EECS 16A Designing Information Devices and Systems I
Fall 2018 Lecture Notes Note 22

22.1 Positioning Sytems: Trilateration and Correlation

In this note, we'll introduce two concepts that are critical in our positioning system, *trilateration* and *correlation*.

To being, let's review the underlying concepts of GPS, introduced in the previous note. Recall that a GPS system consists of two parts: (1) a receiver, with unknown location, and (2) a set of "beacons", that send messages and have known location. In GPS, these are satellites, but they could be anything that can broadcast a message, such as a speaker sending audio signals or a radio transmitter on the ground. The goal is to determine the location of the receiver from the beacon's messages.

We split the process of localizing the receiver into two steps:

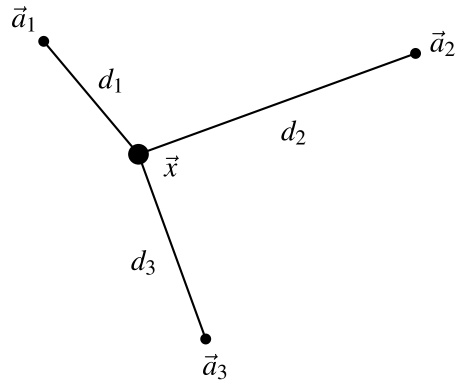
1. Find the distances from the receiver to each beacon, based on the messages received.
2. Combine the distance measurements to determine the receiver's location.

We'll used *correlation* to determine the distances, and we'll use *trilateration* to find the position based on the distance measurements and known beacon locations. Let's start with trilateration.

22.2 Trilateration

In this section, we'll assume that we've already determined the distance from the receiver to each beacon (we'll explain how to measure this distance in the next section).

Let's imagine we have a situation like the one below. We know the locations of the beacons $\vec{a}_1, \vec{a}_2, \vec{a}_3$, but don't know the location of the point at $\vec{x}$ (we'll be trying to find out what $\vec{x}$ is). We do know the distances d_1, d_2, d_3. We're trying to find the coordinates of $\vec{x}$ in this diagram:



We know that:

1. $\|\vec{x}-\vec{a}_1\|^2 = d_1^2$
2. $\|\vec{x}-\vec{a}_2\|^2 = d_2^2$
3. $\|\vec{x}-\vec{a}_3\|^2 = d_3^2$

Rewriting these using transpose notation we get:

$$\vec{x}^T\vec{x} - 2\vec{a}_1^T\vec{x} + \|\vec{a}_1\|^2 = d_1^2 \tag{1}$$
$$\vec{x}^T\vec{x} - 2\vec{a}_2^T\vec{x} + \|\vec{a}_2\|^2 = d_2^2 \tag{2}$$
$$\vec{x}^T\vec{x} - 2\vec{a}_3^T\vec{x} + \|\vec{a}_3\|^2 = d_3^2 \tag{3}$$

But we have squared terms here involving unknowns. That's not really conducive to our style of computation - we prefer only linear terms in unknowns, so we can use linear algebra.

For this, we use a trick. Let's subtract equation 1 from equation 2, and separately again from equation 3. Then we get:

$$2(\vec{a}_1-\vec{a}_2)^T\vec{x} = \|\vec{a}_1\|^2 - \|\vec{a}_2\|^2 - d_1^2 + d_2^2$$

and

$$2(\vec{a}_1-\vec{a}_3)^T\vec{x} = \|\vec{a}_1\|^2 - \|\vec{a}_3\|^2 - d_1^2 + d_3^2$$

We can then stick these into a matrix, which will only have linear terms:

$$\begin{bmatrix} 2(\vec{a}_1-\vec{a}_2)^T \\ 2(\vec{a}_1-\vec{a}_3)^T \end{bmatrix} \vec{x} = \begin{bmatrix} \|\vec{a}_1\|^2 - \|\vec{a}_2\|^2 - d_1^2 + d_2^2 \\ \|\vec{a}_1\|^2 - \|\vec{a}_3\|^2 - d_1^2 + d_3^2 \end{bmatrix}$$

.

This can be solved for a location. Three circles uniquely define a point in 2D. The argument extends in 3D to four spheres. And in 4D to five hyper-spheres. (In the real-world, time is the fourth dimension we need to solve for.)

22.3 Finding distances with correlation

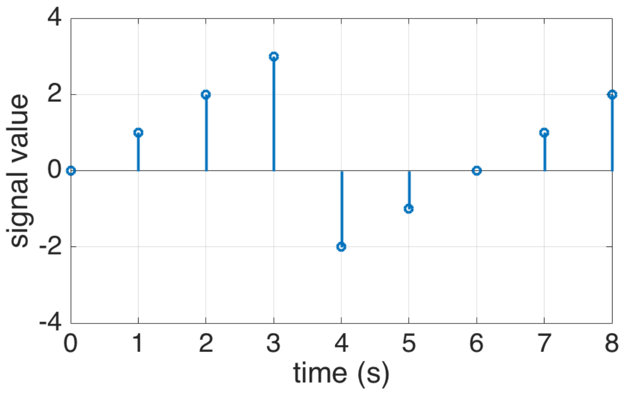
Now, we'll explore how to find the distances between the receiver and the beacons. To do this, we'll need to be a little more explicit about what messages the beacons are sending. We will generally say that the beacons are sending "signals."

22.3.1 Signals

A **signal** is a message that contains information as a function of time[1]. We call a signal a *discrete-time signal* if it is only defined at specific points in time (for example, every minute). In contrast, a *continuous-time signal* is defined over all time. In this class we'll focus on discrete-time signals.

We can represent a discrete-time signal as a list of numbers. It's frequently convenient to write the signal as a vector where each element is the value at a single time point – then we can use all of the concepts of matrix algebra on signals.

Consider the following example: a beacon in our GPS system is sending a message using radio waves. The amplitude of the radio wave as a function of time is a signal, and we only consider discrete times (for example, every second). We can represent the signal graphically or as a vector:



$$\vec{s} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

Every element of the vector represents the signal value at one timestep. We'll use the notation $\vec{s}[k]$ to represent the k-th element of the vector where initial element is at $k = 0$. For example, in the vector $\vec{s}$ above, $\vec{s}[0] = 0$, $\vec{s}[1] = 1$, etc.

22.4 Cross-Correlation

The **cross-correlation** is a measure of the similarity between two signals $\vec{x}$ and $\vec{y}$ based in the inner product. We compute the inner product $\langle \vec{x}, \vec{y} \rangle$, and then the same with $\vec{y}$ shifted by 1, then shifted by 2, and so on. First we'll define two types of cross-correlation, then we'll explore how we can use this concept to find the distance between a beacon and receiver.

22.4.1 Linear Cross-Correlation

A linear cross-correlation is defined as follows:

$$\text{corr}(\vec{x}, \vec{y})[k] = \sum_{i=-\infty}^{\infty} \vec{x}[i]\, \vec{y}[i-k] \tag{4}$$

where $\vec{x}$ and $\vec{y}$ are input signals and $\text{corr}(\vec{x}, \vec{y})[k]$ is the k-th element of their cross-correlation.

[1]Signals can also be functions of other varibles, such as space (like in an image). However, for simplicity we'll define them as a function of time, while knowing that they can be used more generally

Let's break this down: If we would like to find the value of the cross correlation at 0, we plug $k = 0$ into the equation:

$$\text{corr}(\vec{x}, \vec{y})[0] = \sum_{n=-\infty}^{\infty} \vec{x}[n]\vec{y}[n]$$

Recall that the inner product of two vectors $\vec{u}, \vec{v} \in \mathbb{R}^n$ is

$$\langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{n} u_i v_i$$

which is very similar to the cross correlation at 0. However, there is one difference: the summation in the cross correlation equation goes from $-\infty$ to ∞. This means that we will need to know $\vec{x}[n]$ and $\vec{y}[n]$ for all n, even if $\vec{x}, \vec{y}$ were only defined over a finite range. To do this, **we make the assumption that $\vec{x}[n]$ and $\vec{y}[n]$ are 0 for any n that they are not defined for.** (This assumption is what differentiates a linear cross-correlation from a circular cross-correlation, which we will introduce in the next section.)

Consider two vectors, $\vec{s}_1 \in \mathbb{R}^3$ and $\vec{s}_2 \in \mathbb{R}^2$:

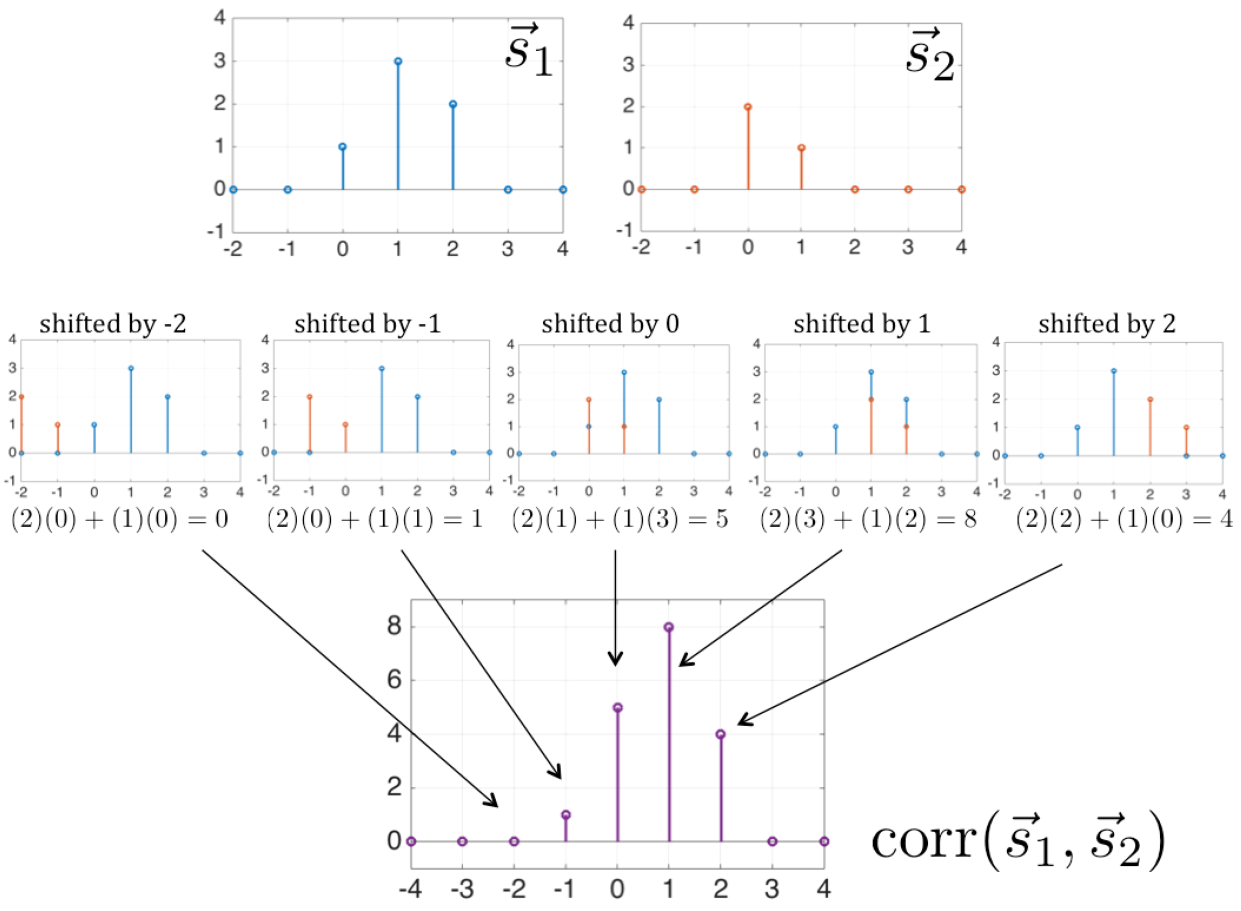
$$\vec{s}_1 = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} \qquad \vec{s}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

We cannot calculate their inner product because they are not the same dimension, but we can calculate $\text{corr}(\vec{s}_1, \vec{s}_2)[0]$ because we assume that the smaller vector has another zero at the end, making them the same size.

$$\text{corr}(\vec{s}_1, \vec{s}_2)[0] = (1)(2) + (3)(1) + (2)(0) = 5$$

Now that we can find the cross correlation at 0, how do we find it at other k values? Looking back at the definition in Eq. (4), we see that $\vec{y}[i-k]$ has all of the same values as $\vec{y}[i]$, but shifted to the right (or down) by k. Therefore, to find $\text{corr}(\vec{s}_1, \vec{s}_2)[k]$, we shift the second signal, $\vec{y}$ to the right by k units, then take the inner product between $\vec{x}$ and the shifted version of $\vec{y}$. As before, we assume that any values not defined in either vector are 0.

The figure below summarizes the linear cross correlation for the vectors $\vec{s}_1$ and $\vec{s}_2$ defined above. Note that both signals have been padded with zeros on either side. Each value of the cross-correlation is found by shifting $\vec{s}_2$ (right for positive values, left for negative values) and then taking the inner product.



Eventually, the two signals no longer overlap (for example, when $\vec{s}_2$ is shifted by -2 units). When the signals don't overlap, the inner product will be 0, since every non-zero element will be multiplied by zero. At this point we don't need to calculate more of the cross correlation. If we write all of the non-zero values in a vector, we get the following in this example:

$$\text{corr}(\vec{s}_1,\vec{s}_2) = \begin{bmatrix} 1 \\ 5 \\ 8 \\ 4 \end{bmatrix}.$$

This is the same as what you get in python with `numpy.correlate([1, 3, 2], [2, 1], 'full')`. The 'full' tag tells python to return all of the non-zero values. However, python doesn't tell you which output corresponds to 0 shift of the signal – you must calculate that yourself based on the length of the input signals.

22.4.2 Circular Cross-Correlation

A circular cross-correlation with period N is defined as follows:

$$\text{circcorr}(\vec{x},\vec{y})[k] = \sum_{i=0}^{N-1} \vec{x}[i]\,\vec{y}[(i-k)_N] \tag{5}$$

There's some new notation here: $(i)_N$ is another way of writing "i mod N." $(i)_N$ is the remainder when i is divided by N. For example, if $N = 3$,

$$\begin{array}{lll} (0)_N = 0 \bmod 3 = 0 & (1)_N = 1 \bmod 3 = 1 & (2)_N = 2 \bmod 3 = 2 \\ (3)_N = 3 \bmod 3 = 0 & (4)_N = 4 \bmod 3 = 1 & (5)_N = 5 \bmod 3 = 2 \end{array}$$

This notation allows us to count in a circular way: we count from 0 to $N-1$ and then repeat, starting at 0.

Now let's go back to our definition of circular cross-correlation. Once again, we'll start by considering when $k = 0$:

$$\text{circcorr}(\vec{x}, \vec{y})[0] = \sum_{i=0}^{N-1} \vec{x}[i]\, \vec{y}[(i)_N]$$

We are able to remove the N subscript in this case because i is always less than N due to the bounds on the summation:

$$\text{circcorr}(\vec{x}, \vec{y})[0] = \sum_{i=0}^{N-1} \vec{x}[i]\, \vec{y}[i]$$

Now this is exactly the inner product of the $\vec{x}$ and $\vec{y}$ if they are both length N. If either vector is longer than N, then it is the inner product of the first N elements. However, this case is less useful since some of the elements are never used in the computation. Therefore, for circular correlations we restrict ourselves to the case where $\vec{x}$ and $\vec{y}$ are both length N.

How about when k is not 0? Now, we take the inner product of $\vec{x}$ and a version of $\vec{y}$ that is circularly shifted by k. We'll denote this circularly shifted vector as $\vec{y}^{(k)}$. For example:

$$\vec{y} = \vec{y}^{(0)} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \qquad \vec{y}^{(1)} = \begin{bmatrix} 4 \\ 1 \\ 2 \\ 3 \end{bmatrix} \qquad \vec{y}^{(2)} = \begin{bmatrix} 3 \\ 4 \\ 1 \\ 2 \end{bmatrix}$$

Using this notation

$$\text{circcorr}(\vec{x}, \vec{y})[k] = \left\langle \vec{x}, \vec{y}^{(k)} \right\rangle = \vec{y}^{(k)^T} \vec{x}$$

We can write all of the values of the correlation in vector format:

$$\text{circcorr}(\vec{x}, \vec{y}) = \begin{bmatrix} \langle \vec{x}, \vec{y}^{(0)} \rangle \\ \langle \vec{x}, \vec{y}^{(1)} \rangle \\ \vdots \\ \langle \vec{x}, \vec{y}^{(N-1)} \rangle \end{bmatrix}$$

Then we expand this into matrix-vector multiplication

$$
\begin{aligned}
\text{circcorr}(\vec{x},\vec{y}) &= \begin{bmatrix} - & \vec{y}^{(0)^T} & - \\ - & \vec{y}^{(2)^T} & - \\ & \vdots & \\ - & \vec{y}^{(N-1)^T} & - \end{bmatrix} \begin{bmatrix} | \\ \vec{x} \\ | \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \vec{y}[0] & \vec{y}[1] & \dots & \vec{y}[N-2] & \vec{y}[N-1] \\ \vec{y}[N-1] & \vec{y}[0] & \dots & \vec{y}[N-3] & \vec{y}[N-2] \\ \vdots & & & & \vdots \\ \vec{y}[2] & \vec{y}[3] & \dots & \vec{y}[0] & \vec{y}[1] \\ \vec{y}[1] & \vec{y}[2] & \dots & \vec{y}[N-1] & \vec{y}[0] \end{bmatrix} \begin{bmatrix} | \\ \vec{x} \\ | \end{bmatrix}}_{C_{\vec{y}}} \\
&= C_{\vec{y}}\vec{x}
\end{aligned}
$$

Here, we've $C_{\vec{y}}$ is a matrix containing all of the shifted versions of $\vec{y}$ in its rows. This is called a **circulant matrix**, and is another way to write a circular correlation.

Circular correlations are the same as linear correlations if we assume that the second input, $\vec{y}$, is repeated indefinitely. (If we circularly shift $\vec{y}$, the end of the vector appears at the beginning. This is exactly what happens if there is another copy of $\vec{y}$ before the initial copy.) As a result, circular correlations are very useful when we have periodic (i.e. repeating) signals, which is what we will use for positioning.

22.4.3 Correlation Interpretation

Now that we've introduced cross-correlation, what does it mean? We can think of the cross-correlation as a sliding inner product. Recall that the inner product can be written as

$$\langle\vec{x},\vec{y}\rangle = \|\vec{x}\|\,\|\vec{y}\|\cos\theta$$

where θ is the angle between the vectors. If θ is low, it indicates that the vectors are pointed in almost the same direction, meaning the vectors are very similar. As θ approaches 90°, the vector point increasingly different directions, indicating that the vector are not similar.

Therefore, when the inner product is large, $\vec{x}$ and $\vec{y}$ are more similar, and when it's small, they are more different. The cross-correlation checks the inner product at all relative shifts between the signals, so it tells us how similar two signals are at every shift.

22.4.4 Correlation Properties

It is important to note that correlation is not commutative:

$$\text{corr}(\vec{x},\vec{y}) \neq \text{corr}(\vec{y},\vec{x})$$

We can think of circular correlation as a special case of linear correlation, so it is not commutative either:

$$\text{circcorr}(\vec{x},\vec{y}) \neq \text{circcorr}(\vec{y},\vec{x})$$

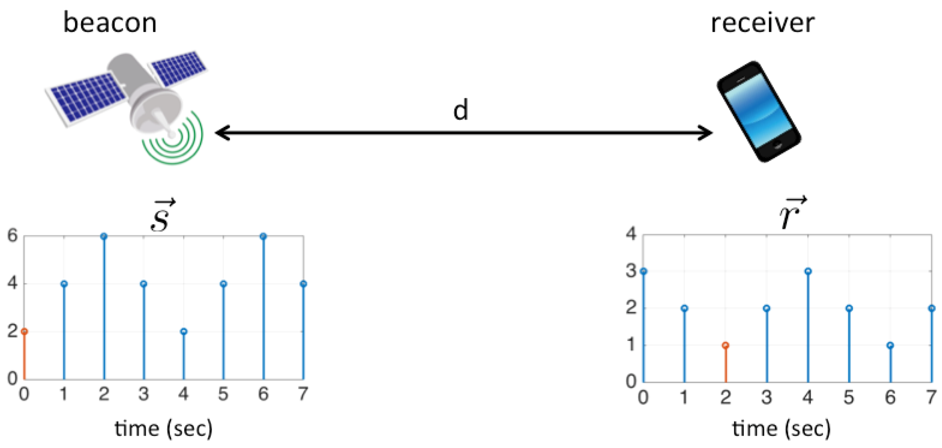
In future classes, you'll learn about the *convolution* which is an operation similar to correlation but commutative. (A mathematical structure in machine learning called a *convolutional neural network* takes its name from this operation.)

We can take a correlation between a signal and itself. This is call an **autocorrelation**. The autocorrelation tells us how similar a signal is to all shifts of itself.

22.5 Modeling the positioning problem

Now that we've defined cross-correlation, we are ready to use it to solve our positioning problem. Recall that we have several beacons that are sending signals that our receiver picks up. We want to determine the distance from the receiver to each beacon. Let's start with the case where there is a single beacon a distance d away from the receiver.

The beacon sends a periodic signal using radio waves (or something else that travels through air, such as light or sound). Since radio waves travel at a finite velocity, there is a delay between the when the beacon sends the signal and when the receiver sees it. For example, in the diagram below, the value marked in red is sent at $t = 0$ and received at $t = 2$, indicating that there is a 2 second delay (assuming there is a sample ever second).



Our goal is to determine this time delay, τ. Then we can use the velocity of radio waves (or light, or sound), v, to determine the distance: $d = v\tau$.

We'll assume that the beacon is sending a known periodic signal, $\vec{s}$, with period N. The receiver will measure the first N samples. We can model the receiver signal, $\vec{r}$ as

$$\vec{r} = \alpha \vec{s}[k - \tau] \tag{6}$$

where α is a scalar that accounts for beacon's signal potentially becoming weaker as it travels. τ is the time delay that we would like to measure.

We have N equations (from each of our N measurements) and two unknowns α and τ. However, τ is related to $\vec{r}$ and $\vec{s}$ in a nonlinear way, so we cannot use our standard linear algebra to solve this problem. What can we do?

Recall that the cross-correlation tells us how much of a signal is present at every shift. Therefore, we can estimate the true shift by finding the index of the cross-correlation with the largest value!

Let's walk through this with the example above. We know that the beacon is sending the following signal, $\vec{s}$ of length $N = 4$ that repeats periodically. We measure the first N samples at the receiver to get $\vec{r}$.

$$\vec{s} = \begin{bmatrix} 2 \\ 4 \\ 6 \\ 4 \end{bmatrix} \qquad \vec{r} = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

We'll try taking the inner product of our received signal with every possible circular shift of our known beacon signal.

$$\begin{aligned}
\text{circcorr}(\vec{r},\vec{s}) &= \begin{bmatrix} \left\langle \vec{r}, \vec{s}^{(0)} \right\rangle \\ \left\langle \vec{r}, \vec{s}^{(1)} \right\rangle \\ \left\langle \vec{r}, \vec{s}^{(2)} \right\rangle \\ \left\langle \vec{r}, \vec{s}^{(3)} \right\rangle \end{bmatrix} \\
&= \begin{bmatrix} - & \vec{s}^{(0)^T} & - \\ - & \vec{s}^{(1)^T} & - \\ - & \vec{s}^{(2)^T} & - \\ - & \vec{s}^{(3)^T} & - \end{bmatrix} \begin{bmatrix} | \\ \vec{r} \\ | \end{bmatrix} \\
&= \begin{bmatrix} 2 & 4 & 6 & 4 \\ 4 & 2 & 4 & 6 \\ 6 & 4 & 2 & 4 \\ 4 & 6 & 4 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \\ 2 \end{bmatrix} \\
&= \begin{bmatrix} 28 \\ 32 \\ 36 \\ 32 \end{bmatrix}
\end{aligned}$$

The maximum is 36 at $k = 2$ which indicates that there is a two sample delay between $\vec{s}$ and $\vec{r}$ (remember, we the first value is at index 0 because there is no shift). Looking back at the figure above, we can see that this is true shift.

We can write this process for estimating τ using optimization notation:

$$\hat{\tau} = \arg\max_k (\text{circcorr}(\vec{r},\vec{s})[k])$$

This means that our estimate of τ (denoted $\hat{\tau}$) is the value of k that makes $\text{circcorr}(\vec{r},\vec{s})[k]$ largest. This style of notation is very common in optimization and machine learning.