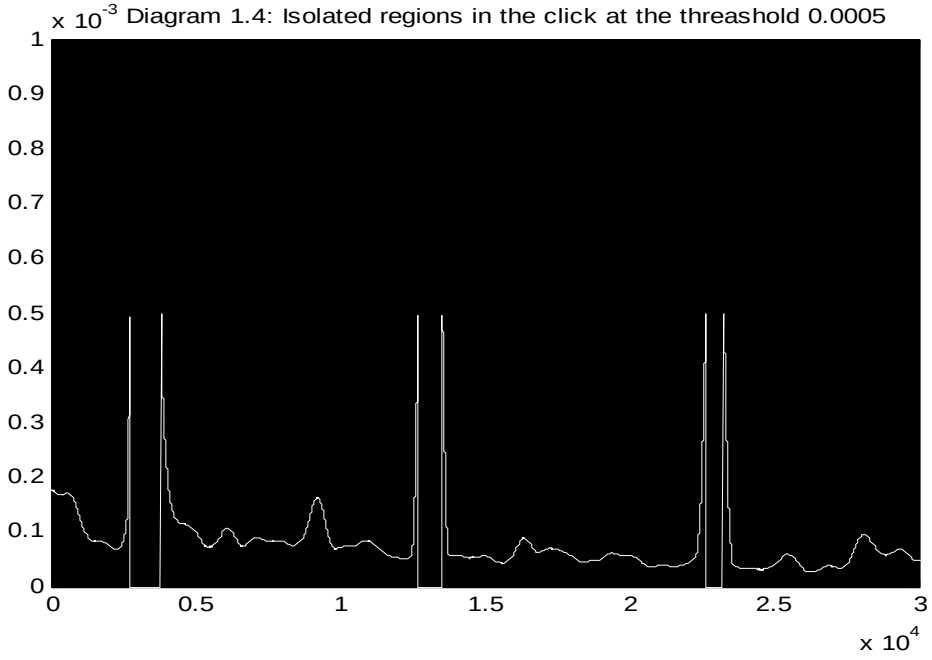


These are the main components for each click and the remainder of the click submerged in the music signal has comparatively very small energy to have any audible effect. Our following strategy involves squaring the resulting signal from the cheby filter. This will make all the input amplitude positive and we are ready to use a single threshold to isolate all the clicks in an iterative algorithm. We also make use of Hamming widow to smooth out the clicks to enhance our capability to efficiently isolate and replace them with linear coefficients. The results are visible in diagram 1.3:

Diagram 1.3 Resulting signal after squaring and passing through a Hamming window

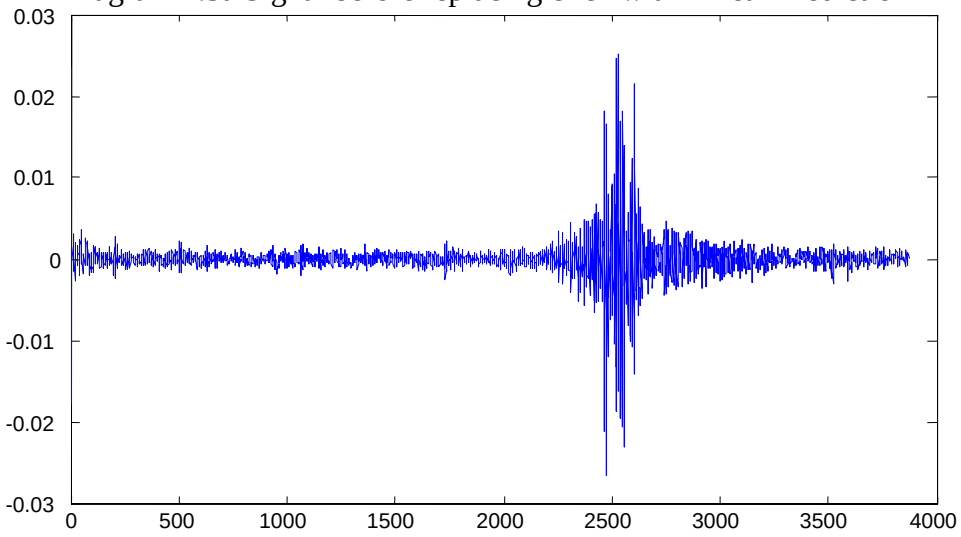


After retrieving smooth clicks we pass it through a loop to get x values above the single threshold amplitude 0.0005. This will give us the regions of the click that we will replace with equal length parts of the music signal following linear prediction. In Diagram 1.4 the blanked out region of the clicks is where we would do signal patching.

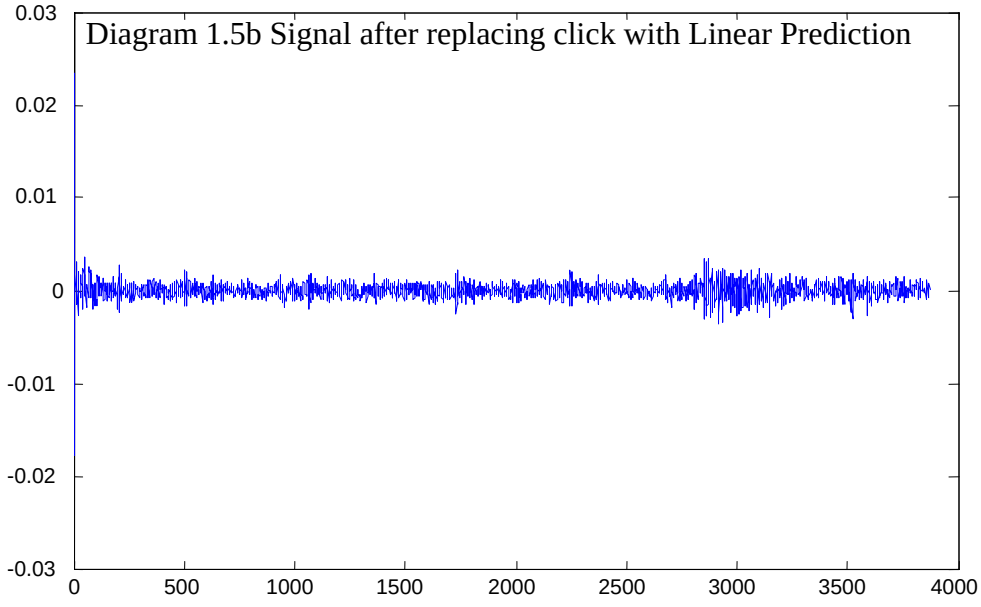


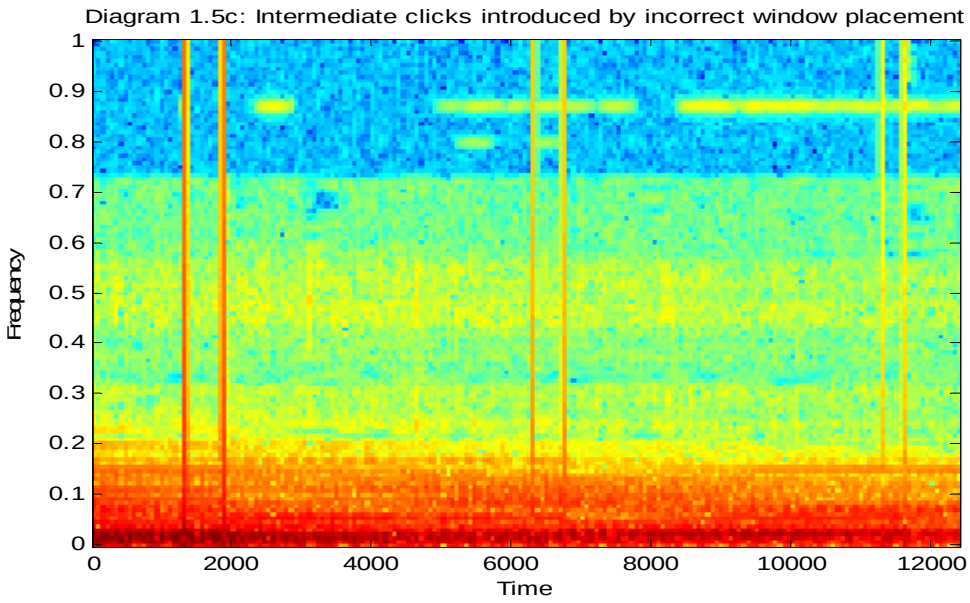
After a click was located an LPC filter was made from the uncorrupted samples before the click. Then an entire section including before and after the click was run through the inverse of the LPC filter to whiten the signal (Diagram1.5a).

Diagram 1.5a Signal before replacing click with Linear Prediction



The click is replaced with a corresponding length of signal. After that the whitened signal is run through the original filter to restore the signal (Diagram 1.5b).





This method is able to remove the click and replace it with a new section of song (Diagram 1.5c). However, initially this method produced two new clicks at the ends of the inserted signal. The click at the beginning of the inserted signal was caused by the fact that the inverse LPC filter was a 256-point filter. Thus it needed 256 points to fill it up before it would begin to produce an accurate prediction. The solution to the problem was to take the beginning of the replacement signal 256 points after the beginning of the inverse LPC output. The other click was an echo the part of the click included in the LPC filter. Since the LPC filter is an IIR filter all past input affects it. Thus, it would begin to reproduce the click at the end of the inserted sample. The solution to this was to extend the replacement sample past the end of the click. The further from the end of the click the end of the replacement sample was the more the second click was attenuated. With the end of the replacement signal set at1000 samples past the end the click the second click was almost completely removed.

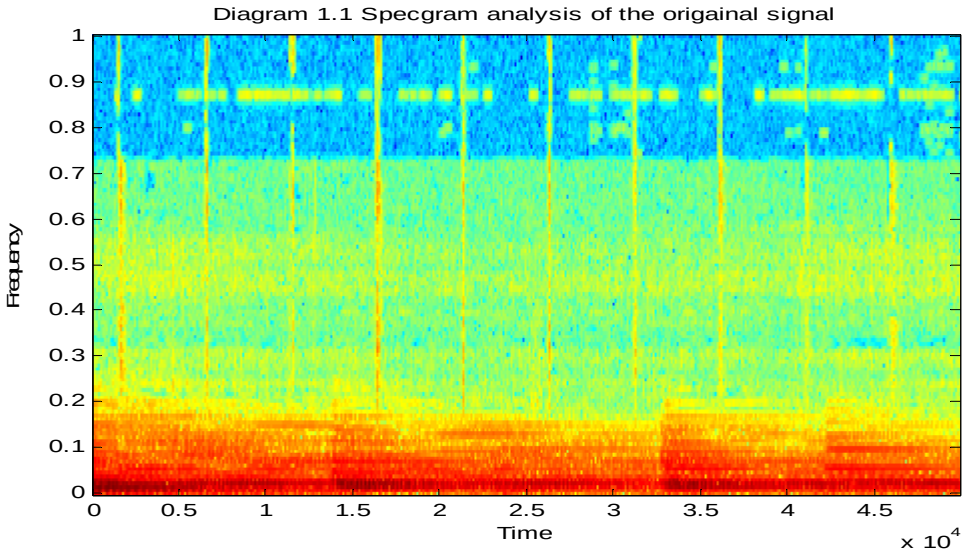
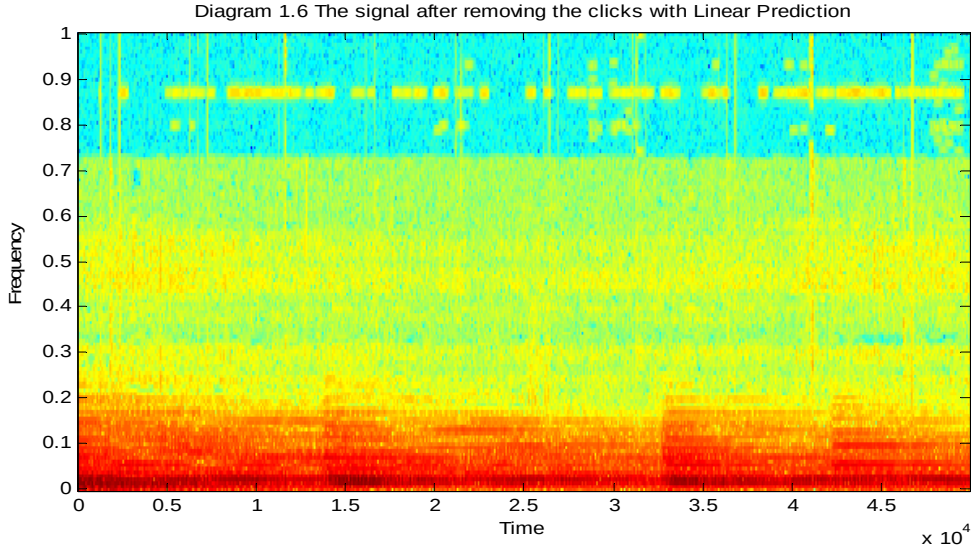


Diagram 1.6 The signal after removing the clicks with Linear Prediction



Results:

The clicks are clear in Diagram 1.1. When the sample was played these clicks were clearly audible. Diagram 1.6 shows the spectrogram of the signal after it was passed though the click removal process. From this diagram it is clear that the majority of the click energy is removed. There is still some remnant energy but it is inaudible when the sample is played.

Conclusion:

The method could successfully find the click and replace them without causing any discontinuities. There is still room improvement though. As a result of the click removal process, a "wobble" has been introduced into the sample. This "wobble" is caused by the repetition of the song where the clicks are. The "wobble is noticeable because the size or the replacement window is large. Thus, for the future work has to be done to minimize the window length. Another area where this method could be improved is the threshold that determines if there was a click or not. Currently the program uses a single threshold which works well for this simple section of song. However, this static threshold probably would not work on another sample which had more high frequency components or was more complex. A dynamic threshold could solve this problem by adjusting the threshold to the local characteristics of the sample. Despite the minor problems, this project has successfully completed all of its objectives.

Appendix A: Source code

```
% Matlab code for the program ClickFilterFinal.m

Clear;

% y stores the snap of 4:05 to 4:15 of the track

y = wavread('08 Warning Sign.wav',[10804500 11245500]);
plot(y);
specgram(y(:,1),1024,44100);
caxis([-40 40]);

%Step 1: Isolate the high frequency components of the
%clicks

[b,a]=cheby1(8,3,5000/22050,'high');
yf = filter(b,a,y(:,1)');
%soundsc (yf, 44100)
caxis ([-80 0])
plot(yf)
hlen=1024;

%Step 2: Square the signal and smooth up with the Hamming
%window

yfpos = conv(hamming(hlen),yf.^2);
yfpos = yfpos(hlen/2 + 128:end);
test1 = (yfpos(209500+[1:100000]));
test2 = y(209500+[1:100000]);
test3 = test2;

%Step 3: Set the threshold at 0.0005 to isolate regions of
%click to be replaced by non corrupted signal

x = find(test1>.0005);
x (length (x) + 1) = 0;
K = 1;
while K < length(x)
I = K;
J = 1;
while x(I) + 1 == x(I+1)
    x1(J) = x(I);
    I = I + 1;
    J = J + 1;
end
```

```
x1(J) = x (I);

gwin = 2000;
lx1 = length(x1);
flen = 256;
ye = test2((x1(1)-gwin):(x1(lx1)+4*flen));

%Step 4: LPC filter was made from the uncorrupted samples
%before the click and an entire section including before
%and after the click was run through the inverse of the LPC
%filter to whiten the signal

a = lpc(ye(1:gwin),flen);
ee = filter(a,1,ye);

%Step 5: The click is replaced with a corresponding length
%of signal

em=ee;
ee (gwin+ [1:lx1]) = ee(flen + [1:lx1]);

%Step 6: After that the whitened signal is run through the
%original filter to restore the signal

ere = filter(1,a,ee);
ex=test3([(x1(1)-gwin):(x1(lx1)+ 4*flen)]);
test3([(x1(1)-gwin):(x1(lx1)+ 4*flen)]) = ere;
K = I + 1

subplot (311)
specgram (test2)
title('Region (x1(1)-length(x1)+ [1:length(x1)- 1]) in
test2:');
subplot (312)
specgram (test3)
title('Region (x1(1)-length(x1)+ [1:length(x1)- 1]) in
test3:');
subplot (313)
specgram (test1)

clear x1;
end
```