Digital Signal Processing

Lecture 7 – Autocorrelation

상명대학교
컴퓨터과학과
강상욱 교수

Definition of correlation

- Very useful tool for signal analysis.
  - The analysis of autocorrelation is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise.
  - Identifying the missing fundamental frequency in a signal implied by its harmonic frequencies.
- Correlation between variables
  - If you know the value of one, you have some information about the other.
  - Pearson product-moment correlation coefficient (to quantify the correlation)

$$\rho = \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{N\sigma_x\sigma_y}, \qquad -1 \le \rho \le 1$$

μ_x, μ_y : mean of x and y

σ_x, σ_y : standard deviation of x and y

The meaning of correlation

- If ρ is positive,
  - The correlation is positive
  - When one variable is high, the other tends to be high.
  - And, vice versa.
- The magnitude of ρ
  - The strength of the correlation.
  - If $\rho = 1 \ or \ -1$, the variables are perfectly correlated and if you know one, you can make a perfect prediction about the other.
  - If $\rho \sim 0$, the correlation is probably weak.
  - "probably weak" because it is also possible that there is a nonlinear relationship that is not captured by the coefficient of correlation.
- Correlation in Python

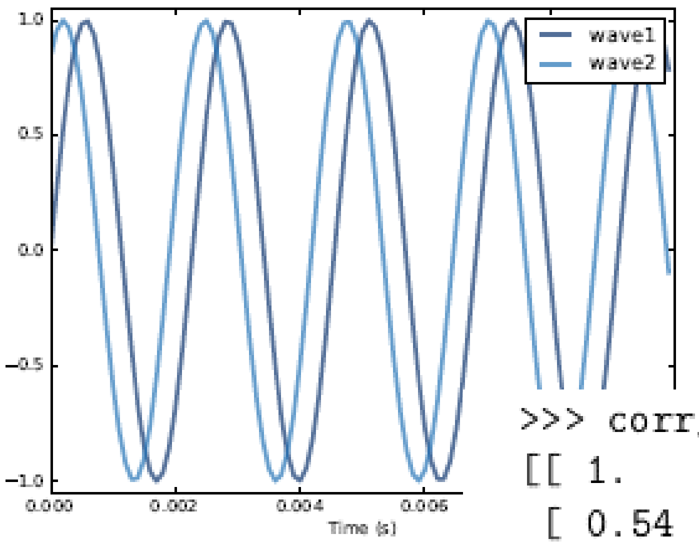
```
>>> from numpy import *
>>> T = array([1.3, 4.5, 2.8, 3.9]) # temperature measurements
>>> P = array([2.7, 8.7, 4.7, 8.2]) # corresponding pressure measurements
>>> print corrcoef([T,P]) # correlation matrix of temperature and pressure
[[ 1. 0.98062258]
 [ 0.98062258 1. ]]
```

Correlation example 1

```
def make_sine(offset):
    signal = thinkdsp.SinSignal(freq=440, offset=offset)
    wave = signal.make_wave(duration=0.5, framerate=10000)
    return wave
```

```
wave1 = make_sine(offset=0)
wave2 = make_sine(offset=1)
```

make_sine(offset) constructs sine waves with different phase offsets




```
>>> corr_matrix = np.corrcoef(wave1.ys, wave2.ys, ddof=0)
[[ 1.    0.54]
 [ 0.54  1.  ]]
```

The option `ddof=0` indicates that `corrcoef` should divide by N, as in the equation above, rather than use the default, $N-1$.

Correlation example 2

As the offset increases, the correlation decreases until the waves are 180 degrees (π) out of phase, which yields correlation -1

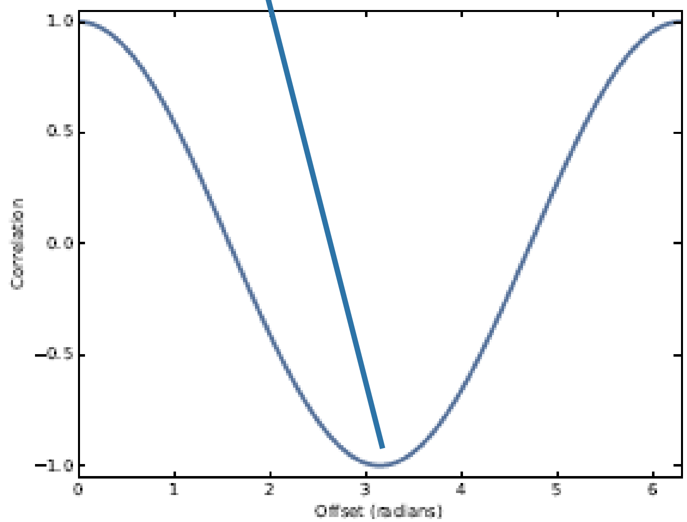


Figure 5.2: The correlation of two sine waves as a function of the phase offset between them. The result is a cosine.

`thinkdsp` provides a simple interface for computing the correlation between waves:

```
>>> wave1.corr(wave2)
0.54
```

Serial correlation I

- The correlation between each element and the next (or the previous).
  - To compute it, we can shift a signal and then compute the correlation of the shifted version with the original.

```
def serial_corr(wave, lag=1):
    n = len(wave)
    y1 = wave.ys[lag:]
    y2 = wave.ys[:n-lag]
    corr = np.corrcoef(y1, y2, ddof=0)[0, 1]
    return corr
```

`serial_corr` takes a Wave object and `lag`, which is the integer number of places to shift the wave. It computes the correlation of the wave with a shifted version of itself.

Serial correlation 2

```
signal = thinkdsp.UncorrelatedGaussianNoise()
wave = signal.make_wave(duration=0.5, framerate=11025)
serial_corr(wave)
```

The result value will be small or large? -- Yes, comparably small (0.06). Why?

```
signal = thinkdsp.BrownianNoise()
wave = signal.make_wave(duration=0.5, framerate=11025)
serial_corr(wave)
```

How about the Brownian noise? -- Yes, comparably large (0.999). Why?

```
signal = thinkdsp.PinkNoise(beta=1)
wave = signal.make_wave(duration=0.5, framerate=11025)
serial_corr(wave)
```

How about pink noise? -- Yes, inbetween Brownian and UU noise (0.851). Why?

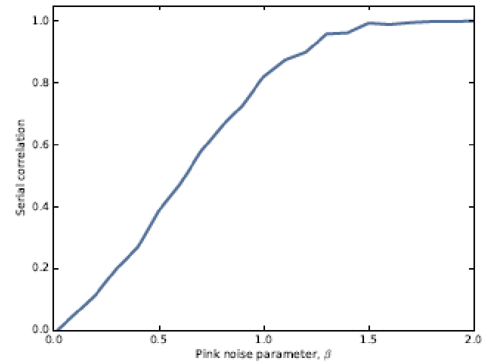


Figure 5.3: Serial correlation for pink noise with a range of parameters.

Autocorrelation I

- Definition
  - Serial_corr can be thought as a function that maps from each value of lag to the corresponding correlation.
  - We can evaluate that function by looping through values of lag.

```
def autocorr(wave):
    lags = range(len(wave.ys)//2)
    corrs = [serial_corr(wave, lag) for lag in lags]
    return lags, corrs
```

`autocorr` takes a Wave object and returns the autocorrelation function as a pair of sequences: `lags` is a sequence of integers from 0 to half the length of the wave; `corrs` is the sequence of serial correlations for each lag.

Autocorrelation 2

- Definition
  - Serial_corr can be thought as a function that maps from each value of lag to the corresponding correlation.
  - We can evaluate that function by looping through values of lag.

```
def autocorr(wave):
    lags = range(len(wave.ys)//2)
    corrs = [serial_corr(wave, lag) for lag in lags]
    return lags, corrs
```

`autocorr` takes a Wave object and returns the autocorrelation function as a pair of sequences: `lags` is a sequence of integers from 0 to half the length of the wave; `corrs` is the sequence of serial correlations for each lag.

Autocorrelation example

For low values of β, the signal is less correlated.

For high values of β, the serial correlation is stronger and drops off more slowly, which is called "long-range dependence

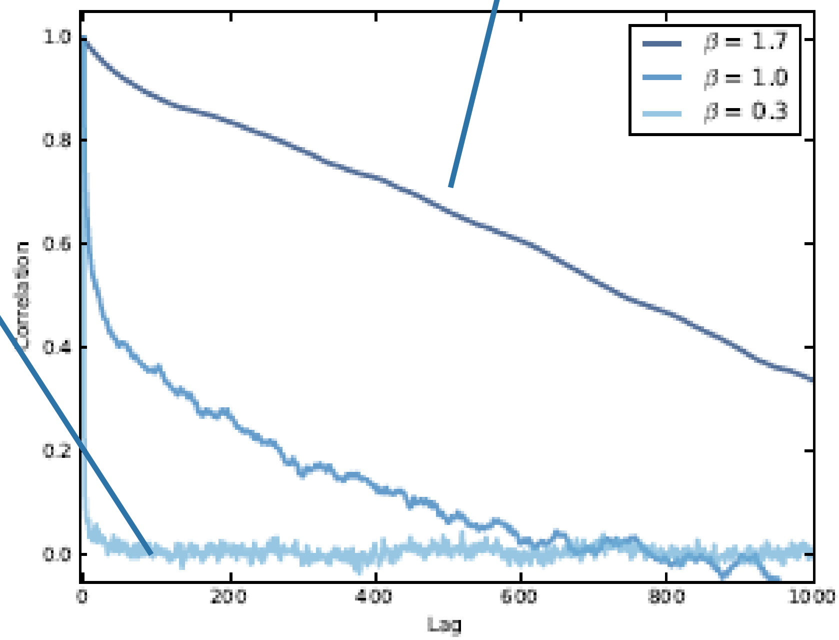
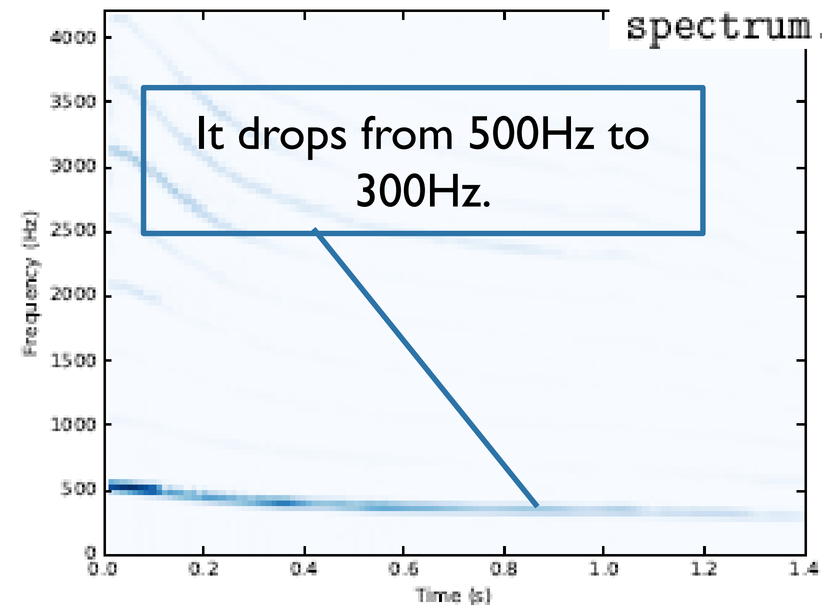


Figure 5.4: Autocorrelation functions for pink noise with a range of parameters.

Autocorrelation of periodic signals 1

- A chirp that starts near 500Hz and drops down to about 300Hz.
  - 28042__bcjordan__voicedownbew.wav : listen to this

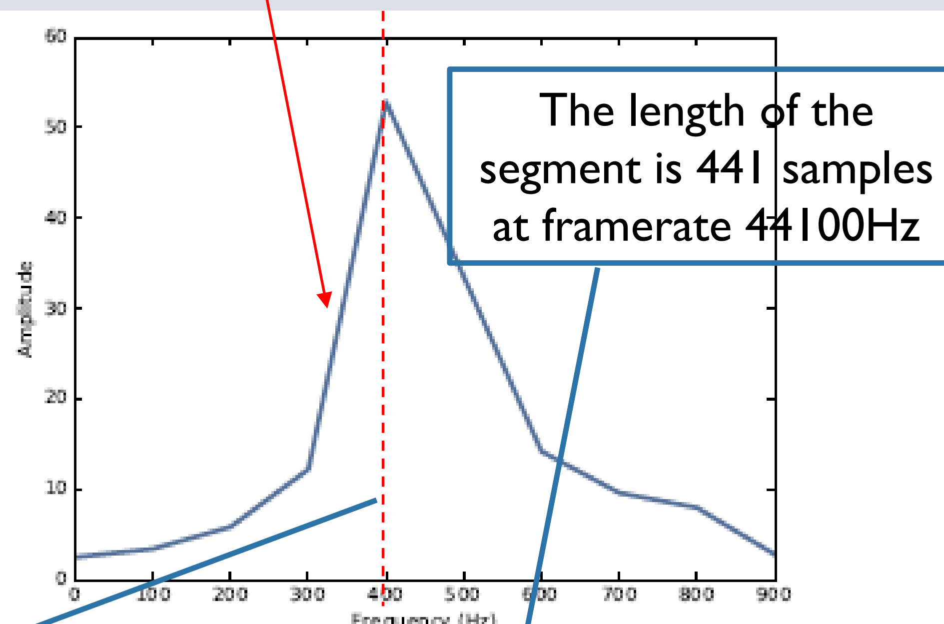
```
duration = 0.01
segment = wave.segment(start=0.2, duration=duration)
spectrum = segment.make_spectrum()
spectrum.plot(high=1000)
```



It drops from 500Hz to 300Hz.

Figure 5.5: Spectrogram of a vocal chirp.

A clear peak near 400Hz



The length of the segment is 441 samples at framerate 44100Hz

Figure 5.6: Spectrum of a segment from a vocal chirp.

Autocorrelation of periodic signals 2

- Is the peak 400Hz precisely?
  - The freq. resolution : 100Hz (Why?)
  - The estimated peak might be off by 50Hz.
  - The peak ranges from 350Hz to 450Hz.
  - We could get better freq. resolution by taking a longer segment, but pitch is changing over time.
- We can estimate the pitch more precisely using autocorrelation.
  - If a signal is periodic, the autocorrelation spikes when the lag equals the period.

Finding period using autocorrelation I

```
def plot_shifted(wave, offset=0.001, start=0.2):
    thinkplot.preplot(2)
    segment1 = wave.segment(start=start, duration=0.01)
    segment1.plot(linewidth=2, alpha=0.8)

    segment2 = wave.segment(start=start-offset, duration=0.01)
    segment2.shift(offset)
    segment2.plot(linewidth=2, alpha=0.4)

    corr = segment1.corr(segment2)
    text = r'$\rho =$ %.2g' % corr
    thinkplot.text(segment1.start+0.0005, -0.8, text)
    thinkplot.config(xlabel='Time (s)')
```

```
lags, corrs = autocorr(segment)
thinkplot.plot(lags, corrs)
```

The result is shown in Figure 5.8

Finding period using autocorrelation 2

The first peak occurs at lag=101.

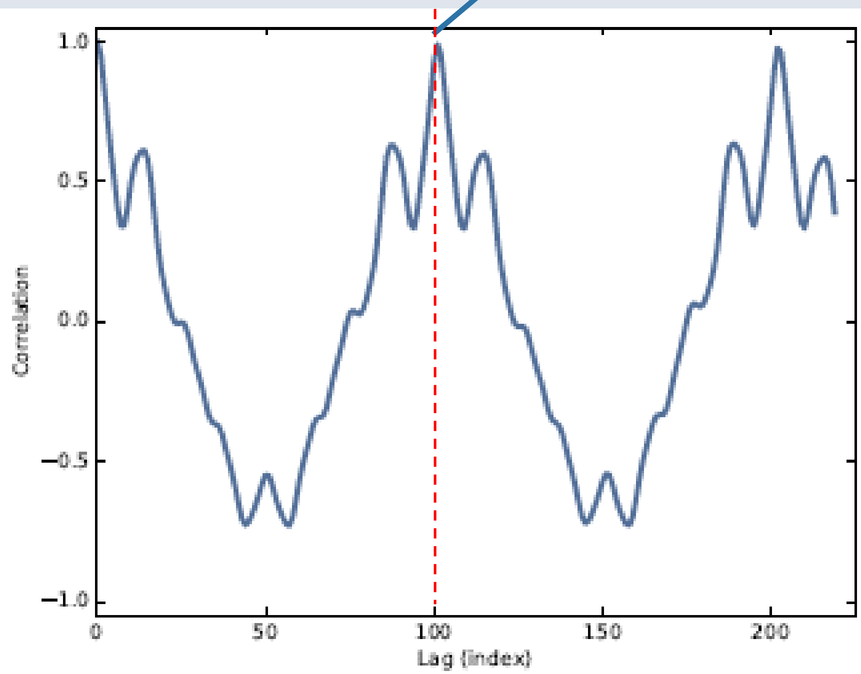


Figure 5.8: Autocorrelation function for a segment from a chirp.

* What is the freq. of the segment?

period = lag / segment.framerate
= 101/44100
freq. = 1/period = 437 Hz

* What is the freq. precision?

If lag=100 → 432 Hz
If lag=102 → 441 Hz, so <10 Hz

Increased framerate incurs increased freq. precision, which contradicts Gabor limit.

* Comparison of the freq. precision?
432 – 437 – 441
350 – 400 – 450

Correlation as a dot product

- In signal processing, unbiased and normalized signals are often used, where the mean is 0 and the standard deviation is 1.
  - $\rho = \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{N\sigma_x\sigma_y} = \frac{\sum_i x_i y_i}{N}$, or
  - $r = \sum_i x_i y_i$
  - If x and y is in the form of vector, r is the dot product.
  - $cos\theta = \frac{x \cdot y}{\|x\|\|y\|}$,where θ is the angle between the vectors. Also see Figure 5.2 which is a cosine wave.

Using Numpy

```
corrs2 = np.correlate(segment.ys, segment.ys, mode='same')
```

The option `mode` tells `correlate` what range of `lag` to use. With the value `'same'`, the range is from $-N/2$ to $N/2$, where N is the length of the wave array.

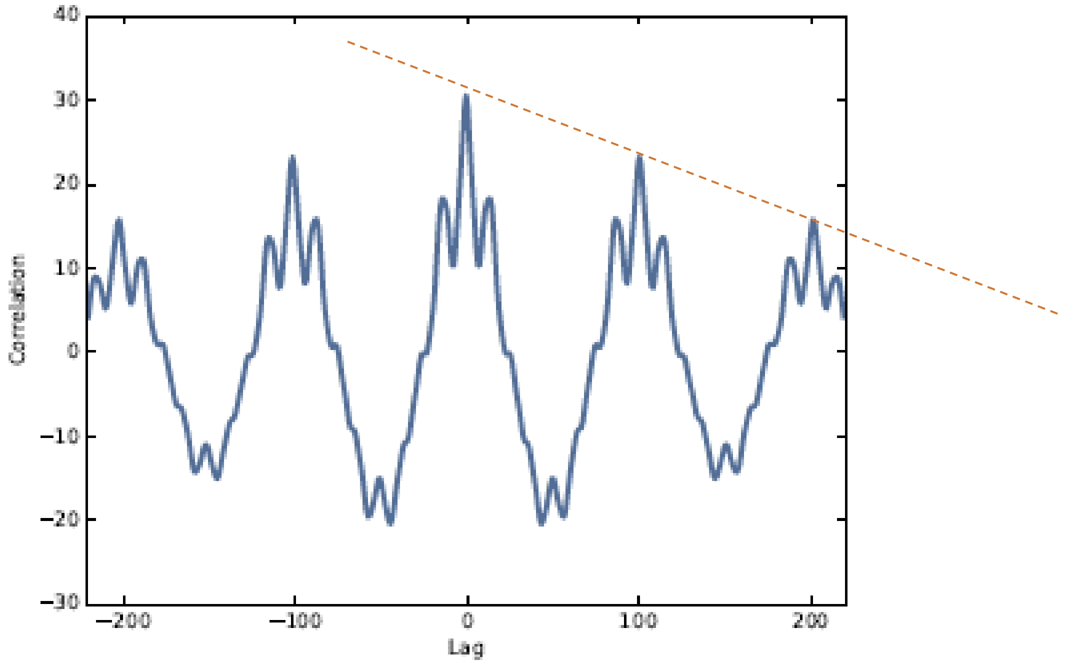


Figure 5.9: Autocorrelation function computed with `np.correlate`.

Using Numpy

```
corrs2 = np.correlate(segment.ys, segment.ys, mode='same')
```

The option `mode` tells `correlate` what range of `lag` to use. With the value `'same'`, the range is from $-N/2$ to $N/2$, where N is the length of the wave array.

- We can correct the decreasing correlation with the following.

Divide the correlation by gradually decreasing numbers

```
lengths = range(N, N//2, -1)
half /= lengths
```

Normalize the result so the correlation with lag=0 is 1.

```
half /= half[0]
```