Lab 01 - Basic Compressor

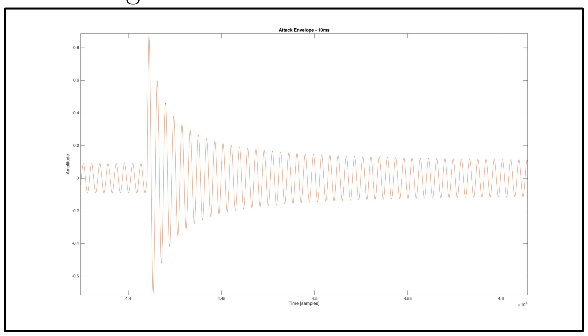
Ron Guglielmone

PROBLEM 1(a)

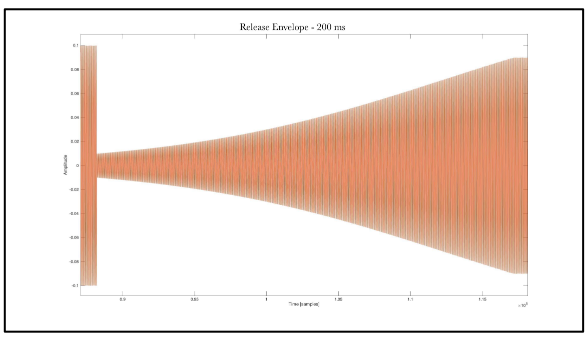
Done.

PROBLEM 1(b)

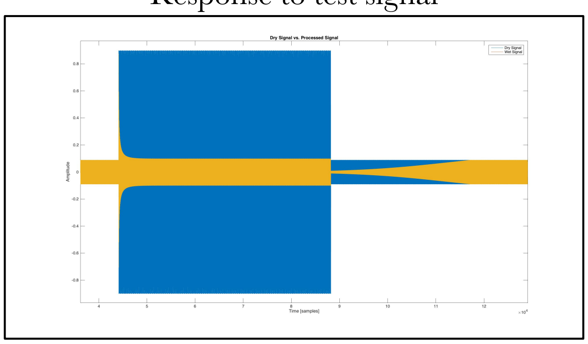
Signal at time of attack: 10 ms



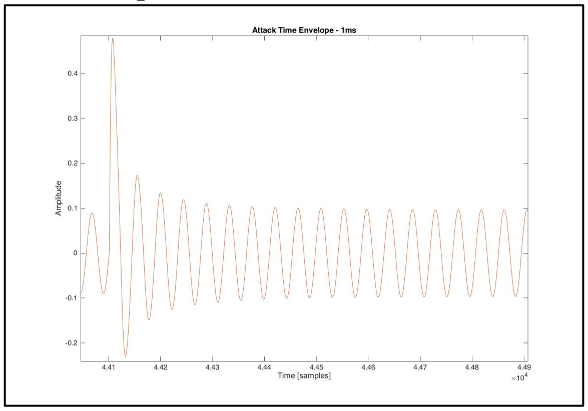
Signal at time of release: 200 ms



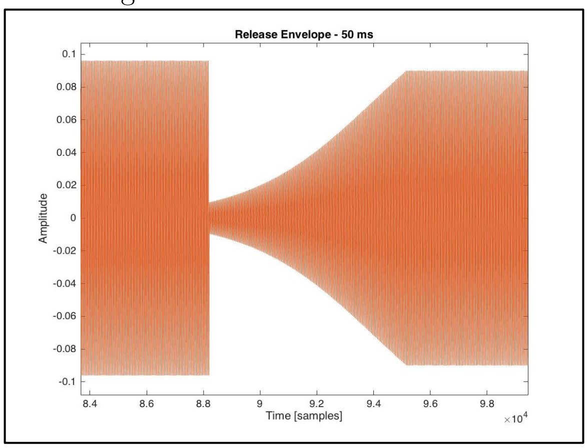
Response to test signal



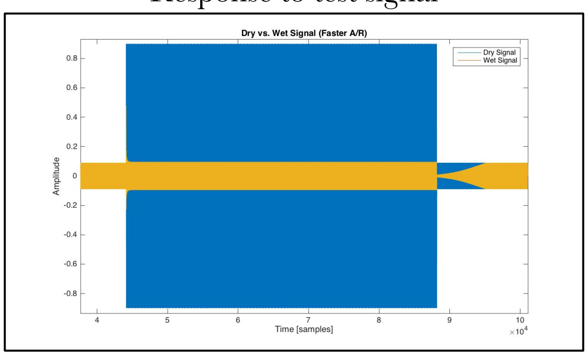
Signal at time of attack: 1 ms



Signal at time of release: 50 ms



Response to test signal



MATLAB script for 1(b):

```
% Ron Guglielmone
% MUSIC 424, CCRMA, Stanford University
% April 11, 2017
%
% Problem 1b
%
% For 10 ms Attack Time and 200 ms Release Time
% plot the output of test signal tdiff1.wav
% around the attack onset.  Then, plot around
% the release onset.

% Read audio file into MATLAB and plot:
[y2,fs2] = audioread('Problem_1b_Pset_1.2.wav');
[y1,fs1] = audioread('tdiff1.wav');
plot(y1);
hold on;
plot(y2);
title('Dry vs. Wet Signal (Faster A/R)');
xlabel('Time [samples]');
ylabel('Amplitude');
legend('Dry Signal', 'Wet Signal');
```

PROBLEM 1(c)

Drum signal, Release Time 200 ms, input gain +3dB, output gain -3dB, threshold at -12dB. Adjust Attack Time from 0.1 ms to 100 ms.

At what values of attack time does the compressor make the drums sound...

- smooth/round: Between 2-4 ms
- tinky/small: Below 0.5 ms
- thuddy/muddy: Between 0.5 ms - 2 ms
- transparent: Above 20 ms

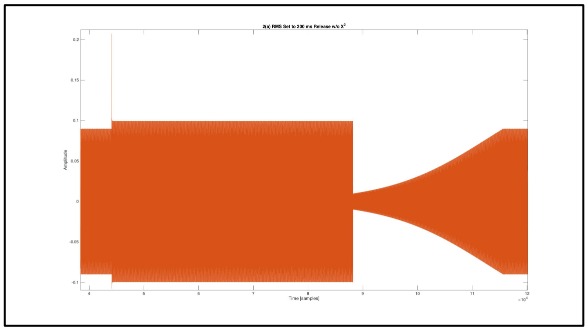
Attack Time 0.1 ms, Release time from 20 ms to 200 ms.

At what values of release time does the compressor make the drums sound

- buzzy: 50 - 100 ms
- roomy: 100 - 200 ms
- even: 200 + ms

PROBLEM 2(a)

Output of the compressor with RMS detector:



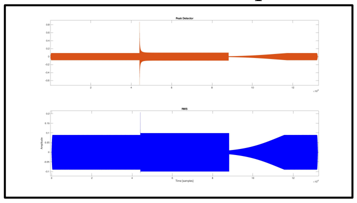
Code change that enabled this:

```
RMS detector
struct RMSDetector {

    float   b0_r, a1_r, b0_a, a1_a, levelEstimate;

    RMSDetector() {

        this->a1_r = 0;
        this->b0_r = 1;
        this->a1_a = 0;
        this->b0_a = 1;
        reset();
    }

    void setTauRelease(float tauRelease, float fs) {
        a1_r = exp( -1.0 / ( tauRelease * fs ) );
        b0_r = 1 - a1_r;
    }

    void setTauAttack(float tauAttack, float fs) {
        a1_a = exp( -1.0 / ( tauAttack * fs ) );
        b0_a = 1 - a1_a;
    }

    void reset() {
        levelEstimate=0;
    }

    void process (float input, float& output) {

        input = pow(input,2); // Square input

            levelEstimate += b0_r * ( fabs( input ) - levelEstimate );

        output = pow(levelEstimate,0.5); // Take square root
    }
};
```

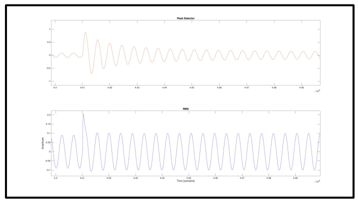
PROBLEM 2(b)

- How does the output envelope compare to the peak detector model?

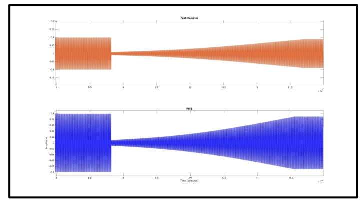
Whole Envelope



Attack Portion



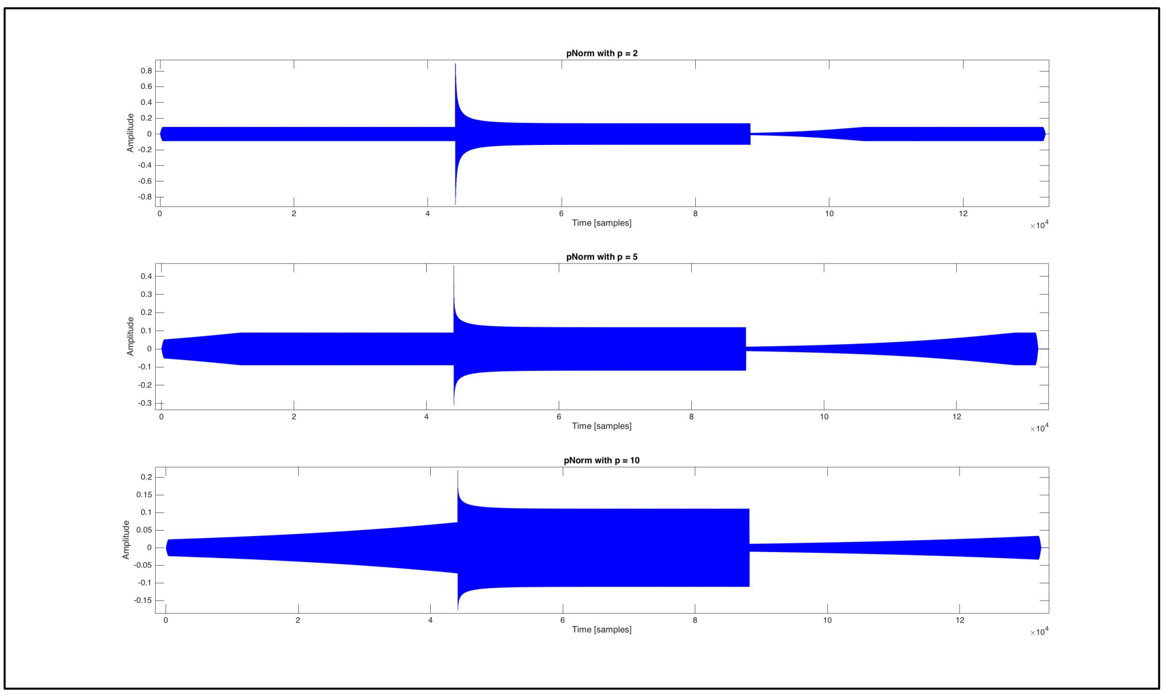
Release Portion



- How does the sound on musical signals differ between the two?

The RMS version "pumps" more, but contains less harmonic distortion.

PROBLEM 3 (a)



Plots of pNorm detector output with p = 2, 5, and 10.

(Full C++ code attached in separate file).

PROBLEM 3(b)

- For a given exponent p, and leaky integrator time constant T, derive expressions for the attack and release time constants of the RMp detector.

I did not leave myself enough time to solve this, but will come back to it...