AUDIO SYNTHESIS WITH SINE WAVES

FANG Jiakun
A0123777@u.nus.edu

ADDITIVE SYNTHESIS AND ADSR

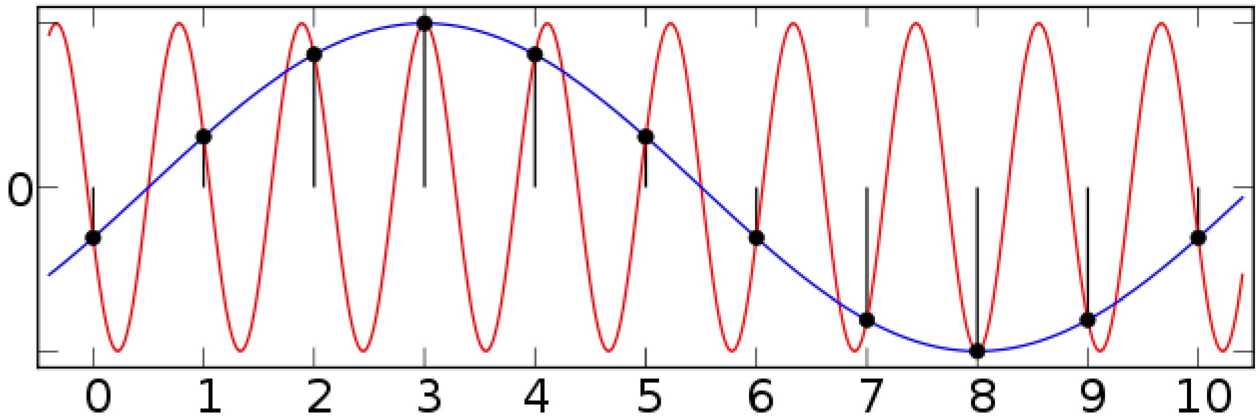
Add 2 sine waves

- Constructing a sine wave of frequency *f* Hz, sample rate *Fs* Hz, duration d seconds, phase φ (in radians), amplitude *A*, with $t = 0, 1, \ldots (d \cdot Fs)$.

$$y_t = A \sin\left(t \cdot \left(\frac{f}{F_s}\right) \cdot 2\pi + \phi\right)$$

- Tip: assume A1 = A2 = 1/ 2 = 0.5

Aliasing: an effect that causes different signals to become indistinguishable when sampled.



$Fs / 2 \geq$ highest frequency

Hint: please choose your harmonics wisely to avoid the aliasing.

ADDITIVE SYNTHESIS AND ADSR

Save the audio

- scipy.io.wavfile.write(filename, rate, data)
- Tip: use the .astype() method to convert the type to 16-bit integer (2^{-15}~2^{15}) if applicable
- Eg: (32767 * data).astype(numpy.int16)

Create a spectrogram

- Run db_spectrum on every buffer and only keep the positive frequencies

ADDITIVE SYNTHESIS AND ADSR - NOTE

We can map directly from sheet music notes to frequency

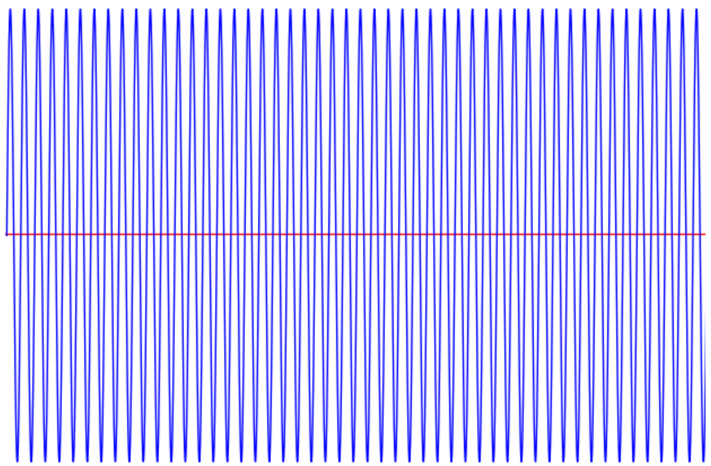
Given a note in MIDI pitch numbers (counting notes with "middle C" being number 60):

$$f = 440 \cdot 2^{\frac{m-69}{12}}$$

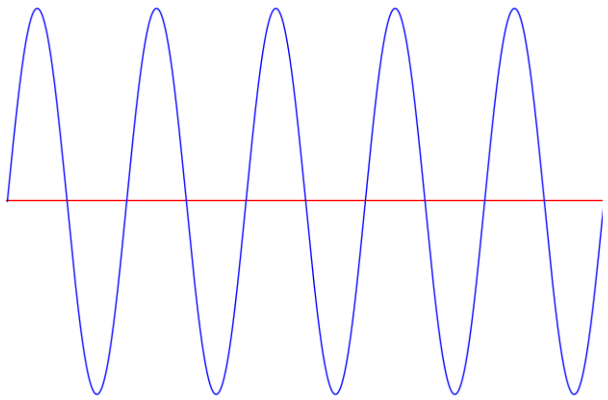
"C major scale"

ADDITIVE SYNTHESIS AND ADSR

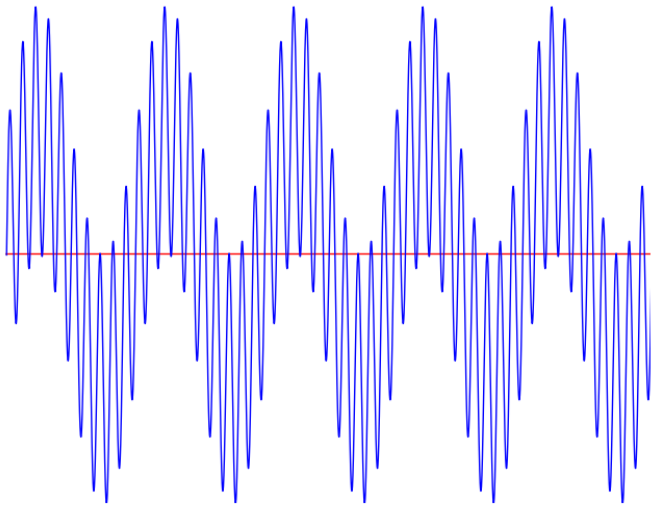
Define two sine waves at $Fs = 8000$, $A1 = A2 = 0.5$; $f1 = 100$, $f2 = 10$



+



=



ADDITIVE SYNTHESIS AND ADSR - HARMONICS

Each of these sine waves has a frequency that is an integer multiple of the fundamental frequency, and we call these the 'harmonics' of the sound.

For natural sounds, the energy of each upper harmonic generally decreases, e.g., given the amplitude of wave *n* at time *t* as *An(t)*,

$$y_t = A \sin \left(t \cdot \left(\frac{f}{F_s} \right) \cdot 2\pi + \phi \right)$$

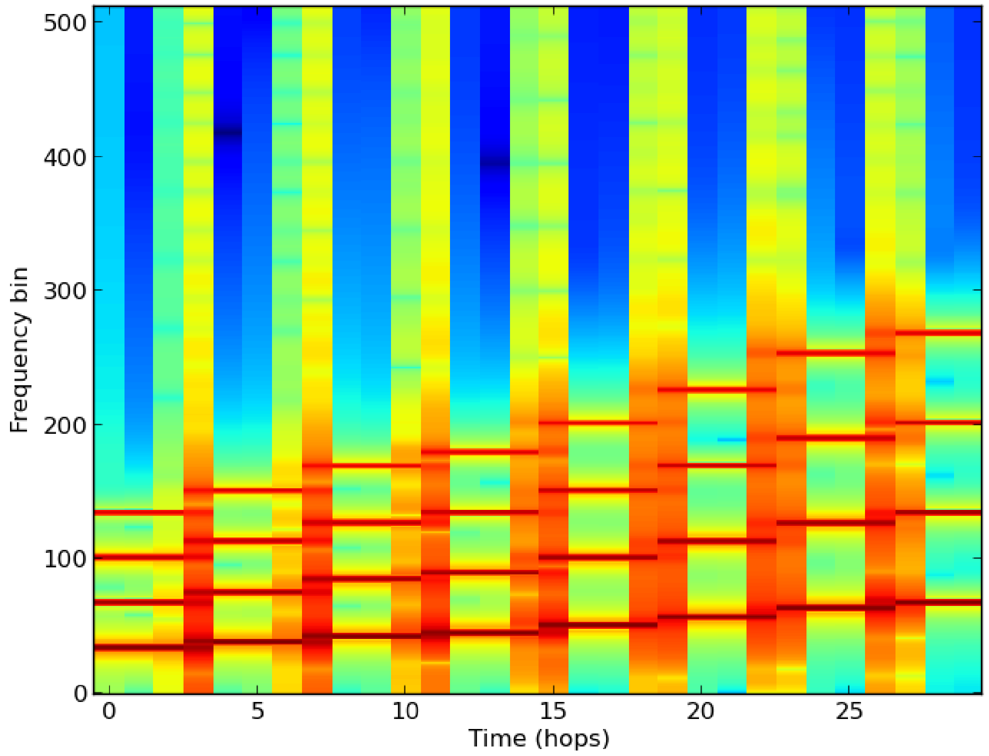
$$A_0(0) = 1.0 \qquad A_1(0) = 0.5$$

$$A_2(0) = 0.25 \qquad A_3(0) = 0.125$$

$$A_n(0) = \frac{1}{n+1}$$

ADDITIVE SYNTHESIS AND ADSR - SPECTROGRAM

A visual representation of sound that displays the amplitude of the frequency of the sound over time.



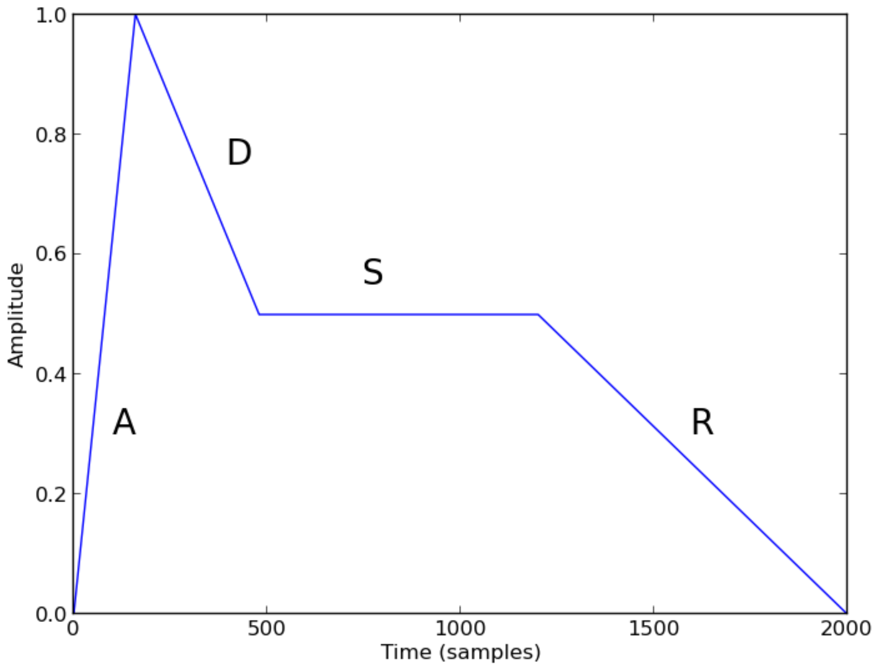
pylab.imshow(data, origin='lower', aspect='auto', interpolation='nearest')

ADDITIVE SYNTHESIS AND ADSR - SPECTROGRAM

Back to Assignment 4

- def db_spectrum (time_domain_data, window):
- fft = scipy.fftpack.fft(window * time_domain_data)
- # only keep the positive frequencies
- fft = fft [: len (fft)/2+1]
- # magnitude sprectrum , no normalization
- magfft = abs (fft)
- # log - magnitude
- epsilon = 1e -10
- db = 20* numpy . log10 (magfft + epsilon)
- return db

ADDITIVE SYNTHESIS AND ADSR



a diagram of an ADSR envelope

Attack phase

- The attack phase begins when the MIDI “note on” message is received – when the key is pressed or the note is encountered in a sequencer.

Decay phase

- The decay phase defines how long the note will take to reach a settled volume after hitting the attack peak.

Sustain phase

- The sustain phase is an important one to understand because *the envelope does not determine its duration*.

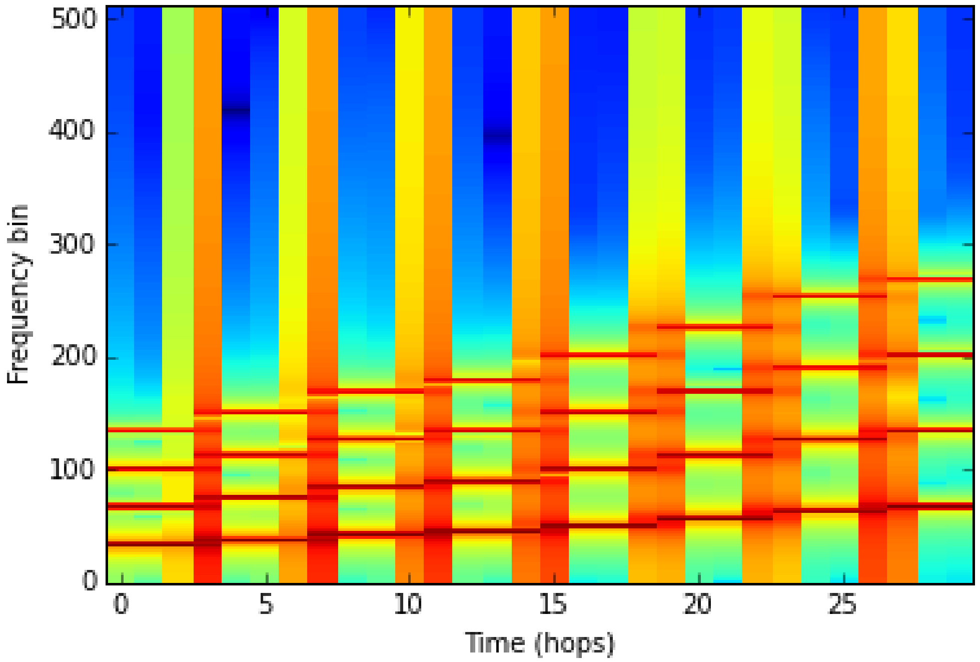
Release phase

- Once the note has been released, the envelope enters its release phase. The line of the envelope here defines how long the note’s volume will take to fade to zero

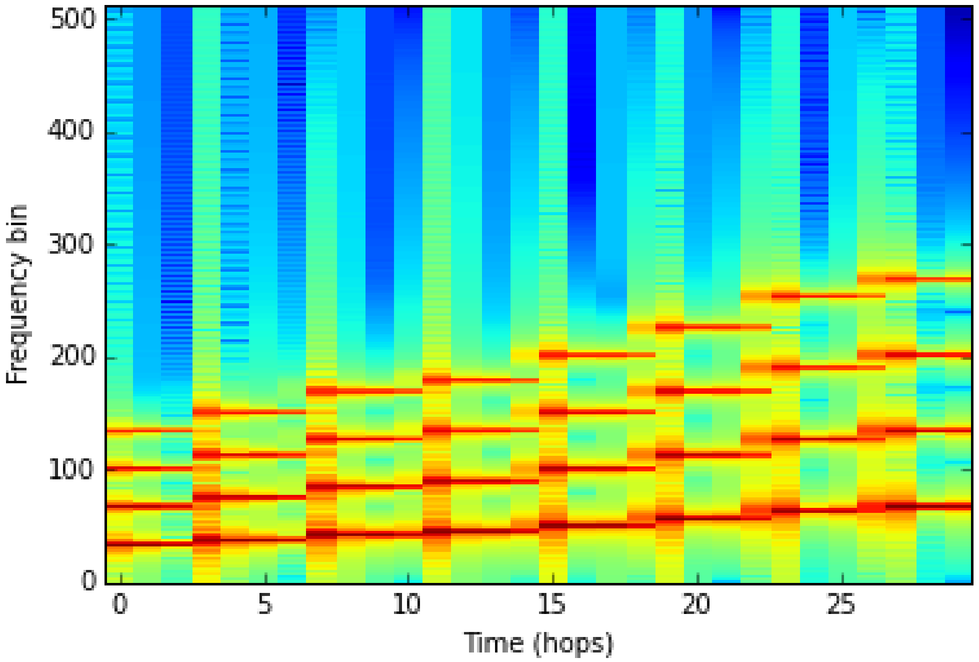
http://www.gavinorland.com/tutorials/what-is-an-adsr-envelope/

ADDITIVE SYNTHESIS AND ADSR

Without ADSR



With ADSR



Problem: as suggested by the spectrogram, there are occasional "clicks" as the sine waves change frequencies.