CSM 16A Module 3 Refresher

Cooper Bedin, Viren Khandal

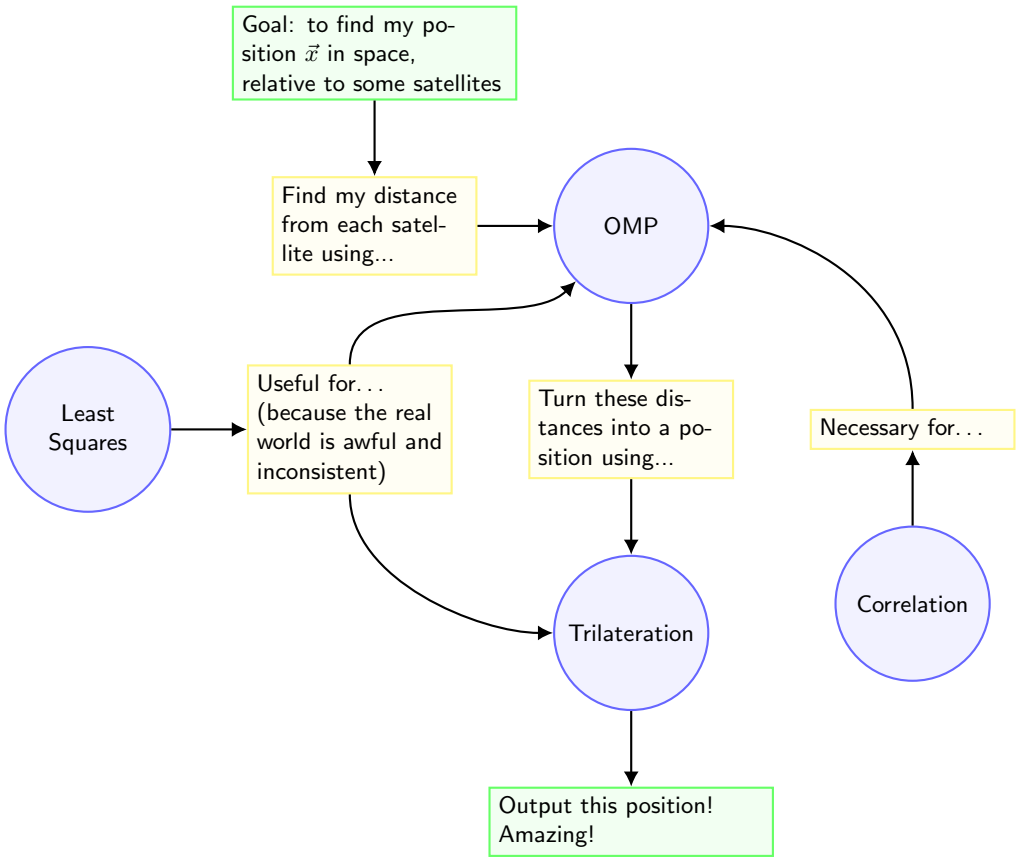
Last updated: April 18, 2021

Hi everyone! I know times are kind of crazy right now, so I hope this document can help a bit! I threw this together last semester to give a kind of refresher on the Module 3 topics for 16A. I try to focus on common misconceptions and the architecture that ties all these topics together, so hopefully it also provides some guidance that you may not find in the official course notes. If you have any questions or comments, you can always reach out to me at `cooper.bedin@berkeley.edu`.

<3 Cooper Bedin (16A SM)

1 First, a quick little roadmap

I'm a huge believer in emphasizing the motivation for topics when teaching them. I sometimes feel that the motivation behind the Module 3 topics, especially in how they all come together, can be...easily missed if you're not paying attention, so, to start off here's a quick survey of how everything goes down in Module 3:

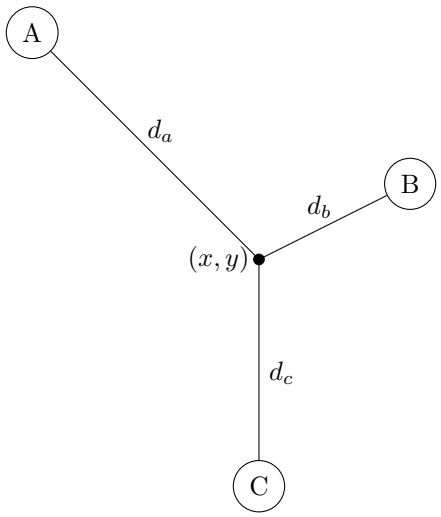


So all of the major topics do fit together! I promise! Each of the blue nodes on the above graph corresponds to a major area that'll be taught, so in this review document I'll be surveying them each individually. One

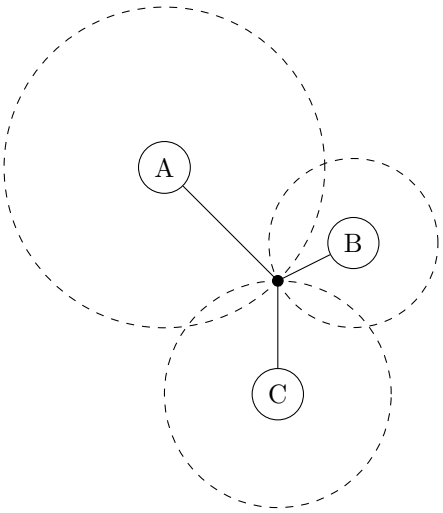
thing you're probably noticing is that the order in which the topics are taught in 16A *does not* correspond to the flow between nodes in this graph. Presumably there is some cosmic reason for this, but as far as you're concerned it means that when students are doing correlation and wondering why the hell they should care, you can point to this map and promise them that things are, in fact, going somewhere.

2 Trilateration

The first part of the module is trilateration! The way the trilateration talk usually goes is: imagine that we have three satellites: A, B, and C, and we are trying to use information from these satellites to find the location of some point (x, y) in the 2D-plane.



In the setup of this problem, we are assuming that we already know the positions of the satellites, and their distances d_a, d_b, d_c to (x, y) (we will have to figure out how to find these distances, but that comes later). It turns out that this information alone is enough to uniquely determine (x, y)—for example, we could plot three circles centered around A, B, and C of radii d_a, d_b, d_c respectively and find the intersection (note that we need at least three circles for the intersection to be unique in the 2D plane):



Cool! But how do we solve a system based on circles? You can probably guess: we're going to use linear algebra! Formally, we represent the position of each satellite with a vector, $\vec{a}, \vec{b}, \vec{c}$, and our unknown point

(x, y) also as a vector $\vec{x} = [x\ y]^T$. Then, we can make use of the euclidean norm of a vector, which is defined as $\|\vec{v}\| = \sqrt{\langle \vec{v}, \vec{v} \rangle}$, i.e., the root of the inner product of $\vec{v}$ with itself. This is cool because the norm of a vector corresponds to its magnitude, which in two and three-dimensions corresponds to length! When we think about it, we already know the lengths of the vectors $\vec{a} - \vec{x}$, $\vec{b} - \vec{x}$, and $\vec{c} - \vec{x}$, because these are just d_a, d_b, and d_c, so, we can write out a system of vector equations:

$$\|\vec{a} - \vec{x}\|^2 = d_a^2 \quad \rightarrow \quad \vec{x}^T\vec{x} - 2\vec{a}^T\vec{x} + \|\vec{a}\|^2 = d_a^2$$

$$\|\vec{b} - \vec{x}\|^2 = d_b^2 \quad \rightarrow \quad \vec{x}^T\vec{x} - 2\vec{b}^T\vec{x} + \|\vec{b}\|^2 = d_b^2$$

$$\|\vec{c} - \vec{x}\|^2 = d_c^2 \quad \rightarrow \quad \vec{x}^T\vec{x} - 2\vec{c}^T\vec{x} + \|\vec{c}\|^2 = d_c^2$$

This is all well and good, but you probably remember that the $\vec{x}^T\vec{x}$ terms make this system non-linear in terms of x and y, and **we. like. linear.** You also may remember though that there's a pretty easy fix for this—just subtract the third equation from the first two!

$$2(\vec{a} - \vec{c})^T\vec{x} + = d_a^2 - d_c^2 - \|\vec{a}\|^2 + \|\vec{c}\|^2$$

$$2(\vec{b} - \vec{c})^T\vec{x} + = d_b^2 - d_c^2 - \|\vec{b}\|^2 + \|\vec{c}\|^2$$

Now we have a system of two linear equations to solve for our two unknowns x, y. You (and your students) may feel more familiar with the distance formula $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$, which can be a lot more intuitive than vectors as we're still getting comfy with linear algebra. The distance formula is also a totally valid and correct way to understand trilateration, and will get you to the same answer because underlyingly it's the same process, so go for it! I chose to notate everything above in vector format because this is how trilateration is formally represented by the course notes, and because we can generalize the above method to an arbitrary number of equations/dimensions with very little extra work.

3 Correlation

Personally, I find manual correlation to be one of the most wacky, unintuitive, and easy-to-mess-up things you learn in 16A, but luckily it is also one of the least important things to remember how to do by hand, so long as we understand the intuition behind it. Just for fun, here's the formula for cross-correlation:

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

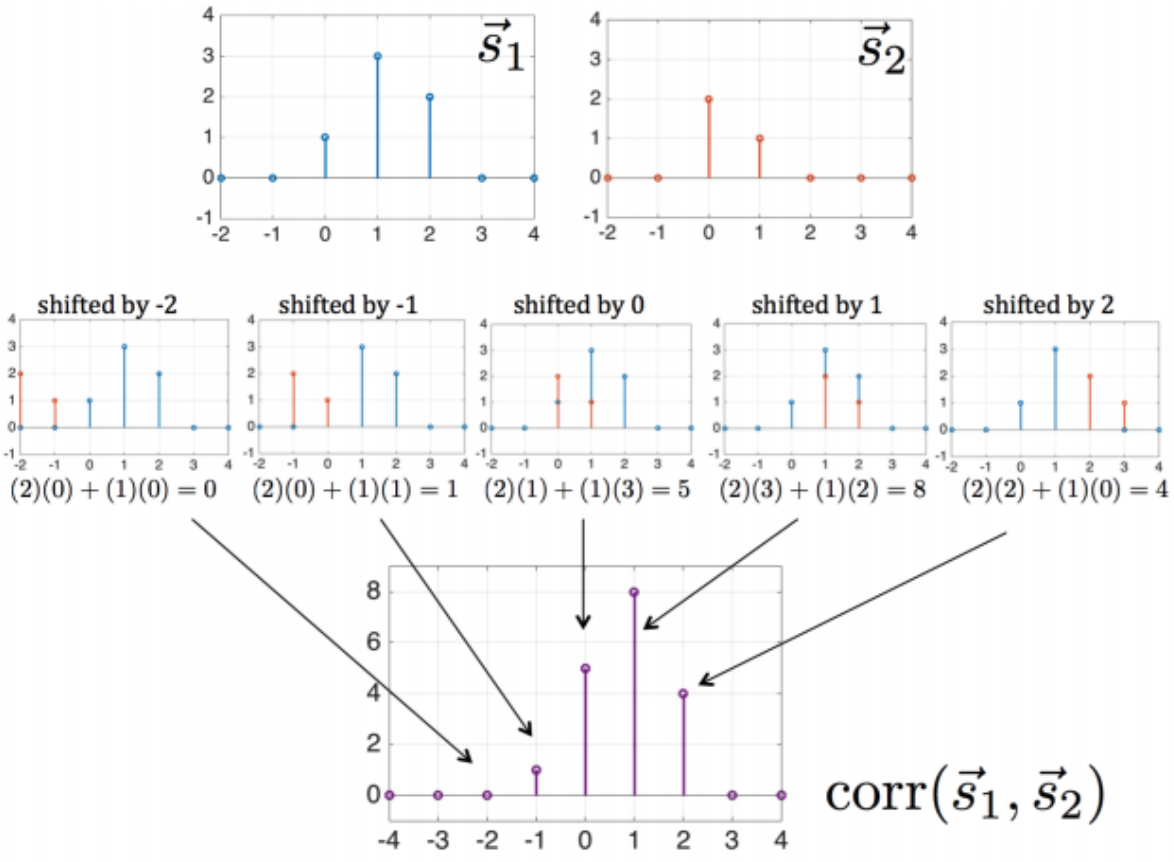
...Not my favorite formula either, but what happens when we do correlation is actually pretty neat! If we dissect this formula, we might notice that the kth term of $\text{corr}_{\vec{x}}(\vec{y})$ is the inner product of $\vec{x}$ with $\vec{y}$ shifted down/to the right (however you want to think about it) by k terms *relative to* $\vec{x}$. This is cool, because it gives us a lens into why correlation is important—the inner product of two vectors will be greater the more similar those two vectors are, so the largest entry in $\text{corr}_{\vec{x}}(\vec{y})$ also corresponds to the k-shift of $\vec{y}$ that makes it most similar to $\vec{x}$.

Just as a quick example, which I always find helpful when teaching correlation, let's say that $\vec{s}_1 = [1\ 3\ 2]$ and $\vec{s}_2 = [2\ 1]$, and we're interested in finding $\text{corr}_{\vec{s}_1}(\vec{s}_2)$. Visually, these would be the dot products that we're executing (note the zero-padding):

$$\left|\begin{array}{ccccc} [2 & 1] & 0 & 0 & 0 \\ 0 & 0 & [1 & 3 & 2] \end{array}\right| \left|\begin{array}{cccc} [2 & 1] & 0 & 0 \\ 0 & [1 & 3 & 2] \end{array}\right| \left|\begin{array}{ccc} [2 & 1] & 0 \\ [1 & 3 & 2] \end{array}\right| \left|\begin{array}{ccc} 0 & [2 & 1] \\ [1 & 3 & 2] \end{array}\right| \left|\begin{array}{ccc} 0 & 0 & [2 & 1] \\ [1 & 3 & 2] & 0 \end{array}\right|$$

$$= 0 \qquad = 1 \qquad = 5 \qquad = 8 \qquad = 4$$

$$\text{corr}_{\vec{s}_1}(\vec{s}_2) = [0\ 1\ 5\ 8\ 4]$$



Now, you may also remember our good friend circular correlation, who is defined as such:

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

The same idea as linear correlation, but instead of letting the entries of $\vec{y}$ trail off into zero-padding land, they swing back around to the first position. Circular correlation is particularly useful for periodic signals of length N—if we were to take the linear correlation of two infinitely-long periodic signals, each entry could very well end up trailing off to $\pm\infty$, as the domain of our sum is infinite. However, if we take the circular correlation instead, then we can confirm the relative shift of $\vec{y}$ against $\vec{x}$ that makes the two most similar.

4 Least squares

Least squares is very well-trod ground within and beyond 16A, and you probably remember her well. Below is a version of the 16A derivation of least squares, for your referencing pleasure:

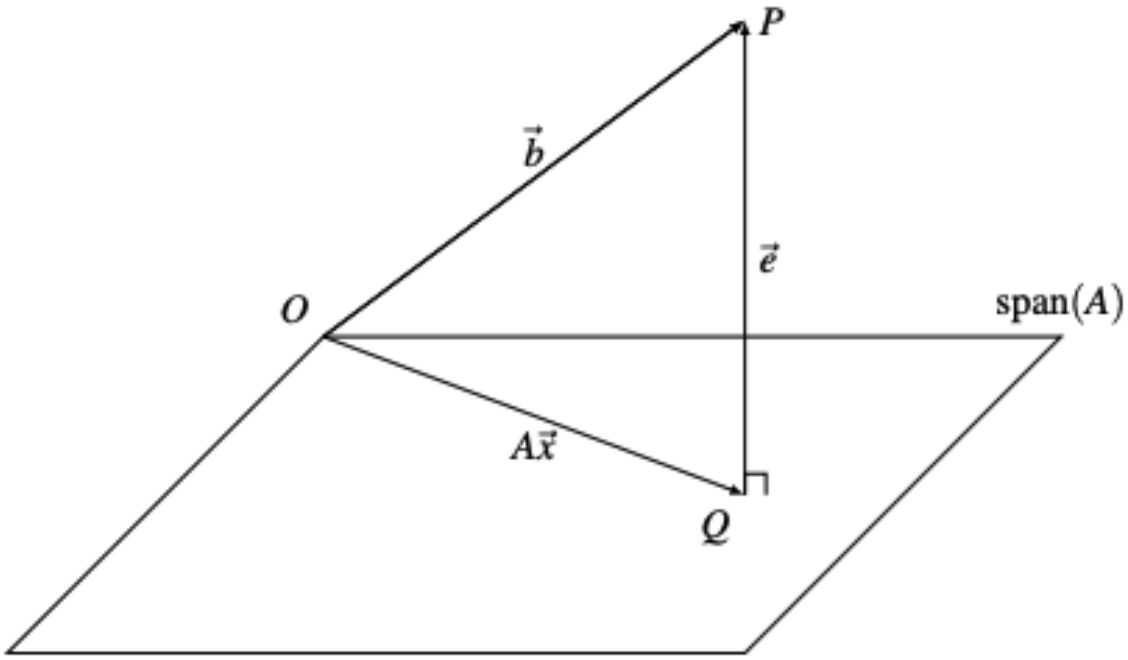
Least squares is based on an attempt to provide a solution $\vec{x}$ to this equation:

$$\vec{b} = A\vec{x}$$

Where each column of A corresponds to some input variable, $\vec{b}$ corresponds to our measured outputs, and each row of A and $\vec{b}$ corresponds to a trial/measurement. We would like to solve for some vector $\vec{x}$ of coefficients that describes how the columns of A can be linearly combined to make $\vec{b}$, however, this is often impossible, due to noise and such (and our system is usually overdetermined in a case like this), so, we take what we can get. We'll call our best estimate $\hat{\vec{x}}$, and introduce a term $\vec{e}$ to represent the difference between

what we would estimate our output to be if we were to execute $A\hat{\vec{x}}$, and our output actually was when we measured for $\vec{b}$. We now have the equation:

$\vec{b} = A\hat{\vec{x}} + \vec{e}$



As you can see in this diagram, A$\vec{x}$ is the projection of $\vec{b}$ on the span of A. This perfectly reflects our equation above and also supports the main crux of least squares: minimize $\vec{e}$ so that our approximation A$\vec{x}$ is as close to $\vec{b}$ as possible.

And with this we have enough degrees of freedom to achieve the much more realistic goal of finding some $\hat{\vec{x}}, \vec{e}$ that satisfies the above equation *and* has the minimal possible $\vec{e}$. We know that $A\hat{\vec{x}}$ is by definition in the column space of A, so if we want to minimize $\vec{e}$ in our sum we want no component of $\vec{e}$ to lie in the column space of A—in other words, the minimum possible $\vec{e}$ will be orthogonal to the column space of A, which allows us to create this additional constraint:

$A^T\vec{e} = \vec{0}$

You've almost certainly seen that picture of the plane with the vector sticking out of it that's decomposed into $A\hat{\vec{x}}$ and $\vec{e}$, where the three vectors form a right triangle—this reasoning is where that picture comes from. The above equation is the mathematical expression of that picture, which we can apply to solve for $\hat{\vec{x}}$. First, we'll rewrite our original equation to be $\vec{e} = \vec{b} - A\hat{\vec{x}}$, and then we'll plug this into the above constraint to get:

$A^T(\vec{b} - A\hat{\vec{x}}) = \vec{0}$

$A^T\vec{b} - A^TA\hat{\vec{x}} = \vec{0}$

$A^T\vec{b} = A^TA\hat{\vec{x}}$

$\hat{\vec{x}} = (A^TA)^{-1}A^T\vec{b}$

And that's where our famous formula comes from! Notably, we are assuming that A^TA is invertible—as

far as 16A is concerned it will be so long as we take **good** measurements.

Least squares comes up all over the place, but in the context of 16A it is particularly useful for trilateration—we will only be able to solve for the intersections of our three circles if all three of them align in *exactly the same point in space*, which given the fallibility of the physical world (the physical world is just the worst) is very unlikely. So, we take a bunch of measurements (often *way* more than you might think we need, to eliminate noise), and plug them into this formula to get our best estimate! We will also see below that it's a critical part of OMP.

The derivation of least squares is itself intellectually neat and all, but personally I would recommend emphasizing instead how to apply this formula—students will have this derivation drilled into them time and time again, but being able to take data and apply least squares to it to make generalizations is also super important and valuable and definitely something they'll be asked to do. Some good questions to ask:

- Given a table of measurements, how do I identify the entries of my A matrix and my $\vec{b}$ vector so that I can apply the least squares?
- How do I apply least squares to find the coefficients of a non-linear system (e.g. $y = ax^2 + bx + c$)?
- Can I apply least squares to circuits problems (e.g. finding an unknown resistance)? (The answer: YES OF COURSE)
- How do I know when I *won't* be able to use least squares?

5 Orthogonal Matching Pursuit (OMP)

We're in the home stretch everyone! OMP is the last part of Module 3, and it'll tie everything together. The premise of OMP is that we have n satellites that are each transmitting some periodic signal $\vec{s}_1, \cdots, \vec{s}_n$ to a receiver. Because all of the satellites are at different distances from our receiver, we will actually receive each $\vec{s}_i$ at some offset τ_i. Additionally, in OMP each s_i as received will be multiplied by some constant a_i that encodes some additional information. Our receiver will receive all of these offset, amplified signals kind of jumbled together, so that we, listening on the other end, receive some signal:

$$\vec{r} = a_1 \vec{s}_1^{(\tau_1)} + \cdots + a_n \vec{s}_n^{(\tau_n)}$$

In this problem, we assume that we know ahead of time all of the $\vec{s}_i$, and so our goal is to find $a_1, \cdots, a_n$ and $\tau_1, \cdots, \tau_n$ to dissect our received signal into its components—that's a lot of unknowns! So far, we don't in 16A have a tool to solve a problem like this, but OMP is an algorithm that can do just that:

1. Take the circular correlation of $\vec{r}$ with all $\vec{s}_i$. Find the maximum value across all of these correlation vectors, i.e., the choices of $\vec{s}_i$ and τ_i that maximize $|\text{circcorr}(\vec{r}, \vec{s}_i)[\tau_i]|$ (our a_i may be negative, so we will consider large negative correlations as well). We will then assume that the vector $\vec{s}_i$ at a shift of τ_i has a predominating presence in $\vec{r}$.
2. We now have an estimate $\hat{\vec{r}} = a_1 \vec{s}_i^{(\tau_i)}$, where a_1 is still unknown. We can solve for a_1 by turning this into a least squares problem, where $A = \vec{s}_i^{(\tau_i)}$, $\vec{x} = a_1$, and $\vec{b} = \vec{r}$, and then apply our formula! (another amazing application of least squares :0)
3. We will be left with some residual vector $\vec{e} = \vec{r} - \hat{\vec{r}} = \vec{r} - a_1 \vec{s}_i^{(\tau_i)}$. If we can find the composition of $\vec{e}$ in terms of the remaining $\vec{s}_i$, then we'll have an even better estimate. We can do this by much the same process:
   (a) Again, take the circular correlation of $\vec{e}$ with all $\vec{s}_i$ and find the $\vec{s}_j^{(\tau_j)}$ that maximizes our correlation.
   (b) Augment the matrix A from the last time we did least squares by adding a column $\vec{s}_j^{(\tau_j)}$. Re-execute least squares to solve for new a coefficients (note that the a_i that correspond to the signals we have already found may change).

(c) Update our residual $\vec{e} = \vec{e} - \hat{\vec{e}}$

(d) Repeat!

This process is **super tedious** to do by hand, which is why we have computers, thank goodness. So, instead of drilling in the process with examples (if you're super thirsty for that you can dig into the course notes), I'd like to highlight the elements of the design of this algorithm that make it work:

- OMP is based on the assumption that, for each $\vec{s}_i^{(\tau_i)}$ that is a component of $\vec{r}$, the correlation of $\vec{r}$ and $\vec{s}_i$ at shift τ_i will be relatively high, and the correlation of $\vec{r}$ with $\vec{s}_i$ at any other timeshift or with any signals that are not part of our sum will be relatively low/close to zero. This is partly assured by the property of inner products that the inner product of a vector with itself will be relatively high, but we can strengthen this assumption even more by carefully choosing our $\vec{s}_i$ based on the gold codes—a **magical** collection of vectors, where each vector's correlation with itself at a shift of zero is relatively large, and each vector's correlation with any other vector in the set/with itself at a non-zero shift is relatively close to zero. Some wizard did the hard work for us and found these (as far as 16A is concerned) so that we can just use them.

- One of the properties of least squares is that, by design, our residual vector $\vec{e}$ is orthogonal to the column space of our matrix A. Because the columns of A are made out of the $\vec{s}_i^{(\tau_i)}$ that have already been chosen. This, along with with the facts mentioned in the previous bullet point, makes it very very unlikely that when we run OMP on $\vec{e}$ we will choose an $\vec{s}_i$ that has already been chosen. So, the number of vectors in our sum will grow with each iteration.

- OMP will almost certainly not find *exactly* the right composition of $\vec{r}$, and sometimes we don't care for that level of precision anyways, so we have to put some kind of a stopping constraint in place. We can either limit the number of vectors chosen to be in our sum, which will put a hard cap on the number of iterations of OMP we execute, or we can put a limit on the size of the residual $\vec{e}$ before we decide that we are close enough. In general, we'll do both.

- Oftentimes we will use OMP when there are *a bunch* of signals, but we are told that our received signal $\vec{r}$ is sparse: in our sum $\vec{r} = a_1\vec{s}_1^{(\tau_1)} + \cdots + a_n\vec{s}_n^{(\tau_n)}$, most of the a_i are zero. OMP thrives in this situation because it is "greedy": it will find the best solution possible made up of as few signals as you want, by iteratively removing the signals that correspond to the greatest a_i, and ignore the other signals. Pretty neat!

OMP is of particular interest to us as we are designing this GPS system because, in terms of satellites, the timeshifts τ_i correspond to delays between the transmission and reception of signals. If a satellite is transmitting some periodic signal to my phone, my phone's going to receive this signal on some delay proportional to the speed of sound, which corresponds to a circular shift in the signal. We can then convert these delays into distances, which we we can pass as information to our trilateration algorithm. This course note from last semester describes how exactly to do this using **more linear algebra**, which is neat but in my experience not heavily emphasized—so long as students understand the underlying intention of this process, the finer details can be abstracted away.