EECS 16A Designing Information Devices and Systems I
Spring 2021 Homework 12

This homework is due April 23, 2021, at 23:59.
Self-grades are due April 26, 2021, at 23:59.

Submission Format
Your homework submission should consist of **one** file.

- `hw12.pdf`: A single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.
  If you do not attach a PDF "printout" of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible. Assign the IPython printout to the correct problem(s) on Gradescope.

Submit the file to the appropriate assignment on Gradescope.

1. Reading Assignment

For this homework, please review Note 22 (Trilateration and Correlation), and read Note 23 (Least Squares). You are always encouraged to read beyond this as well.

(a) In trilateration, the distances between the beacons and the unknown location $\vec{x}$ involve quadratic terms of $\vec{x}$. What trick can we use to get a system of linear equations in $\vec{x}$?

Solution: We can get a system of linear equations by subtracting one non-linear equation from another and eliminating the quadratic terms.

(b) Suppose the signal $x[n]$ is only defined for timesteps $0, 1, \cdots, 5$. For the purpose of computing linear cross-correlation, what value of $x[n]$ do we assume when n is a timestep out of the range: $0 \leq n \leq 5$ (e.g. $n = 6$ or $n = -1$)?

Solution: When n is a timestep out of the range: $0 \leq n \leq 5$, we consider $x[n]$ to be zero.

2. Mechanical Trilateration

Learning Goal: *The objective of this problem is to practice using trilateration to find the position based on the distance measurements and known beacon locations.*

Trilateration is the problem of finding one's coordinates given distances from known beacon locations. For each of the following trilateration problems, you are given 3 beacon locations ($\vec{s}_1$, $\vec{s}_2$, $\vec{s}_3$) and the corresponding distance (d_1, d_2, d_3) from each beacon to your location. Find your location or possible locations. If a solution does not exist, state that it does not.

(a) $\vec{s}_1 = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$, $d_1 = 5$, $\vec{s}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $d_2 = 2$, $\vec{s}_3 = \begin{bmatrix} -11 \\ 6 \end{bmatrix}$, $d_3 = 13$.

Solution: Here we show a general approach to the trilateration problem, so that we can immediately write the linear system of equations for all three parts. However, if you solved directly using concrete values, give yourself full credit.

$$\begin{aligned}\|\vec{x}-\vec{s}_1\|^2 &= d_1^2\\ \|\vec{x}-\vec{s}_2\|^2 &= d_2^2\\ \|\vec{x}-\vec{s}_3\|^2 &= d_3^2\end{aligned}$$

We can expand each left hand side out in terms of the definition of the norm:

$$\|\vec{x}-\vec{s}_i\|^2 = \langle \vec{x}-\vec{s}_i, \vec{x}-\vec{s}_i\rangle = (\vec{x}-\vec{s}_i)^T(\vec{x}-\vec{s}_i)$$

$$\begin{aligned}\vec{x}^T\vec{x} - 2\vec{x}^T\vec{s}_1 + \vec{s}_1^T\vec{s}_1 &= d_1^2\\ \vec{x}^T\vec{x} - 2\vec{x}^T\vec{s}_2 + \vec{s}_2^T\vec{s}_2 &= d_2^2\\ \vec{x}^T\vec{x} - 2\vec{x}^T\vec{s}_3 + \vec{s}_3^T\vec{s}_3 &= d_3^2\end{aligned}$$

Finally, take one equation and subtract it from the other two to get a system of linear equations in $\vec{x}$:

$$\begin{aligned}2\vec{x}^T\vec{s}_3 - 2\vec{x}^T\vec{s}_1 &= d_1^2 - d_3^2 + \vec{s}_3^T\vec{s}_3 - \vec{s}_1^T\vec{s}_1\\ 2\vec{x}^T\vec{s}_3 - 2\vec{x}^T\vec{s}_2 &= d_2^2 - d_3^2 + \vec{s}_3^T\vec{s}_3 - \vec{s}_2^T\vec{s}_2\end{aligned}$$

We can express as a matrix equation in $\vec{x}$:

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

We have that:

$$\begin{aligned}2(\vec{s}_3-\vec{s}_1) &= \begin{bmatrix}-30\\ 2\end{bmatrix}\\ 2(\vec{s}_3-\vec{s}_2) &= \begin{bmatrix}-24\\ 14\end{bmatrix}\\ d_1^2 - d_3^2 + \|\vec{s}_3\|^2 - \|\vec{s}_1\|^2 &= 25 - 169 + 157 - 41 = -28\\ d_2^2 - d_3^2 + \|\vec{s}_3\|^2 - \|\vec{s}_2\|^2 &= 4 - 169 + 157 - 2 = -10\end{aligned}$$

Which gives us the system $\begin{bmatrix}-30 & 2\\ -24 & 14\end{bmatrix}\vec{x} = \begin{bmatrix}-28\\ -10\end{bmatrix}$ with solution $\vec{x} = \begin{bmatrix}1\\ 1\end{bmatrix}$.
A solution existing for this system of linear equations does not necessarily guarantee consistency of the system of nonlinear equations, but we can validate:

$$\begin{aligned}\left\|\begin{bmatrix}1\\ 1\end{bmatrix} - \begin{bmatrix}4\\ 5\end{bmatrix}\right\|^2 &= \left\|\begin{bmatrix}-3\\ -4\end{bmatrix}\right\|^2 = 25 = d_1^2\\ \left\|\begin{bmatrix}1\\ 1\end{bmatrix} - \begin{bmatrix}1\\ -1\end{bmatrix}\right\|^2 &= \left\|\begin{bmatrix}0\\ 2\end{bmatrix}\right\|^2 = 4 = d_2^2\\ \left\|\begin{bmatrix}1\\ 1\end{bmatrix} - \begin{bmatrix}-11\\ 6\end{bmatrix}\right\|^2 &= \left\|\begin{bmatrix}12\\ -5\end{bmatrix}\right\|^2 = 169 = d_3^2\end{aligned}$$

(b) $\vec{s}_1 = \begin{bmatrix}0\\0\end{bmatrix}$, $d_1 = 5\sqrt{2}$, $\vec{s}_2 = \begin{bmatrix}10\\0\end{bmatrix}$, $d_2 = 5\sqrt{2}$, $\vec{s}_3 = \begin{bmatrix}5\\0\end{bmatrix}$, $d_3 = 5$.

Solution: Using what was shown in part (a) we have that:

$$\begin{aligned}2(\vec{s}_3 - \vec{s}_1) &= \begin{bmatrix}10\\0\end{bmatrix}\\ 2(\vec{s}_3 - \vec{s}_2) &= \begin{bmatrix}-10\\0\end{bmatrix}\\ d_1^2 - d_3^2 + \|\vec{s}_3\|^2 - \|\vec{s}_1\|^2 &= 50 - 25 + 25 - 0 = 50\\ d_2^2 - d_3^2 + \|\vec{s}_3\|^2 - \|\vec{s}_2\|^2 &= 50 - 25 + 25 - 100 = -50\end{aligned}$$

Which gives us the system $\begin{bmatrix}10 & 0\\-10 & 0\end{bmatrix}\vec{x} = \begin{bmatrix}50\\-50\end{bmatrix}$ with solution $\vec{x} = \begin{bmatrix}5\\\alpha\end{bmatrix}$. However, not all values of α are valid, so we check with the third distance equation:

$$\left\|\begin{bmatrix}5\\\alpha\end{bmatrix} - \begin{bmatrix}5\\0\end{bmatrix}\right\|^2 = 5^2 \implies \alpha^2 = 25 \implies \alpha = \pm 5$$

The system of nonlinear equations is consistent with this solution. We do not have enough information to uniquely determine our location, but we know we are at either $\vec{x} = \begin{bmatrix}5\\5\end{bmatrix}$ or $\vec{x} = \begin{bmatrix}5\\-5\end{bmatrix}$.

(c) $\vec{s}_1 = \begin{bmatrix}3\\4\end{bmatrix}$, $d_1 = 5$, $\vec{s}_2 = \begin{bmatrix}0\\-2\end{bmatrix}$, $d_2 = 2$, $\vec{s}_3 = \begin{bmatrix}-12\\5\end{bmatrix}$, $d_3 = 12$.

Solution: Using again what was shown in part (a) we have that:

$$\begin{aligned}2(\vec{s}_3 - \vec{s}_1) &= \begin{bmatrix}-30\\2\end{bmatrix}\\ 2(\vec{s}_3 - \vec{s}_2) &= \begin{bmatrix}-24\\14\end{bmatrix}\\ d_1^2 - d_3^2 + \|\vec{s}_3\|^2 - \|\vec{s}_1\|^2 &= 25 - 144 + 169 - 25 = 25\\ d_2^2 - d_3^2 + \|\vec{s}_3\|^2 - \|\vec{s}_2\|^2 &= 4 - 144 + 169 - 4 = 25\end{aligned}$$

Which gives us the system $\begin{bmatrix}-30 & 2\\-24 & 14\end{bmatrix}\vec{x} = \begin{bmatrix}25\\25\end{bmatrix}$. While a solution, $\vec{x} = \begin{bmatrix}-\frac{75}{93}\\\frac{75}{186}\end{bmatrix}$, for this system of linear equations exists, it will yield inconsistent distances when substituted back into the nonlinear equations. Therefore there is no solution.

3. Mechanical Projections

Learning Goal: *The objective of this problem is to practice calculating projection of a vector and the corresponding squared error.*

(a) Find the projection of $\vec{b} = \begin{bmatrix}3\\2\\-1\end{bmatrix}$ onto $\vec{a} = \begin{bmatrix}1\\0\\1\end{bmatrix}$. What is the squared error between the projection and $\vec{b}$, i.e. $\|e\|^2 = \|\text{proj}_{\vec{a}}(\vec{b}) - \vec{b}\|^2$?

Solution:

$$\text{proj}_{\vec{a}}(\vec{b}) = \frac{\langle \vec{b}, \vec{a} \rangle}{\|\vec{a}\|^2}\vec{a} = \frac{\vec{b}^T\vec{a}}{\|\vec{a}\|^2}\vec{a} \tag{1}$$

First, compute $\|\vec{a}\|^2 = \langle \vec{a}, \vec{a} \rangle = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = 2.$

Second, compute $\langle \vec{b}, \vec{a} \rangle = \begin{bmatrix} 3 & 2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = 2.$

Plugging in, $\text{proj}_{\vec{a}}(\vec{b}) = \frac{2\vec{a}}{2} = \vec{a}.$

The squared error between $\vec{b}$ and its projection onto $\vec{a}$ is $\|e\|^2 = \|\vec{a} - \vec{b}\|^2 = 12.$

(b) **(OPTIONAL)** Find the projection of $\vec{b} = \begin{bmatrix} 1 \\ 4 \\ -5 \end{bmatrix}$ onto the column space of $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$. What is the squared error between the projection and $\vec{b}$, i.e. $\|e\|^2 = \|\text{proj}_{\text{Col}(\mathbf{A})}(\vec{b}) - \vec{b}\|^2$?

Solution: Let $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $\vec{x} \in \mathbb{R}^2$ such that the projection of $\vec{b}$ onto the column space of $\mathbf{A}$ is $\mathbf{A}\vec{x}$.

We will compute $\hat{\vec{x}}$ by solving the following least squares problem,

$$\min_{\vec{x}} \|\mathbf{A}\vec{x} - \vec{b}\|^2 \tag{2}$$

The solution yields,

$$\hat{\vec{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\vec{b} \tag{3}$$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}^T \begin{bmatrix} 1 \\ 4 \\ -5 \end{bmatrix} \tag{4}$$

$$= \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -4 \\ 4 \end{bmatrix} \tag{5}$$

$$= \begin{bmatrix} -2 \\ 4 \end{bmatrix} \tag{6}$$

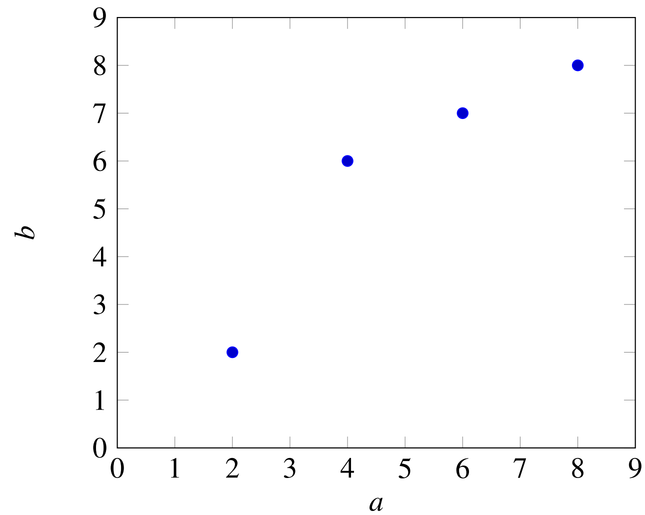
Plugging in, the projection of $\vec{b}$ onto the column space of $\mathbf{A}$ is $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -2 \\ 4 \end{bmatrix} = \begin{bmatrix} -2 \\ 4 \\ -2 \end{bmatrix}$.

The squared error between the projection and $\vec{b}$ is $\|\vec{e}\|^2 = \left\| \begin{bmatrix} -2 \\ 4 \\ -2 \end{bmatrix} - \begin{bmatrix} 1 \\ 4 \\ -5 \end{bmatrix} \right\|^2 = 18.$

4. Mechanical Least Squares

Learning Goal: *The goal of this problem is to use least squares to fit different models (i.e. equations) to a data set and find the squared error of each model.*

Depending on the least squares model's number of parameters, the model will fit the data better or worse. A better model results in a lower squared error than a worse one. In part (a), we consider a linear model that contains a single slope parameter and intercepts the vertical axis at zero. In part (b), we consider a linear model with a possibly non-zero vertical axis intercept parameter, also known as an affine model.



a	2	4	6	8
b	2	6	7	8

(a) Consider the above data points. Find the linear model of the form

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

that best fits the data, where x is a scalar that minimizes the squared error

$$\|\vec{e}\|^2 = \left\| \begin{bmatrix} a_1 \\ \vdots \\ a_4 \end{bmatrix} x - \begin{bmatrix} b_1 \\ \vdots \\ b_4 \end{bmatrix} \right\|^2 = \|\vec{a}x - \vec{b}\|^2.$$

Note that we can model this linear model as a generic system $\mathbf{A}\vec{x} = \vec{b}$ where $\mathbf{A} = [\vec{a}]$ and $\vec{x} = [x]$. Once you've computed the optimal solution $\hat{\vec{x}}$, compute the squared error between your model's prediction $\mathbf{A}\hat{\vec{x}}$ and the actual b values.

You may use a calculator but show your work. Do not directly plug your numbers into IPython.

Note: By using this linear model, we are implicitly forcing the line to go through the origin.

Optional but recommended: Plot the best fit line along with the data points to examine the quality of the fit. You may plot however you wish - one option is to use the helper code in the IPython notebook provided.

Solution:

Define $\vec{a} = \begin{bmatrix} 2 & 4 & 6 & 8 \end{bmatrix}^T$ and $\vec{b} = \begin{bmatrix} 2 & 6 & 7 & 8 \end{bmatrix}^T$. Applying the linear least squares formula, we get

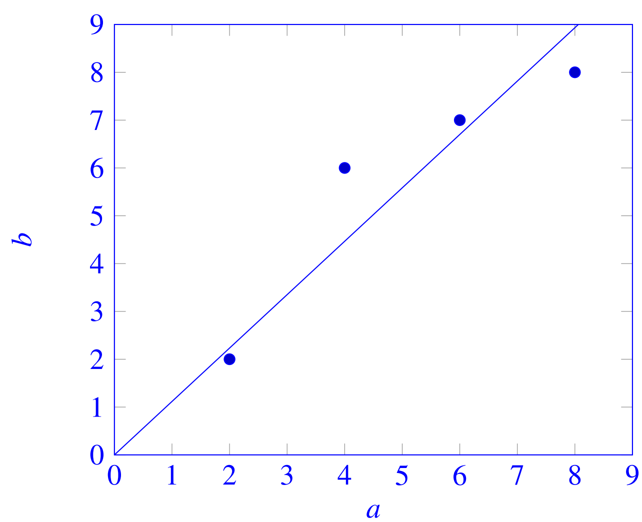
$$\begin{aligned}
\hat{\vec{x}} &= (\vec{a}^T\vec{a})^{-1}\vec{a}^T\vec{b} \\
&= \left(\begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix}^T \begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix} \right)^{-1} \begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix}^T \begin{bmatrix} 2 \\ 6 \\ 7 \\ 8 \end{bmatrix} \\
&= (120)^{-1}(134) = 1.1167
\end{aligned}$$

The error between the model's prediction $\hat{b}$ and actual values b is

$$\begin{aligned}
\vec{e} &= \hat{\vec{b}} - \vec{b} = \vec{a}\hat{\vec{x}} - \vec{b} \\
&= 1.1167 \begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix} - \begin{bmatrix} 2 \\ 6 \\ 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 0.234 \\ -1.534 \\ -0.3 \\ 0.934 \end{bmatrix}
\end{aligned}$$

and the sum of squared errors is

$$\vec{e}^T\vec{e} = 3.367$$



(b) Now, let us consider an affine model for the same data, i.e. one with a non-zero vertical b-intercept. We believe we can get a better fit for the data by assuming an affine model of the form

$$ax_1 + x_2 = b,$$

for each point. Note that x_1 is the slope and x_2 is the b-intercept here. We can write this equation jointly for all the points using the vector notation below:

$$\vec{a}x_1 + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} x_2 = \vec{b}.$$

Make sure you understand why a column vector of 1's is required above. In order to do this, we need to augment our $\mathbf{A}$ matrix from the previous part for the least squares calculation with a column of 1's (do you see why?), so that it has the form

$$\mathbf{A} = \begin{bmatrix} a_1 & 1 \\ \vdots & \vdots \\ a_4 & 1 \end{bmatrix}.$$

Set up a least squares problem to find the optimal x_1 and x_2 and compute the squared error between your model's prediction and the actual $\vec{b}$ values. Is it a better fit for the data? Provide a quantitative, numerical justification.

Optional: Plot your affine model and examine qualitatively how close the best fit line is to the data points compared to part (a). You may plot however you wish - one option is to use the helper code in the IPython notebook provided.

Solution:

Let $\vec{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. Using the least squares formula with the new augmented $\mathbf{A}$ matrix, we calculate the optimal approximation of $\vec{x}$ as

$$\begin{aligned} \hat{\vec{x}} &= \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\vec{b} \\ &= \left(\begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \\ 8 & 1 \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \\ 8 & 1 \end{bmatrix}\right)^{-1} \begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \\ 8 & 1 \end{bmatrix}^T \begin{bmatrix} 2 \\ 6 \\ 7 \\ 8 \end{bmatrix} \\ &= \begin{bmatrix} 120 & 20 \\ 20 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 2 & 4 & 6 & 8 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \\ 7 \\ 8 \end{bmatrix} \\ &= \frac{1}{120(4) - 20(20)} \begin{bmatrix} 4 & -20 \\ -20 & 120 \end{bmatrix} \begin{bmatrix} 134 \\ 23 \end{bmatrix} \\ \hat{\vec{x}} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} &= \begin{bmatrix} 0.95 \\ 1 \end{bmatrix} \end{aligned}$$

The linear model's prediction of $\vec{b}$ is given by

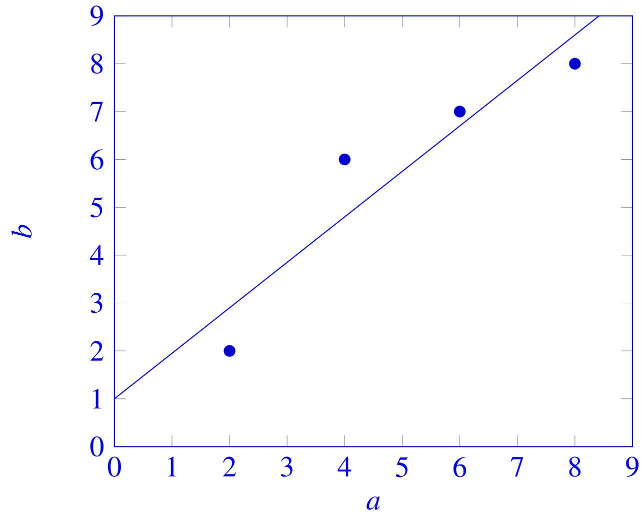
$$\hat{\vec{b}} = \mathbf{A}\hat{\vec{x}} = \begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \\ 8 & 1 \end{bmatrix} \begin{bmatrix} 0.95 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.9 \\ 4.8 \\ 6.7 \\ 8.6 \end{bmatrix}$$

and the error is given by

$$\vec{e} = \hat{\vec{b}} - \vec{b} = \begin{bmatrix} 0.9 & -1.2 & -0.3 & 0.6 \end{bmatrix}^T$$

The summed squared error is

$$\vec{e}^T\vec{e} = 2.7$$



We can see qualitatively from the plot that the line passes closer to the data points than the best fit line found previously. We see quantitatively that the sum of the squared errors is lower than that of the model found in part (a).

(c) **Prove the following theorem.**
Let $\mathbf{A} \in \mathbb{R}^{m \times n}$. If $\hat{\vec{x}}$ is the solution to the least squares problem

$$\min_{\vec{x}} \left\| \mathbf{A}\vec{x} - \vec{b} \right\|^2,$$

then the error vector $\mathbf{A}\hat{\vec{x}} - \vec{b}$ is orthogonal to the columns of $\mathbf{A}$, i.e. show that: $\mathbf{A}^T(\mathbf{A}\hat{\vec{x}} - \vec{b}) = \vec{0}$.
Note: Trying to show individually that $\langle \vec{a}_i, \mathbf{A}\hat{\vec{x}} - \vec{b} \rangle = \vec{a}_i^T(\mathbf{A}\hat{\vec{x}} - \vec{b}) = 0$ for $i = 0, \ldots, n$, where $\vec{a}_i$ is the ith column of $\mathbf{A}$ can be a bit tricky, however, stacking all of the $\vec{a}_i^T$ on top of each other and showing that $\mathbf{A}^T(\mathbf{A}\hat{\vec{x}} - \vec{b}) = \vec{0}$ is easier.
For this question, it is sufficient to prove that $\mathbf{A}^T(\mathbf{A}\hat{\vec{x}} - \vec{b}) = \vec{0}$.
Hint: Can you substitute $\hat{\vec{x}}$ using the least-squares formula?

Solution: We want to show that the error in the least squares estimate is orthogonal to the columns of the $\mathbf{A}$, i.e., we want to show that $\mathbf{A}^T(\mathbf{A}\hat{\vec{x}} - \vec{b})$ is the zero vector. Plugging in the linear least squares formula for $\hat{\vec{x}}$, we get

$$\begin{aligned}
&\mathbf{A}^T \left(\mathbf{A}\hat{\vec{x}} - \vec{b}\right) \\
&= \mathbf{A}^T \left(\mathbf{A}\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\vec{b} - \vec{b}\right) \\
&= \mathbf{A}^T\mathbf{A}\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\vec{b} - \mathbf{A}^T\vec{b} \\
&= \mathbf{I}\mathbf{A}^T\vec{b} - \mathbf{A}^T\vec{b} \\
&= \mathbf{A}^T\vec{b} - \mathbf{A}^T\vec{b} = \vec{0}
\end{aligned}$$

5. GPS Receivers

Learning Goal: *This problem will help to understand how GPS satellites transmit encoded signals to GPS receivers and how a receiver decodes the received signals and calculates the distance to the satellites using the signal propagation delays. It also shows how the GPS is designed to be immune to noise.*

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. In this problem, we will understand how a receiver (e.g. your cellphone) can disambiguate signals from the different GPS satellites that are simultaneously received.

GPS satellites employ a special coding scheme where each GPS satellite uses a unique 1023 element long sequence as its "signature." These codes used by the satellites are called "Gold codes," and they have some special properties:

- The auto-correlation of a Gold code (cross-correlation with itself) is very **high** at the 0^{th} shift and very **low** at all other shifts.
- The cross-correlation between different Gold codes is very **low** at all shifts, i.e. different Gold codes are almost orthogonal to each other.

The important thing to know is that the Gold codes are 1023 element vectors where each element is either $+1$ or -1, and that any Gold code is "almost orthogonal" to any other Gold code.

A receiver listening for signature transmissions from a satellite has copies of all of the different GPS satellites' Gold codes. The receiver can determine how long it took for a particular GPS satellite's signal to reach it by **taking the cross-correlation of the received signal with a satellite's Gold code. (The Gold code is the signal which is shifted during cross-correlation — as discussed in lecture.)** The shift value (delay) that corresponds to distinct peaks (positive/negative) in the correlation determines the "propagation delay" between when the GPS satellite transmitted it's signal and when the receiver received it. This time delay can then be converted into a distance (*in the case of GPS, electromagnetic waves are used for transmissions, distance is equal to the speed of light multiplied by the time delay*).

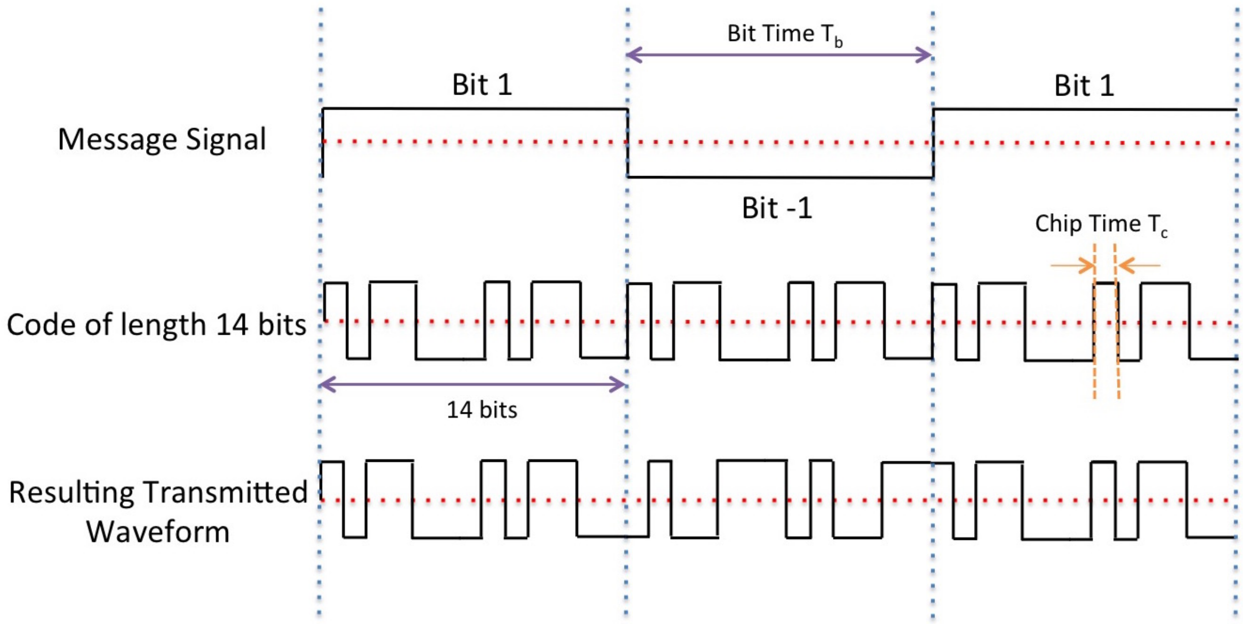
The GPS satellite is constantly transmitting its signature. In addition to identifying itself through its signature, it can also **"modulate" the signature** to communicate more information. Modulating a signature means multiplying the entire signature block by $+1$ or -1, as shown in the figure.

In the figure below, the signature is of length 14 as an example, i.e. it is made of 14 symbols (each one is either $+1$ or -1). T_c is the duration of one ± 1 symbol, and T_b is the duration of a whole signature. The figure shows 3 blocks of length 14 being transmitted. The message signal (made of $+1$ and -1 as well) multiplies the entire block of the signature, to give the resulting transformed waveform at the bottom of the figure. The message being transmitted in the figure is $\begin{bmatrix}1 & -1 & 1\end{bmatrix}$. So to send these three symbols of message, we need to send 14×3 symbols of the gold code.

Now, when a receiver receives a signal, in addition to finding the time delay between transmission and reception, the receiver will be able to decode the message by noting a very positive correlation if the message bit is equal to 1, and a very negative correlation if the message bit is equal to -1.

For the problem you will now do, $T_b = 1023T_c$.

You will use the ideas of linear correlation to figure out which of the satellites are transmitting.



For the purpose of this question we only consider 24 GPS satellites. Download the IPython notebook and the corresponding data files for the following questions.

Note: this code is calculation-heavy, and can take up to a few minutes to run for each code block. Be patient!

The iPython code required for part (a) is already given for you. You only need to write codes for parts (b)–(g).

(a) Auto-correlate (i.e. cross-correlate with itself) the Gold code of satellite 10 and plot it. What do you observe? Use the helper function `array_correlation` in the notebook to perform correlation for all sub-parts of this problem. Try to understand what the helper function `array_correlation` is doing.

Solution: The autocorrelation peaks at offset = 0 (when the signals are perfectly aligned). The correlation of a Gold code with a shifted version of itself is not significant.

(b) Cross-correlate the Gold code of satellite 10 with satellite 13 and plot it. What do you observe?

Solution: We see that the cross-correlation of a Gold code of any satellite with any other satellite is very low. Therefore, if we search for satellite 10 in a signal by cross-correlation and satellite 13 is the only one transmitting, then we will not get a correlation peak. This is true for any pair of satellites.

(c) Consider a random signal, i.e. a signal that is not generated due to a specific code but is a random ± 1 sequence. A helper function `integernoise_generator` in the notebook will generate this for you. Cross-correlate it with the Gold code of satellite 10. What do you observe? What does this mean about our ability to identify satellites in the presence of random ± 1 noise?

Solution: We see that the cross-correlation of the Gold code of any satellite with integer noise is very low. This indicates that we can still figure out the presence of a satellite even if it is buried in noise.

(d) The signal actually received by a receiver will be the satellites' transmissions plus additive noise, and this need not be just noise that takes values ± 1. Use the helper function `gaussiannoise_generator` in the notebook to generate a random noise sequence of length 1023, and compute the cross-correlation of this sequence with the Gold Code of satellite 10. What does this mean about our ability to identify satellites in the presence of real-valued noise?

For the next subparts of this problem, the received signals are corrupted by real-valued noise. Use the observation from this subpart for solving the rest of the question.

Solution: We see that the Gold code of any satellite with Gaussian noise is very low. This indicates that we can still figure out the presence of a satellite even if it is buried in Gaussian noise.

(e) The receiver may receive signals from multiple satellites simultaneously, in which case the signals will all be added together. In addition, noise might be added to the signal. What are the satellites present in the received signal from `data1.npy`?

Use helper function `find_peak` in your code to help you figure out whether peak correlation values of magnitude greater than a pre-specified threshold value are present. Note that the threshold is 800 for this problem.

Solution: The satellites that are present are satellites 4, 7, 13, and 19.

(f) Let's assume that you can hear only one satellite, Satellite X, at the location you are in (though this never happens in reality). Let's also assume that this satellite is transmitting an unknown sequence of $+1$ and -1 of length 5 (after encoding it with the 1023 bit Gold code corresponding to Satellite X).

First, find out from `data2.npy` which satellite is transmitting using the same procedure that you used in part (e).

Next, find the 5 element sequence of ± 1's that is being transmitted. To do this, you can observe the cross-correlation of the received signal from `data2.npy` with the Gold code of Satellite X and then visually find the peaks (positive /negative) and use these to understand the message.

Solution: Satellite 3 is transmitting 1, -1, -1, -1, 1.

(g) **(OPTIONAL)**

Signals from different transmitters arrive at the receiver with different delays. We use these delays to figure out the distance between the satellite and receiver.

The signals from different satellites are superimposed on each other with different offsets at the start.

What satellites are you able to see in `data3.npy`? Assume that all satellites begin transmission at time 0. What are the delays of all the satellites that are present? Assume you are told that all the satellites have the same message signal given by $\begin{bmatrix}1 & 1 & -1 & -1 & -1\end{bmatrix}$.

Solution: The satellites present in this data are 5 and 20.

The correlation array index where satellite 5's first peak is located is 253. The correlation array index where satellite 20's first peak is located is 506. These would correspond to delays of 253 and 506.

6. Audio File Matching

Learning Goal: *This problem motivates the application of correlation for pattern matching applications such as Shazam.*

Many audio processing applications rely on representing audio files as vectors, referred to as audio *signals*. Every component of the vector determines the sound we hear at a given time. We can use inner products to determine if a particular audio clip is part of a longer song, similar to an application like *Shazam*.

Let us consider a very simplified model for an audio signal, $\vec{x}$. At each timestep k, the audio signal can be either $x[k] = -1$ or $x[k] = 1$.

(a) Say we want to compare two audio files of the same length N to decide how similar they are. First, consider two vectors that are exactly identical, namely $\vec{x}_1 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$ and $\vec{x}_2 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$. What is the inner product of these two vectors? What if $\vec{x}_1 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$ but $\vec{x}_2$ oscillates between 1 and -1? Assume that N, the length of the two vectors, is an even number.

Use this to suggest a method for comparing the similarity between a generic pair of length-N vectors.

Solution:

The inner product of $\vec{x}_1 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$ and $\vec{x}_2 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$ is $\vec{x}_1 \cdot \vec{x}_2 = N$. The inner product of $\vec{x}_1 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$ and $\vec{x}_2 = \begin{bmatrix}1 & -1 & 1 & -1 & \cdots & 1 & -1\end{bmatrix}^T$ is $\vec{x}_1 \cdot \vec{x}_2 = 0$ when the vector length is even. To compare two vectors of length N composed of 1 and -1, we take the inner product of the two vectors, a large inner product means the vectors have a similar direction.

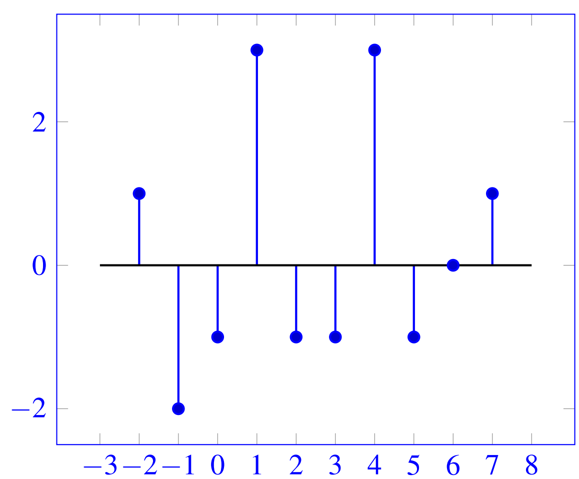
In many circumstances, an inner product with a very large negative value would mean the vectors are very different, but it turns out that humans are unable to perceive the sign of sound, so two sounds vectors $\vec{x}$ and $-\vec{x}$ sound exactly the same. As a result, for this problem we are interested in is the **absolute value** of the dot product, but in many other problems, we will interpret a large negative dot product as very different vectors. Don't take off points in parts (a), (b), or (c) if you didn't mention the absolute value.

(b) Next, suppose we want to find a short audio clip in a longer one. We might want to do this for an application like *Shazam*, which is able to identify a song from a short clip. Consider the vector of length 8, $\vec{x} = \begin{bmatrix}-1 & 1 & 1 & -1 & 1 & 1 & -1 & 1\end{bmatrix}^T$.

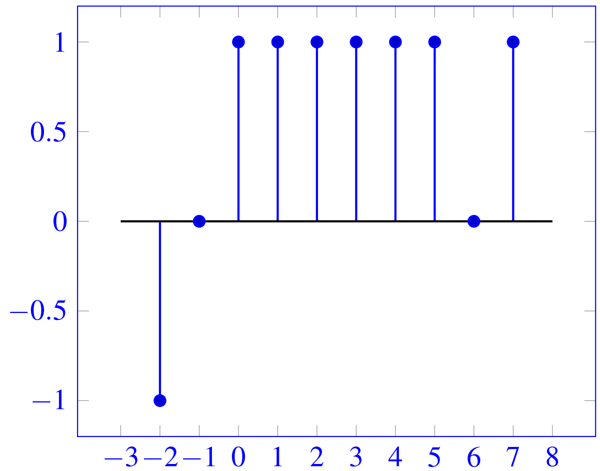
We want to find the short segment $\vec{y} := \begin{bmatrix}y[0] & y[1] & y[2]\end{bmatrix}^T = \begin{bmatrix}1 & 1 & -1\end{bmatrix}^T$ in the longer vector. To do this, perform the linear cross correlation between these two finite length sequences and identify at what shift(s) the linear cross correlation is maximized. Apply the same technique to identify what shift(s) gives the best match for $\vec{y} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$.

(If you wish, you may use iPython to do this part of the question, but you do not have to.)

Solution:



The above plot is $\text{corr}_{\vec{x}}(\vec{y})[k]$ where $\vec{y} = \begin{bmatrix}1 & 1 & -1\end{bmatrix}^T$. At shifts 1 and 4 the cross correlation is its *maximum possible value*, 3. These are both good matches.



The above plot is $\text{corr}_{\vec{x}}(\vec{y})[k]$ where $\vec{y} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$. At shifts 0 through 5 the cross correlation is only 1. There is not a really good match like before.

(c) Now suppose our audio vector is represented using integers beyond simply just 1 and -1. Find the short audio clip $\vec{y} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$ in the song given by $\vec{x} = \begin{bmatrix} 1 & 2 & 3 & 1 & 2 & 2 & 3 & 10 \end{bmatrix}^T$. Where do you expect to see the peak in the correlation of the two signals? Is the peak where you want it to be, i.e. does it pull out the clip of the song that you intended? Why?

(If you wish, you may use iPython to do this part of the question, but you do not have to.)

Solution:

Applying the technique in part (b), we get the best match to be $\begin{bmatrix} 2 & 3 & 10 \end{bmatrix}^T$ as this has the largest dot product with $\vec{y} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$. This is not where we expect to see the peak, as we observe the short audio clip $\vec{y}$ appears at the beginning of the song.

This happens because the volume at the end of the song is louder than the beginning of the song. Despite the angle not matching as well, the louder volume causes the linear cross correlation to be larger.

(d) Let us think about how to get around the issue in the previous part. We applied cross-correlation to compare segments of $\vec{x}$ of length 3 (which is the length of $\vec{y}$) with $\vec{y}$. Instead of directly taking the cross correlation, we want to normalize each inner product computed at each shift by the magnitudes of both segments, i.e. we want to consider $\frac{<\vec{x}_k, \vec{y}>}{\|\vec{x}_k\| \|\vec{y}\|}$, where $\vec{x}_k$ is the length 3 segment starting from the k-th index of $\vec{x}$. This is referred to as normalized cross correlation. Using this procedure, now which segment matches the short audio clip best?

Solution: Using the normalized cross correlation procedure, the best match for the short audio clip is at the 0^{th} shift and it perfectly matches the clip.

(e) We can use this on a more 'realistic' audio signal – refer to the IPython notebook, where we use normalized cross-correlation on a real song. Run the cells to listen to the song we are searching through, and add a simple comparison function `vector_compare` to find where in the song the clip comes from. Running this may take a couple minutes on your machine, but note that this computation can be highly optimized and run super fast in the real world! Also note that this is not exactly how Shazam works, but it draws heavily on some of these basic ideas.

Solution:

See `sol12.ipynb`.

7. Homework Process and Study Group

Who did you work with on this homework? List names and student ID's. (In case you met people at homework party or in office hours, you can also just describe the group.) How did you work on this homework? If you worked in your study group, explain what role each student played for the meetings this week.

Solution:

I first worked by myself for 2 hours, but got stuck on problem 5. Then I met with my study group.

XYZ played the role of facilitator ... etc. We were still stuck on problem 5 so we went to office hours to talk about the problem.

Then I went to homework party for a few hours, where I finished the homework.