EE 301 Lab 4 –Signal Correlation and Detection

In this lab we will gain experience with the signal correlation and detection in Matlab.

1 What you will learn

In this lab assignment, we will be examining a computational method called correlation to detect the presence of a signal with a known form. This computation can also measure the similarity between two signals. We will see its application when sending several streams of information simultaneously over a single communications channel, as well as sending information (for example a radar signal) in a noisy environment.

2 Background Information and Notes

2.1 Correlation

Let us say that we have two discrete-time signals, x[n] and y[n]. The following formula gives the correlation, C(x,y), between the two signals,

$$C(x,y) = \sum_{n=n_1}^{n_2} x[n]y[n] \tag{2.1}$$

where n_1 and n_2 define the computational interval. This will be referred to as "in-place" correlation. Notice that correlation is achieved by multiplying the two signals together and then summing their product.

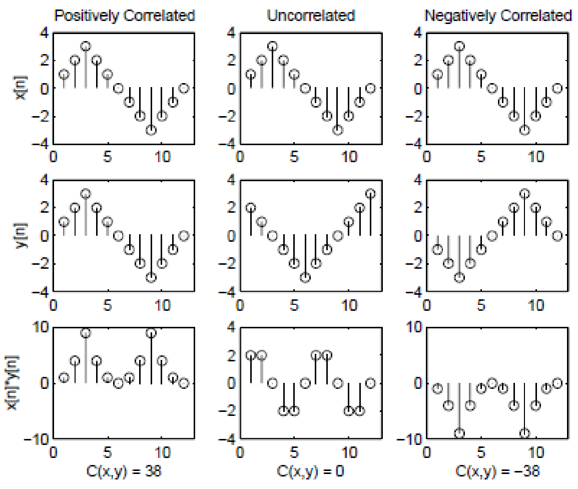

Figure 2.1: Examples of positively correlated, uncorrelated, and anticorrelated signals.

Consider the examples in Figure 2.1. In the left-most column, C(x,y) > 0, which is called positively correlated. This indicates that the signals are more similar than they are dissimilar. In the middle column, C(x,y) is zero, and thus the two signals are said to be uncorrelated. In the right-most column C(x,y) < 0, which means that the two signals are anticorrelated and are mostly dissimilar.

The two signals on the left most-column are identical. This is a special case and looking at equation 2.1, one sees that C(x,x) is simply the **energy** of x[n]. When $C(x,y)=\sqrt{E(x)E(y)}$ (as in the case of the left-most column), the two signals are perfectly correlated and represent the maximum possible correlation value.

We can then define a normalized correlation, C_N(x,y) to be,

$$C_N(x,y)=\frac{C\ (x,y)}{\sqrt{E(x)E(y)}}=\frac{1}{\sqrt{E(x)E(y)}}\sum_{n=n_1}^{n_2}x[n]y[n] \qquad (2.2)$$

The value C_N(x,y) will lie within the range -1 and 1. A normalized correlation value of 1 indicates they are perfectly correlated and a value of -1 indicates they are perfectly anticorrelated. When two signals have a normalized correlation of 1, they are perfectly correlated; if they have a normalized correlation of -1, then they are perfectly anticorrelated.

2.2 Running correlation

We can now modify the correlation equation to calculate the distance to a certain object. If we transmit a radar pulse, x[n], wait for the pulse to reflect off of an object and receive the returned signal, y[n], we can say that y[n] is a delayed version of x[n], or mathematically,

$$y[n] = x[n - n_0] \qquad (2.3)$$

The value of n_0 is unknown but is proportional to the distance of the object. Thus our task is to find the value of n_0.

The process of determining n_0 goes as follows. We first guess that n_0 is zero, calculate the correlation which is just the correlation between x[n] and y[n], and call it r[0]. Let's now say that n_0 is equal to one. We will shift x[n] one sample, and now correlate x[n-1] with y[n]. This correlation value will be stored as r[1]. Continuing on, the general formula of this procedure is then,

$$r[k]=C\big(x[n-k],y[n]\big)=\sum_{n=-\infty}^{\infty}x[n-k]y[n] \qquad (2.4)$$

Once r[k] equals the value of the correlated signals x[n] and y[n], we know the value of n_0. This computation is known as running correlation.

An algorithm for running correlation

Here, we provide an algorithm for running correlation. In this algorithm we refer to the signal we are looking for (i.e. the transmitted radar signal) as x[n], following (2.4). The algorithm proceeds as:

1. Initialize an input buffer to all zeros. The input buffer is an array with a length that is equal to the duration of x[n].
2. For each sample that comes in:

(a) Update the buffer with the method below:
   i. Discard the sample at the beginning of the buffer.
   ii. Shift the rest of the samples one place towards the beginning of the buffer.
   iii. Insert the incoming sample at the end of the buffer.
(b) Initialize a running sum variable to zero.
(c) For each position, n, in the buffer:
   i. Multiply the n^{th} position in the input buffer by the n^{th} sample of x[n].
   ii. Add the resulting product to the running sum.
(d) Output the running sum as the next sample of the correlation signal.

This algorithm will be implemented in the guided exercises of this lab assignment. It is good to note that (a) can be accomplished using a single line of code. Similarly parts (b) through (d) can be implemented in a single line of code using one of Matlab's built in functions and its vector arithmetic capabilities.

2.3 Using running correlation for signal detection

The running correlation example for the radar pulse is an idealized system. In reality, the pulse will incur multiple reflections, become distorted or pass through a noisy system. Thus, it is not possible to find an exact numerical match to the correlated signals. The remedy to this problem is to define a threshold, $c_{T,}$ that will indicate if a signal is present or not. For the radar example, let us define the threshold to be $c_T = E(x)/2$.

2.4 Using correlation for simultaneous communications

We may also use the correlation detector to transmit multiple (binary) signals on a single channel. Each sender will have their own specific code signal that the receiver will use to decipher their message. The code signals may look like those shown in Figure 2.3.

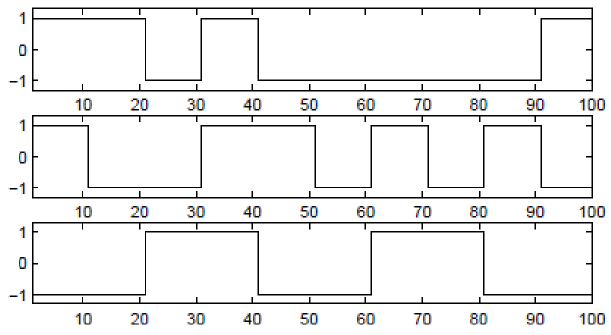

Figure 2.2: Example of code signals for simultaneous communications

In Figure 2.2 the code signals are made up of regions of constant 1 or -1, specifically each consists of ten chips. To send a binary "one," the sender sends their code signal. To send a binary "zero," the sender sends a negated version of their code signal. The receiver then correlates the message (in-place) with the sender's code signal. When the correlation is greater than zero, the receiver records a "one" and if the correlation is less than zero, the receiver records a "zero".

The system is designed in such a way that the correlation between two different code signals (or between a code signal and noise) is reasonably small. Notice however that longer code signals have greater energy and are more easily distinguished from other code signals or from noise.

The description above involved using in-place correlation, however running correlation may also be used (and will be implemented in this guided exercises), as long as we sample the resulting correlation signal at the appropriate times. For example if the code signals are N samples long, we want to pick off the $(k \times N)^{th}$ sample to decode the k^{th} transmitted bit.

2.5 Noise, detector errors, and setting the threshold

Due to a noisy environment, the radar detector we designed has two types of error: a false alarm and a miss. A false alarm occurs when we detect a reflection when no actual reflection exists. This is found when the correlation of the noise with the transmitted signal exceeds the threshold. A miss occurs when the detector fails to detect a real reflection because the noise causes the correlation to drop below the threshold. The trade off is that raising the threshold decreases the likelihood of a false alarm, while lowering the threshold decreases the likelihood of a miss.

We can then calculate the probability of a false alarm and a miss by knowing the noise signal, n[k]. This is simply done by recording a signal when no radar pulse is transmitted. We will then calculate the running correlation between n[k] and the radar pulse we intend to transmit, x[k]. If we call the resulting correlation signal $n_c[k]$, all that is needed is a test whether a sample has exceeded the threshold. It turns out that a threshold of E(x)/2 yields the same number of false alarms as misses, if the distribution of $n_c[k]$ is symmetric. We will also see that a histogram of the values in $n_c[k]$, can be used to determine the error rates.

2.6 Some useful Matlab commands

- To calculate the in-place correlation of signals x and y you can use:

  ```
  >> c_xy = sum(x.*y);
  ```

  Note that x and y need to be the same size.

- To shift values in a buffer (which is just a vector) and append new values, discard the number at the beginning (or end) of the buffer and append the new value. If the buffer is named b, we can do this by either the command b = b(2:end) or b = (1:end-1). We can then append the new number by using an array concatenation operation. For example, if b is a row vector and we'd like to append a new value at the beginning and drop the last value in the buffer, we'd use the code line

```
>> b = [new_sample, b(1:end-1)];
```

- To count how many elements in a vector meet a certain condition we can use the count command in Matlab. The following code will count the number of elements in v that equal 3,

```
>> count = sum( v ==3 );
```

3 Guided Exercises

1. Download the file code_signal.m and use it to create the following signals:

```
>> code 1 = code_signal(75,10);
>> code 1 = code_signal(50,10);
>> code 1 = code_signal(204,10);
```

   (a) Use *subplot* and *stairs* to plot the three code signals on three separate axes in the same figure. After plotting each signal, call axis([1, 100, -1.5, 1.5]) to make sure that the signal is visible.
   - Include your figure, with axis labels on each subplot, a figure number and caption, and the generating code in your report.

   (b) For each of the three signals generated above, calculate:
   - Their mean values
   - Their energies

   (c) Calculate the "in-place" correlation for the following pairs of signals.
   - Code 1 and code 2
   - Code 1 and code 3
   - Code 2 and code 3

   (d) Which of the above pairs are:
   - Positively correlated?
   - Uncorrelated?
   - Anticorrelated?

2. Download the file run_corr.m. run_corr.m is a "skeleton" file for an implementation of the "real-time" running correlation algorithm described in Section 2.2. It accepts two input signals, performs running correlation on them, and produces the correlation signal with a length equal to the sum of the lengths of the input signal minus one.

(a) Complete the function, following the algorithm given in Section 2.2. You can use the completed demo version of the function, run_corr_demo1.p to check your function's output.
   - Include your Matlab code in your report.

(b) Use run_corr.m to compute the running correlation between the following pairs of signals, and plot the resulting correlation signals on the same figure using subplot.
   - Code1 and code 2
   - Code3 and itself

(c) When performing running correlation with a signal and itself, the resulting correlation signal has some special properties. Look at the correlation signal that you computed between code3 and itself.
   - Is the correlation signal symmetric? It can be shown that it should be.
   - What is the maximum value of the correlation signal? How does this relate to the energy of the code3?

3. Download the file lab4_data.mat and load it into your workspace. The file contains the variable dsss, which we will use in this problem. dsss is a signal that is the sum of four sequences of different code signals corresponding to bit sequences from four different users. One of the code signals is a ten chip signal corresponding to the integer 170, while another is a six chip signal corresponding to the integer 25. The other two code signals are unknown to us. In this problem, we will try to extract the bit sequences for the known code signals from dsss. Start by generating the following code signals:

```
>> cs1 = code_signal(170,10);
>> cs2 = code_signal(25,6);
```

(a) Use *subplot* and *stairs* to plot dsss, cs2 and cs2 on three separate axes of the same figure.

(b) Start by using run_corr to correlate the received signal dsss with the longer code signal cs1. Call the resulting signal cor1. Now to decode the sequence of message bits from this user, we need to extract the appropriate samples from cor1. That is, we need to extract just those samples of the running correlation that correspond to the appropriate in-place correlations. We can do this in Matlab using the following command:

```
>>sub_cor1 = cor1(length(cs1):length(cs1):length(cor1));
```

Each sample of sub_cor1 is used to make the decision about one of the user's bits. When it is greater than zero, i.e. the correlation of the received signal with the code signal is positive, the decoder decides the bit is 1. When it is less than zero, the decoder decides the user's bit is 0.
   - On two subplots of the same figure, use *plot* to plot cor1, and *stem* to plot sub_cor1.

- Decode the sequence of bits. You can do this visually or with Matlab. Hint: The sequence is 10 bits long, and the first 3 bits are "011"

(c) Repeat the procedure in a and b above, this time using the code signal cs2. Call your correlation signal cor2, and the vector of extracted values sub_cor2.

- On two subplots of the same figure, use plot to plot cor2 and stem to plot the signal sub_cor2.
- Decode the sequence of bits. Hint: there are 17 bits in this sequence.
- Since the code signal cs2 has less energy (because it is shorter), there is a greater chance of error. Are there any decoded bits that you suspect might be incorrect? Which ones? Why?

4. lab4_data.mat contains three other signals: radar_pulse, radar_received, and radar_noise. The received signal contains several reflections of the transmitted radar pulse and noise. The signal radar_noise contains noise with similar characteristics to the noise in the received signal without the reflected pulses.

(a) First let's take a look at the first two signals.

- Calculate the energy of radar_pulse, E(x).
- Use *subplot* to plot radar_pulse and radar_received in separate axes of the same figure.
- Can you identify the reflected pulses in the received signal by visual inspection alone?

(b) Use run_corr to correlate radar_received with radar_pulse.

- Plot the resulting correlation signal.
- Where are the received pulses? Visually identify simple locations of each pulse in the correlation signal.
- Given the speed of light is 3×10^8 m/s and the sampling frequency of the detector is 10^7 samples per second, what is the approximate distance to each object? (Remember that the radar pulse must travel to the object and then back again.)

(c) In a real radar detector, the correlation signal would be compared to a threshold to perform the detection. To estimate the error rates for such a detector, let's consider a threshold that is equal to one-half the energy of the transmitted pulse, i.e. $c = E(x)/2$. Perform running correlation between radar_pulse and radar_noise call the resulting correlation signal noise_c.

- Plot noise_c.
- For how many samples is noise_c greater than this threshold? Use this value to estimate the false alarm rate.
- For how many samples is noise_c less than this threshold minus the energy of the transmitted pulse? Use this value to estimate the miss rate.
- What is the total error rate for this threshold?

(d) We can also use a histogram to judge the number of errors.

- Use *hist* and plot the histogram of noise_c using 100 bins.
- Describe how you could derive the error numbers in problem 4c from the histogram.

(e) Suppose that the detector false alarms are considered to be more serious than detector misses. Thus, we have determined that we want to raise the threshold so that we achieve a false alarm rate of approximately 0.004. Find a threshold that satisfies this requirement.

- What is your threshold?
- What is the false alarm rate on this noise signal with your threshold?
- What is the miss rate on this noise signal with your threshold?
- What is the total error rate for the new threshold? Compare this to the total error rate of the threshold using in problem 4c.

4 Review

1. None.

5 Lab Report

1. The first page of your Lab report should be a cover sheet with your name, USC ID and Lab #. Please note that all reports should be typed.
2. Answer all the questions which were asked in the lab report. Kindly display the code lines you executed to arrive at your answer along with figures to support them. Please give written explanation or put comment lines where necessary. Please note that each figure should have proper labels for the x and y axis and should have a suitable title.
3. Answer the review questions.
4. Submit a printout of your completed M-file documenting all the lab exercises.

This lab document and the figures contained were adapted from a University of Michigan Signals and Systems course lab handout (2002).