**EE286-MP1**Synthesizer Machine Problem

Narz Marbeth David  
Carl Justin de Guia

**ADSR Envelopes**

When a mechanical musical instrument produces sound, the relative volume of the sound produced changes over time. The way that this varies is different from instrument to instrument [1].

A module was developed to extract ADSR parameters from the samples and use these to generate ADSR envelopes. The ADSR extractor identifies the attack start and end times, decay start and end times, sustain levels, and release start and end times by examining the differential of the intensity contour.

The attack start is identified as the point where the power rises by at least 6dB per frame. This 6dB threshold was chosen based on the average hearing thresholds for tones between 200Hz and 1000Hz. The attack ends on reaching the maxima of the intensity contour. The decay phase also starts at this point. To identify the decay's end, the ADSR extractor traverses the intensity differential forward in time, until it reaches a change >=0dB, which ends the decay phase. The release phase is subsequently identified by looking for the onset of the decay rate of -0.5db/frame. The sustain level is then approximated to be the average power in the time frame sandwiched between the decay and release phase. The following diagram shows the intensity contours of each sample plotted against the generated ADSR envelope. The attack, decay and release time bounds are marked as x on the plots.

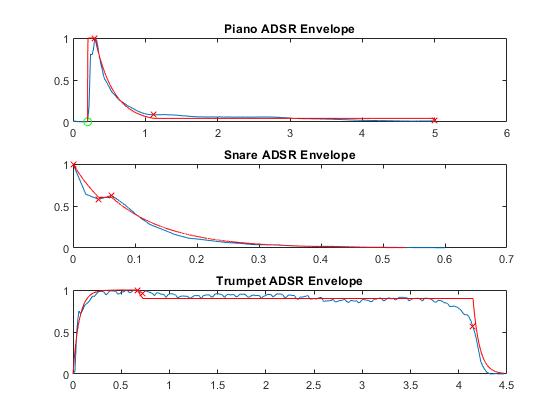
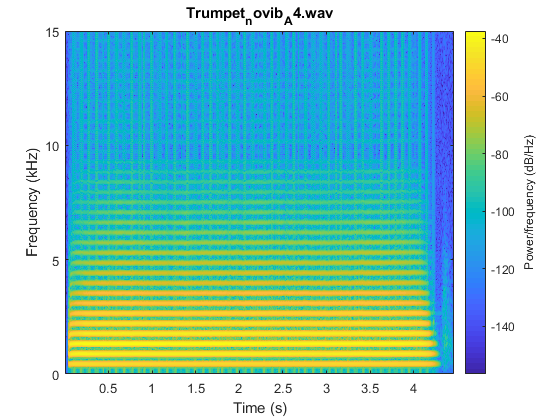
[](https://github.com/narzdavid/EE286-MP1/blob/master/envelopes.jpg?raw=true)

Figure 1. ADSR Envelopes

**Trumpet**

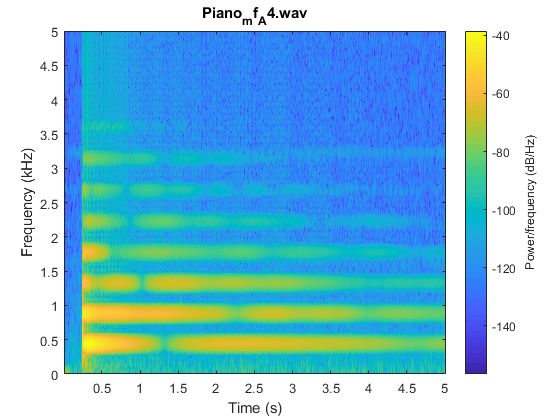
As seen in the spectrogram below, the trumpet shows peaks at F0 = 440Hz and its other harmonics at nF0. The energy is consistent along the time domain, and the energy decreases at higher harmonics.

[](https://github.com/carldegs/EE286-MP1/blob/master/spectrogram_trumpet.png?raw=true)

To synthesize the trumpet sample, synthesis.m generates sinusoids at the frequency peaks observed in the spectrogram. As seen in Figure 1, the decay and attack phase are very short and almost imperceptible.

**Piano**

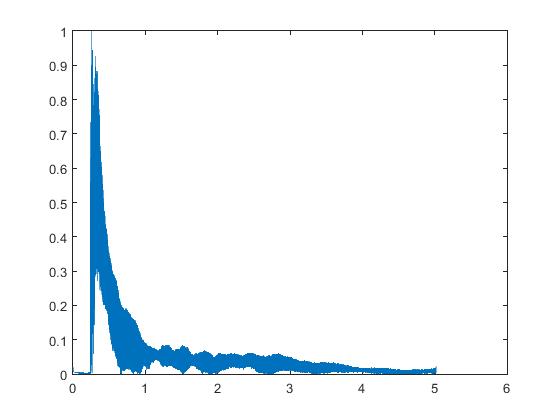
The piano also shows peaks at F0 = 440Hz and its harmonics but it is non-evident as it reached the 7th and 8th harmonics. The energy in the time domain is also not consistent, with multiple tracks at each pitch.

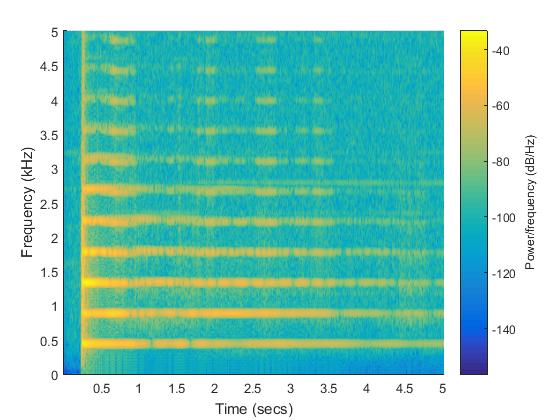
[](https://github.com/carldegs/EE286-MP1/blob/master/spectrogram_piano.png?raw=true)

To synthesize the piano sample, a sum of sine waves with the above frequencies are generated, similar to the trumpet.

**Envelope**

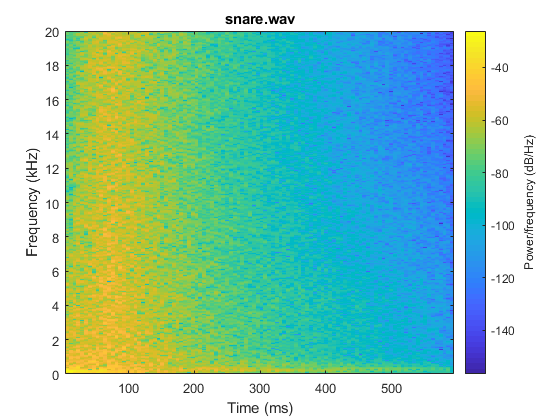
Instead of creating an ADSR envelope, **Hilbert Transform** is used to create an envelope based on the original signal. The created envelope as seen below.

[](https://github.com/carldegs/EE286-MP1/blob/master/piano_envelope.jpg?raw=true)

Applying this to the sum of sine waves at f = NF0 (N = 1:6) then adding a high pass filter at f = 3kHz returns the synthesized piano. [](https://github.com/carldegs/EE286-MP1/blob/master/piano_synth.jpg?raw=true)

**Snare**

The snare is a percussion instrument. This is illustrated in the spectrogram below, which unlike the piano or the trumpet, does not show peaks at multiples of an F0. However, there is a single band centered at 170Hz with a bandwidth of around 100Hz.

[](https://github.com/carldegs/EE286-MP1/blob/master/spectrogram_snare.png?raw=true)

The snare sample audio features a bright-sounding reverberation that decays quickly over time. The reverberation is due to the vibrations produced in the steel cables wired at the base of a snare drum.

**Karplus-Strong Drum Synthesis**

The timbre of the snare is difficult to replicate because it does not operate on harmonics. The Karplus-Strong algorithm was selected because of its ability to produce

A variant of the Karplus-Strong Algorithm [2] was used to synthesize the snare. The original variant is commonly used to synthesize guitar sounds, with the following averaging function:

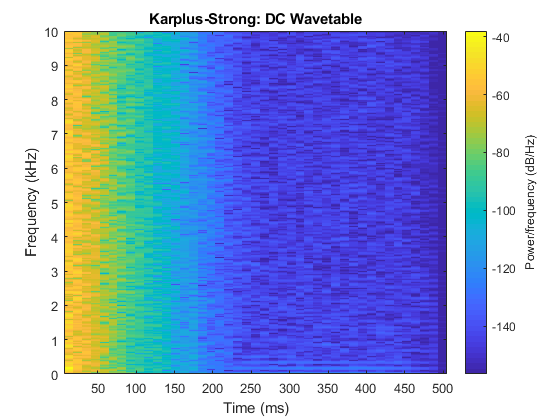
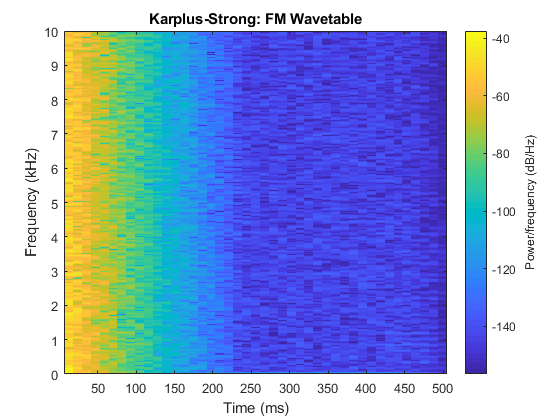
y[n] = 0.5\*(y\_p[n] + y\_p[n-1])

The algorithm is modified, so that the delay line averaging function changes its sign with probability p.

y[n] = 0.5\*(y\_p[n] + y\_p[n-1]), probability p

y[n] = -0.5\*(y\_p[n] + y\_p[n-1]), probability 1-p

The results in [2] already synthesize a sound with the ADSR properties of a drum - rapid attack, minimal sustain, and mostly decay. The wavetable used in [2] was a DC signal. This algorithm works well simulating drum sounds that don't resonate. However, in its simplest form, this does not replicate the vibrancy of the snare sound. To add complexity to the timbre, the wavetable was changed to an FM signal. The FM signal was designed to have the same center frequency and bandwidth as the measured snare spectrum. Another parameter that was tweaked was the stretch factor, also mentioned in [2]. The synthesizer then applies the ADSR envelope extracted from the original snare.

Two snare samples are synthesized from synthesis.m. The first sample was generated from a DC wavetable fed to a Karplus Strong delay line. The second sample used an FM wavetable instead with a center frequency at the 170Hz, where the original snare sample has an energy peak. The probability and stretch factor are determined from the experiments in snare.m; the same p and stretch\_factor values are applied to both samples. The spectrograms of the two samples are shown below:[](https://github.com/carldegs/EE286-MP1/blob/master/spectrogram_snare_dc.png) [](https://github.com/carldegs/EE286-MP1/blob/master/spectrogram_snare_fm.png)

**Results**

**Synthesized samples**

Synthesized Trumpet:

* [trumpet.wav](https://github.com/carldegs/EE286-MP1/blob/master/synth/trumpet.wav)

Synthesized Snare Clips:

* Snare with DC wavetable [snare\_karplus\_dc\_wavetable.wav](https://github.com/carldegs/EE286-MP1/blob/master/synth/snare_karplus_dc_wavetable.wav)
* Snare with FM wavetable [snare\_karplus\_fm\_wavetable.wav](https://github.com/carldegs/EE286-MP1/blob/master/synth/snare_karplus_fm_wavetable.wav)

Synthesized Piano:

* [piano.wav](https://github.com/carldegs/EE286-MP1/blob/master/synth/piano.wav)

**Recommendations**

To improve the naturalness of the pitched instruments, piano and trumpet, it is worth exploring the following:

* Calculating the residual signal after taking out the harmonics, and regenerating th
* Applying phase modulation
* Using FM signals instead of sines to create wider bands at the harmonic frequencies

The Karplus-Strong implementation here can be further extended to better approximate the snare by:

* Exploring other wavetables such as, noise, pink noise, sawtooth, square
* Adding a formant shaping filter to the delay line
* Using a multidimensional wavetable

This project is also available in: [https://github.com/carldegs/EE286-MP1](https://github.com/carldegs/EE286-MP1/tree/master/synth)

**References**

[1] Marshall, D. and Sidorov, K., "CM3106 Chapter 5: Digital Audio Synthesis", School of Computer Science & Informatics, Cardi University, UK

[2] Florian LE BOURDAIS, "Understanding the Karplus-Strong with Python (Synthetic Guitar Sounds Included)", 20 Apr 2017, <https://flothesof.github.io/Karplus-Strong-algorithm-Python.html>