MATLAB function circcorr.m

The MATLAB function `circcorr.m` is provided for computing the auto- and cross-correlations necessary for AAE575 Homework 1 and 2. This function computes the correlation between two discrete-time arrays, $x[k]$, $y[k]$, for all relative delays between them. This provides a numerical approximation of the continuous-time cross-correlation

$$R(\gamma) = \frac{1}{T_I}\int_0^{T_I} x(t)y(t+\gamma)dt \tag{1}$$

When provided with the sample period, `circcorr.m` will compute the delay in the same time dimensions as this period.

To illustrate the use of `circcorr.m`, a set of example data are provided in the MATLAB file `circcorrex.mat`. The array "x" contains samples of a BOC(1,1) signal, collected at a sample rate of **8 MHz** . The array is **7821** samples long, representing data collected for $(7821/8E6) =$ **9.7762E-4 sec.**. Before we look at generating the autocorrelation of this signal, the first 5 microseconds are plotted in figure 1 superimposed on the continuous-time BOC(1,1) signal. Observe that the sampling is not synchronous with the BOC(1,1) signal sub-carrier. In other words, the samples ('X') are not aligned with the start and end of each sub-carrier pulse. This should make sense, since the sample rate (8 MHz) is not an integer multiple of the sub-carrier frequency (1.023 MHz). It is very important to understand this possibility, as receivers generally do not sample at a rate equal to a harmonic of the transmitted signal. (You will see this in Homework 2, in which the provided data is sampled at approximately 5.7 MHz).

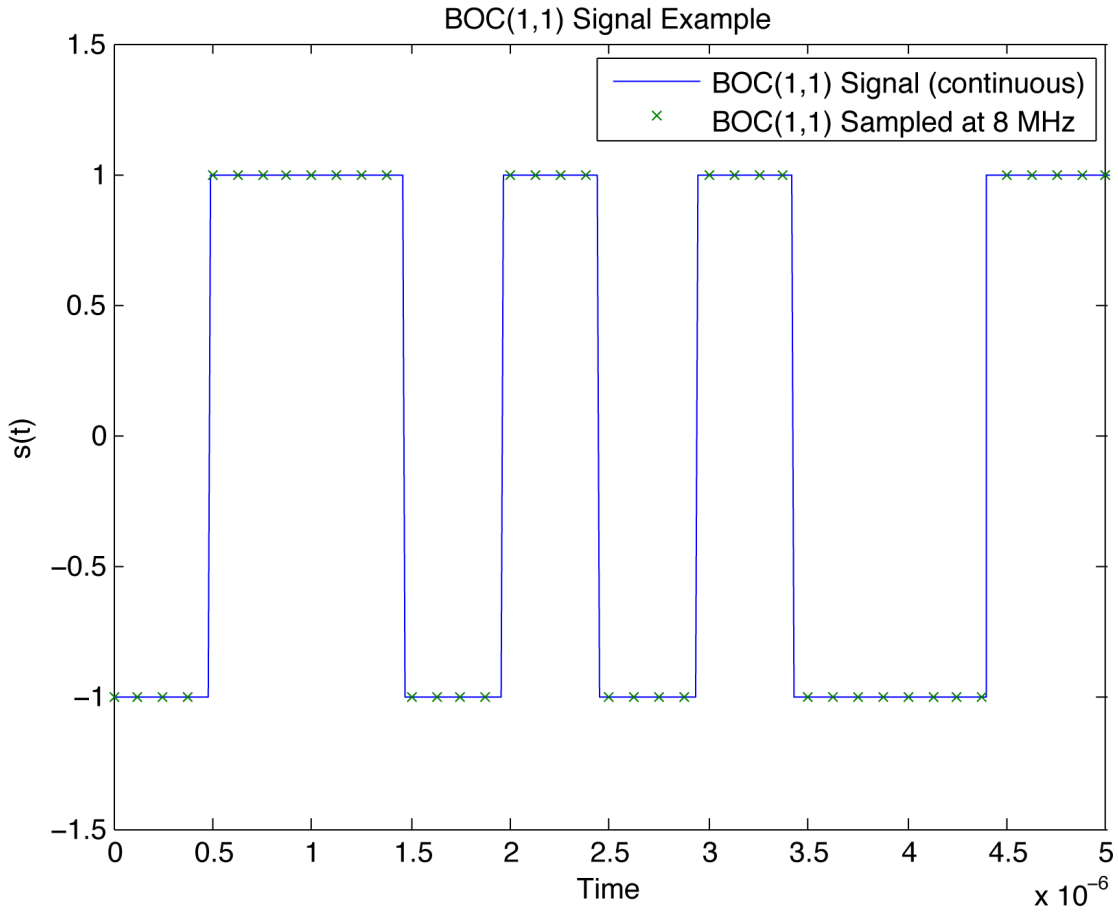

Figure 1: First 5 μs of data. Observe that samples (X) are not synchronous with the pulses in the continuous-time (solid-line) signal.

The inputs to `circcorr.m` are the two arrays that are to be correlated, and the sample period. The sample period is the inverse of the sample rate. Applying to the example data with the matlab command:

```
[R, lag] = circcorr(x,x, 1/8e6);
```

will produce the correlation between x and x (autocorrelation) at 7821 delays (the same number of input points), in the array R. The corresponding 7821 delays are stored in the array "lag", which will have the same dimensions as the value given for the sample period (seconds in this example). Plotting R vs. lag (and scaling the x-axis to microseconds for clarity) produces figure 2. The analytical model for a BOC(1,1) signal, assuming infinitely-long random modulation, is also shown for comparison. The spacing between samples (X) is equal to the sample period of the original data $(1/8E6)$ sec **=0.125 microsec.**

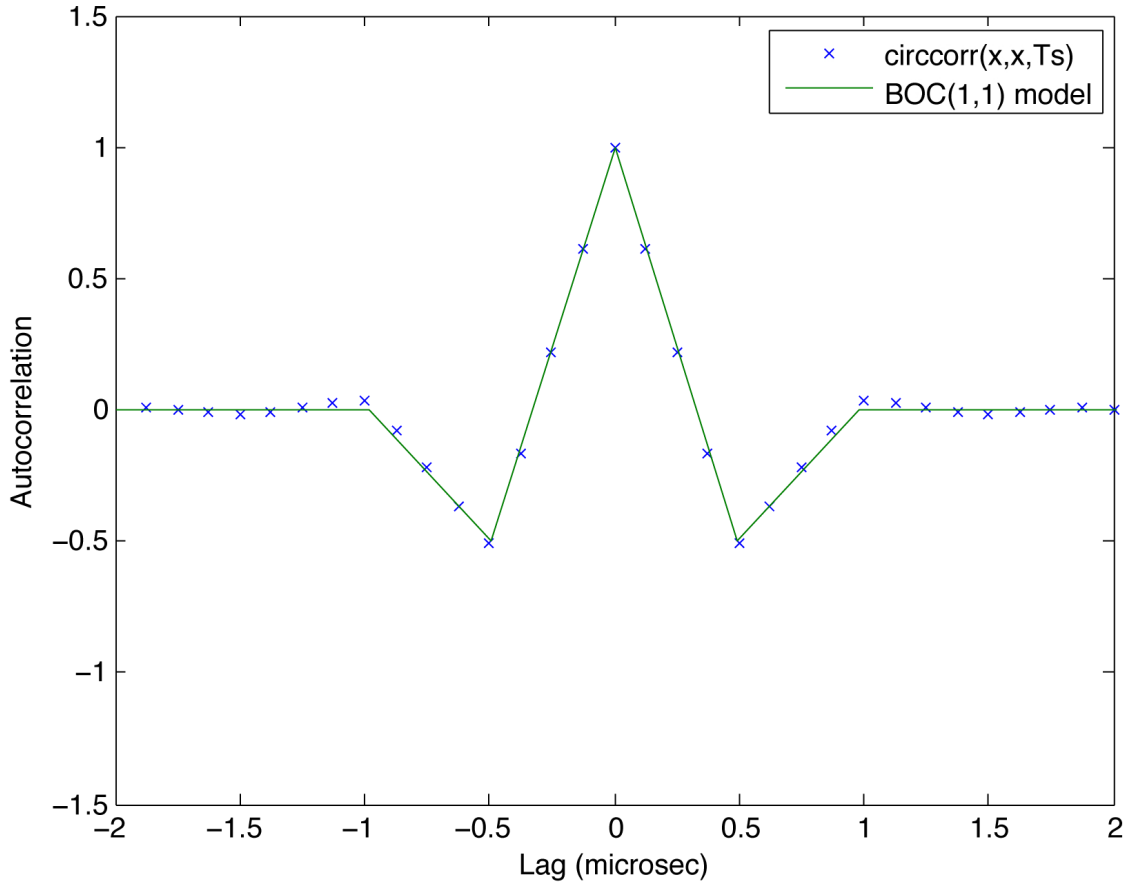

Figure 2: Autocorrelation of the signal x. Numerical values generated from circcorr.m are shown as ('X'), the continuous-time model (assuming an infinitely-long random modulation) is plotted as the solid line.

Background on circcorr.m: How does it work ?

I may have mentioned in class, that computing the correlation in () by direct numerical integration of the product $x(t)y(t+\gamma)$ each time, for each value of γ would not be very efficient. In the past, before microprocessors were as powerful as they are now, receivers did use custom-fabricated chips to perform this integration in hardware for a limited number of delays (typically between 10's and 100's) in parallel. The method used in `circcorr.m` makes use of the Fast Fourier Transform (FFT) to compute this integral **for all delays** , at once, using far fewer operations than would be required in a numerical integration of (). This approach is used in some modern receivers, in particular, those employing the so-called "software defined radio" (SDR) idea.

To derive this method, we first approximate the integral in () as a summation of N evenly spaced samples of the two signals $x(t)$ and $y(t)$, collected at a sample rate of $f_s = 1/T_s$, thus $T_I = NT_s$.

$$x[k] = x(kT_s) \tag{2}$$
$$y[k] = y(kT_s) \tag{3}$$

Further, we will restrict our interest to only delays separated by the same sample period, $\gamma = mT_s$

$$R[m] = R(mT_s) \tag{4}$$

Therefore, equation () becomes the discrete-time summation

$$R[m] = \frac{1}{T_I}\sum_{k}^{N} x[k]y[k+m]T_s = \frac{1}{N}\sum_{k}^{N} x[k]y[k+m] \tag{5}$$

$x[k]$ and $y[k]$ can be written as the inverse discrete fourier transforms (IDFT) of $X[p]$ and $Y[q]$, using the definitions

$$x[k] = \tfrac{1}{N}\textstyle\sum_{p}^{N} X[p]e^{\frac{2\pi pkj}{N}} \tag{6}$$
$$y[k] = \tfrac{1}{N}\textstyle\sum_{q}^{N} Y[q]e^{\frac{2\pi qkj}{N}} \tag{7}$$

(where $j = \sqrt{-1}$ is the imaginary unit).

$$R[m] = \frac{1}{N^3}\sum_{k}^{N}\sum_{p}^{N}\sum_{q}^{N} X[p]Y[q]e^{\frac{2\pi pk}{N}j}e^{\frac{2\pi q(k+m)}{N}j} \tag{8}$$

The order of the sums can be changed, moving the summation over k inside, and grouping together only those exponential terms that involve this index.

$$R[m] = \frac{1}{N^3}\sum_{p}^{N} X[p]\sum_{q}^{N} Y[q]e^{\frac{2\pi qm}{N}j}\sum_{k}^{N} e^{\frac{2\pi(p+q)k}{N}j} \tag{9}$$

The last sum gives the discrete-time delta function, from the following identity

$$\sum_{k}^{N} e^{\frac{2\pi(p+q)k}{N}j} = \delta_{p,-q} = \begin{cases} 1/N & ,p+q=0 \\ 0 & ,p+q\neq 0 \end{cases} \tag{10}$$

Substituting this identity into (9) would eliminate the sum over q, as that would only include one values, that for which $q = -p$. Therefore,

$$R[m] = \frac{1}{N^2}\sum_{p}^{N} X[p]Y[-p]e^{-\frac{2\pi pm}{N}j} \tag{11}$$

which is recognized as the discrete fourier transform of the product $X[p]Y^*[p]$

Before we look inside `circcorr.m`, two more details need to be known. Equations (**??**) and (**??**) define the discrete Fourier transform and inverse discrete Fourier transform, respectively, for any finite number of samples, N. The **fast** Fourier transform (FFT) and its inverse (IFFT) are simply numerically efficient methods of implementing (**??**) and (**??**), optimized when N is a power of 2. The FFT works only with positive integers k, and a finite length of data which numerically approximates the infinite bounds of the Fourier transform

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi ft}dt \tag{12}$$

This definition of frequencies results in a function that is necessarily periodic, which can be visualized as "wrapping around", with its value for negative frequencies found as $X[N] = X[-1]$. This can easily be shown in (). It is the visualization of this wrapping around, which gives this approach the name "circular correlation", and hence the function name. One consequence of this definition, and the use of only