# Tuesday Warm Up, Unit 2: Applications

Prof. Jordan C. Hanson

April 29, 2025

## 1 Memory Bank

1. **Cross-correlation**. Cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other. For discrete, finite signals $x[n]$ and $y[n]$, each defined for $n = 0, 1, \ldots, N-1$, is given by:

$$r_{xy}[k] = \sum_{n=0}^{N-1} \bar{x}[n] \cdot y[n+k]$$

for those values of $k$ where the summation is well-defined (i.e., $n+k$ lies within the valid range of $y[n]$). The bar above the signal $x[n]$ represents the complex conjugate. If $x$ and $y$ are real, $\bar{x} = x$ and $\bar{y} = y$.

2. **Cross-correlation and the FFT**. Let $\mathcal{F}(x[n])$ represent the DFT of $x[n]$. The DFT of the cross-correlation of $x[n]$ and $y[n]$ is

$$\mathcal{F}(r_{xy}[k]) = \overline{\mathcal{F}(x[n])} \cdot \mathcal{F}(y[n]) \quad (1)$$

That is, the complex conjugate of the DFT of $x[n]$ times the DFT of $y[n]$ is the DFT of the cross-correlation.

3. Like FFT convolution, we can use the above theorem to accelerate cross-correlation calculations. After computing the right hand side of Eq. 1, take the real part of the `ifft()` to find $r_{xy}[n]$.

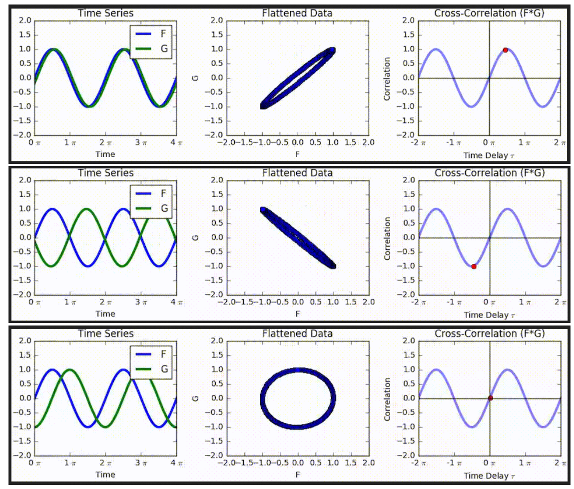

Figure 1: (Top) Nearly ideally correlated signals. (Middle) Nearly ideally anti-correlated signals. (Bottom) Uncorrelated signals. Adapted from **Wikipedia: Cross-correlation**.

## 2 Cross-Correlation and the FFT

1. Using the `xcorr` function from the `octave` package `signal`, (a) write code that calculates the cross-correlation between a sine and cosine wave. (b) Show that when the signals are *in phase*, the cross-correlation is maximized. (c) Show that when the signals are *out of phase*, the cross-correlation is mimimized. (d) What phase leads to no correlation? *Hint, consult Fig. 1.*

2. For the same reason that the basic convolution is computationally slow, cross-correlation is also computationally slow. In this exercise, FFT cross-correlation will be compared to `xcorr`. Use the following code to create and normalize two gaussian pulses:

```
clear;
close;
home;
pkg load signal
%Define a gaussian pulse
fs = 100.0e6; %Hz
dt = 1/fs; %seconds
T = 100e-6; %seconds
t = dt:dt:T; %seconds
mu1 = 10e-6; %seconds
s1 = 1.0e-6; %seconds
f_c = 1.0e6; %Hz
mu2 = 40e-6; %seconds
s2 = s1;
signal_1 = cos(2*pi*f_c*t).*exp(-0.5*(t-mu1).^2./s1^2);
signal_2 = cos(2*pi*f_c*t).*exp(-0.5*(t-mu2).^2./s2^2);
signal_1 = signal_1/sqrt(sum(signal_1.*signal_1));
signal_2 = signal_2/sqrt(sum(signal_2.*signal_2));
```

3. Using `[R,lag] = xcorr(signal_2,signal_1)`, find the maxmimum correlation coefficient and the corresponding lag between `signal_1` and `signal_2`.

4. Repeat the procedure using FFT cross-correlation. Remember to take the *real* part of the `ifft` of the product of the `fft()` of the signal before analyzing it. *Hint:*

```
R_fft = real(ifft(conj(fft(signal_1)).*fft(signal_2)));
```

5. Show that both the standard and FFT cross-correlation produce consistent results.