CS 591 S1 – Computational Audio -- Spring, 2016

Wayne Snyder
Computer Science Department
Boston University

Today
- Pitch Detection using autocorrelation and “peak picking” and parabolic interpolation
- Time Domain vs Frequency Domain
- The Naïve Discrete Fourier Transform (not accounting for phase): developing the algorithm from first principles (starting with multiplying sine waves)

Next Time:
- A brief introduction to complex numbers and phasors
- Finding phase in the Fourier Transform

Pitch Determination using Autocorrelation

From last time.....

def acorr(X, lag):
    sum = 0
    for i in range(len(X) - lag):
        sum += X[i]*X[i+lag]
    return sum/(len(X)-lag)
Pitch Determination using Autocorrelation

When you have a simple sine wave and the perfect lag, you are essentially squaring the wave (cf. HW 02) and this maximizes the sum of products.

We will return to multiplying sine waves in the second part of the lecture, as a warmup to the Fourier Transform....

Pitch Determination using Autocorrelation

When we graph the autocorrelation coefficient for various lag times, the peaks in the curve show where the correlation between the lagged signal is strongest, which corresponds to frequency F0. Let’s look at some (artificial) examples.

Spectrum:

\[ (441, 1.0, 0) \]
Pitch Determination using Autocorrelation

Spectrum:

\[ \left( 441, 1.0, \pi \right) \]

Note that the phase does not matter!

Pitch Determination using Autocorrelation

Spectrum:

\[ \left( 441, 1.0, 2.3 \right) \]

Note that the phase does not matter!
Pitch Determination using Autocorrelation

Spectrum (with even harmonics):

\[
\left[(441, 0.6, 0),
(441*3, 0.2, 0),
(441*5, 0.2, 0)\right]
\]

Peak found at 
100 1.0 = 441.0 Hz
200 1.0 = 220.5 Hz
300 1.0 = 147.0 Hz
400 1.0 = 110.25 Hz

F0 = 441.0

Pitch Determination using Autocorrelation

Spectrum:

\[
\left[(110, 1.0, 0)\right]
\]

Peak found at 
406 (1.0001) = 189.4293 Hz
685 (1.001) = 54.7826 Hz

F0 = 189.4292839782234
Pitch Determination using Autocorrelation

Signal Window for X

Autocorrelation

Bell Spectrum:

Steel String Spectrum:
Pitch Determination using Autocorrelation

Now let's try some actual music signals... Cello.wav at 11.35

Signal Window for X

Autocorrelation

Peaks found at lags
627 (-8.9778) = 183.2787 Hz
65 (-8.3434) = 74.1136 Hz
535 (-8.9435) = 51.1789 Hz
1293 (8.9669) = 34.3726 Hz
5710 (1.4005) = 25.7895 Hz
5137 (-8.9525) = 36.0364 Hz

Pitch Determination using Autocorrelation

WTC1_01.wav at 0.3 sec

Signal Window for X

Autocorrelation

Peaks found at lags
84 (-9.1452) = 925.9 Hz
256 (-9.1593) = 252.9 Hz
712 (-9.1839) = 375.9 Hz
85 (-9.2112) = 212.9 Hz
421 (-8.2574) = 184.7586 Hz
385 (-8.23) = 87.2537 Hz
560 (-8.1134) = 74.0727 Hz
573 (-8.7926) = 51.3075 Hz
Pitch Determination using Autocorrelation

**SteelString.wav at 0.5 sec (220 Hz)**

![Signal Window for X](image1)

**Autocorrelation**

Peaks found at lags:
- 100: -9.2756 Hz
- 200: -9.5993 Hz
- 300: -9.2277 Hz
- 400: -9.2277 Hz
- 500: -9.2277 Hz
- 600: -9.2277 Hz
- 700: -9.2277 Hz
- 800: -9.2277 Hz
- 900: -9.2277 Hz

**SteelString.wav at 0.006 sec (220 Hz)**

![Signal Window for X](image2)

**Autocorrelation**

Peaks found at lags:
- 200: -9.0293 Hz
- 211: -9.3225 Hz
- 400: -9.5379 Hz
- 412: -9.8642 Hz
- 517: -10.7383 Hz
- 511: -10.7383 Hz

- Bell Spectrum: 220 Hz
- SteelString.wav at 0.5 sec: 220 Hz
- SteelString.wav at 0.006 sec: 220 Hz
Pitch Determination using Autocorrelation

SteelString.wav at 0.006 sec for 0.015 seconds duration

Peaks found at lags
-2 (8.8036) = 3675.0 Hz
-11 (-8.7656) = 397.2973 Hz
-91 (-8.61391) = 238.8981 Hz
-99 (-8.7676) = 228.5 Hz
-113 (-8.6664) = 287.0423 Hz

Pitch Determination using Autocorrelation

SteelString.wav at 0.006 sec for 0.01 seconds duration

Peaks found at lags
-11 (8.7624) = 3302.877 Hz
-9 (-8.8037) = 882.6 Hz
-7 (-8.2824) = 650.289 Hz
-47 (-8.59286) = 412.1095 Hz
-49 (-8.8386) = 295.9732 Hz
-75 (-8.3131) = 252.8 Hz
Pitch Determination using Autocorrelation

**Genesis01.wav at 2.87 sec**

![Signal Window for X](image1)

![Autocorrelation](image2)

Peaks found at lags:

- 16: 2756.23 Hz
- 78: 139.8 Hz
- 111: 297.287 Hz
- 120: 244.521 Hz
- 149: 295.973 Hz
- 164: 221.666 Hz
- 212: 289.489 Hz
- 303: 153.936 Hz
- 341: 251.162 Hz
- 407: 188.353 Hz

**Pitch Determination using Autocorrelation**

**Beethoven.Seventh.wav at 3.125**

![Signal Window for X](image3)

![Autocorrelation](image4)

Peaks found at lags:

- 68: 1754.8 Hz
- 70: 882.8 Hz
- 103: 588.0 Hz
- 108: 430.637 Hz
- 125: 352.8 Hz
- 311: 292.853 Hz
- 316: 258.448 Hz
Pitch Determination using Autocorrelation

**Bell.wav at 0.03**

![Graph showing signal window and autocorrelation for Bell.wav]({#image_url#})

Peaks found at lags:
- 5: 2606 Hz
- 9: 4000 Hz
- 14: 3516 Hz
- 19: 2376 Hz
- 23: 1752 Hz
- 28: 1200 Hz
- 32: 857 Hz
- 36: 505 Hz
- 42: 253 Hz
- 47: 127 Hz

**Cymbal.wav at 0.3 sec**

![Graph showing signal window and autocorrelation for Cymbal.wav]({#image_url#})

Peaks found at lags:
- 11: 4090 Hz
- 19: 2321 Hz
- 25: 1516 Hz
- 26: 1479 Hz
- 30: 1329 Hz
- 32: 1106 Hz
- 38: 849 Hz
- 41: 670 Hz
- 45: 505 Hz
- 50: 353 Hz
- 55: 253 Hz
- 60: 180 Hz
- 74: 127 Hz
Pitch Determination using Autocorrelation

HigherLove.wav at 1.143 sec

Digital Audio Fundamentals: Fourier Transform

Recall that we can view a signal in two ways: as a graph of amplitudes vs time (the Time Domain) or (because all periodic waves can be expressed as the sum of simple component waves) as a graph of frequencies vs amplitudes (the Frequency Domain):

We can transform signals from one domain to the other (approximately) using the Fourier Transforms.

Fourier Analysis: Time Domain (wav file) => Frequency domain (spectrum)
Fourier Synthesis: Frequency Domain => Time Domain (e.g. HW01.P3)
In a sense we are considering the sum of component waves from two different directions in three-dimensional space:

We are ignoring for the present the fact that musical signals change significantly over time; assume that the “window” of time in which we do analysis is short enough (e.g. 1/10 sec) that the frequencies are stable enough to analyze. This will be a significant complication later on...

We will investigate Fourier Analysis first by considering a simple case first. We wish to determine the spectrum from a wave file, under the assumption that the wave file (i) is composed of sine wave of various frequencies all of phase 0, and (ii) that the component sine waves do not change amplitude in the wave file. Similarly, we could consider a window of N samples in the middle of a wave file. We will create a text file giving the “bar graph” of the spectrum.
Let's visualize again at what happens when we square a wave:

Note that the product has twice the frequency, and all values are positive!

Let's try general multiplication with same phase, different frequencies:

The faster frequency “flips” the lower frequency back and forth over the 0 line in a series of smaller sinusoids.

Note that the product is shown on a larger scale; it has the same max amplitude and length!
Product of waves of same amplitude, same phase, different frequencies:

Notice that for different frequencies, the product signal flips back and forth over the 0 line, spending about as much time above as below..
Product of waves of same amplitude, same phase, different frequencies:

Notice that for different frequencies, the product signal flips back and forth over the 0 line, spending about as much time above as below..
Product of waves of same amplitude, same phase, different frequencies:

Notice that for different frequencies, the product signal flips back and forth over the 0 line, spending about as much time above as below..
Product of waves of same amplitude, same phase, different frequencies:

BUT at equal frequencies, the curve is always positive...

What happens in the limit as we approach the same frequency?

Product of waves of same amplitude, same phase, different frequencies:

What happens in the limit as we approach the same frequency?

Does the product gradually become more positive, or does it flip suddenly when the freqs are equal?
Digital Audio Fundamentals: Multiplying Sine Waves

Product of waves of same amplitude, same phase, different frequencies' summing the samples over the interval:

Does the product gradually become more positive, or does it flip suddenly when the freqs are equal?

Let's measure by summing the samples in the product!

Digital Audio Fundamentals: Multiplying Sine Waves

Product of waves of same amplitude, same phase, different frequencies' summing the samples over the interval:

Does the product gradually become more positive, or does it flip suddenly when the freqs are equal?

Let's sum the samples over the interval!
Digital Audio Fundamentals: Multiplying Sine Waves

Product of waves of same amplitude, same phase, different frequencies' summing the samples over the interval:

Does the product gradually become more positive, or does it flip suddenly when the freqs are equal?

Let's sum the samples over the interval!
Product of waves of same amplitude, same phase, different frequencies' summing the samples over the interval:

Does the product gradually become more positive, or does it flip suddenly when the freqs are equal?

Let's sum the samples over the interval!
Digital Audio Fundamentals: Multiplying Sine Waves

Product of waves of same amplitude, same phase, different frequencies' summing the samples over the interval:

Does the product gradually become more positive, or does it flip suddenly when the freqs are equal?

Let's sum the samples over the interval!
Product of waves of same amplitude, same phase, different frequencies' summing the samples over the interval:

Does the product gradually become more positive, or does it flip suddenly when the freqs are equal?

Let's sum the samples over the interval!
Product of waves of same amplitude, same phase, different frequencies' summing the samples over the interval:

Does the product gradually become more positive, or does it flip suddenly when the freqs are equal?

Let's sum the samples over the interval!
Digital Audio Fundamentals: Multiplying Sine Waves

Product of waves of same amplitude, same phase, different frequencies’ summing the samples over the interval:

Does the product gradually become more positive, or does it flip suddenly when the freqs are equal?

Let’s sum the samples over the interval!
Digital Audio Fundamentals: Multiplying Sine Waves

Product of waves of same amplitude, same phase, different frequencies’ summing the samples over the interval:

Looks like it flips suddenly from near 0 to very positive when frequencies are identical.

Digital Audio Fundamentals: Multiplying Sine Waves

If we work with much smaller gradations, we see better what happens; here is a graph of the sum of the product wave over 100 samples, with a fixed frequency 20, and the second wave’s frequency on the x_axis:

Thus, multiplying sin waves gives us a way to identify a single wave: when we hit exactly the right frequency, we’ll get a spike in the sum of the amplitudes in the product wave.
Moreover, it gives us the amplitude of the single wave, due to this fact about the sin function in the discrete case:

The sum of N samples of two sine waves of the same frequency and of amplitudes A1 and A2 is equal to $A1 \times A2 \times N / 2$ (can derive from “half angle formula” from HW02 solution).
Digital Audio Fundamentals: Multiplying Sine Waves

To simplify, we’ll make our “probe wave” of amplitude 1, and thus, the amplitude of our “test wave” is just \( 2 \times \text{Sum} / \text{Number of Samples} \):

\[
2 \times 20 / 100 = .4
\]

Digital Audio Fundamentals: Discrete Sine Transform

We will use “probe waves” which are sine waves which start at phase 0 at the beginning of the window, and complete an integral number of cycles at the end of the window; hence the only component waves we can detect have exactly these frequencies:

- 1 Cycle
- 2 Cycles
- 3 Cycles … Etc.

up to Nyquist limit.
Finally, our Naïve Discrete Fourier Transform algorithm analyzes a sequence of N samples by performing the same operation using integral probe frequencies of 1 .. N/2 for the time duration of the sample space (due to the Nyquist Limit). By calculating $2^\text{Sum/Number of samples}$ we can calculate the amplitudes.

Windowing effects: How does the choice of window size affect the results?

In general, we will be working in a window of N samples at some point in the complete wav file (or array):