EECS 16A Designing Information Devices and Systems I
Fall 2020 Homework 13

This homework is due December 4, 2020, at 23:59.
Self-grades are due December 7, 2020, at 23:59.

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

- `hw13.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

Review Note 22 (Trilateration and Correlation) and read Note 23 (Least Squares) and Note 25 (Trilateration). Write the equation to find the best-fit vector $\hat{\vec{x}}$ in a system modelled by $\mathbf{A}\vec{x} = \vec{b}$. When is this best-fit vector $\hat{\vec{x}}$ unique? *Hint: When does* $\mathbf{A}^T\mathbf{A}$ *have a unique inverse?*

2. Mechanical: Projections

(Contributors: Michael Kellman)

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. $\|e\|^2 = \|\text{proj}_{\vec{a}}(\vec{b}) - \vec{b}\|^2$?

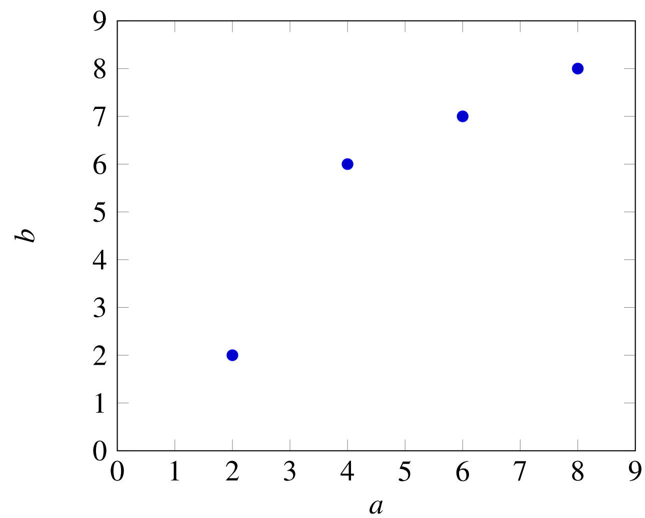
(b) Find the projection of $\vec{b} = \begin{bmatrix} 1 \\ 4 \\ -5 \end{bmatrix}$ onto the subspace spanned by the vectors $\left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\}$. What is the projection's squared error with $\vec{b}$, i.e. $\|e\|^2 = \|\text{proj}_{\vec{a}}(\vec{b}) - \vec{b}\|^2$?

3. Mechanical: Least Squares

(Contributors: Adhyyan Narang, Aviral Pandey, Jiazhen Chen, Michael Kellman, Wahid Rahman)

Learning Goal:

The goal of this problem is to use least squares to fit different models (i.e. equations) to a data set. Depending on the 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

$$\|\vec{e}\|^2 = \left\| \begin{bmatrix} a_1 \\ \vdots \\ a_4 \end{bmatrix} x - \begin{bmatrix} b_1 \\ \vdots \\ b_4 \end{bmatrix} \right\|^2 = \|\vec{a}x - \vec{b}\|^2. \tag{1}$$

Note: By using this linear model, we are implicitly forcing the line to go through the origin.
You may use a calculator but show your work. Do not directly plug your numbers into IPython. 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 and the actual b values as shown in Equation 1. **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.

(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.

(c) **Prove the following theorem.**
Let $\mathbf{A} \in \mathbb{R}^{m,n}$. 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?

4. GPS Receivers

(Contributors: Ava Tan, Aviral Pandey, Craig Schindler, Lam Nguyen, Michael Kellman, Moses Won, Terry Chern, Titan Yuan, Urmita Sikder, Vijay Govindarajan, Vasuki Narasimha Swamy)

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 system 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 "spread-spectrum" technology (very similar to Code-division multiple access, CDMA, which is commonly used in cellphone transmissions) and a special coding scheme where each transmitter is assigned a code that serves as its "signature".

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.

Gold codes are generated using a linear feedback shift register (LFSR). Understanding how this works is out of scope for the class, but you can read more about LFSR and CDMA if you are interested.

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).

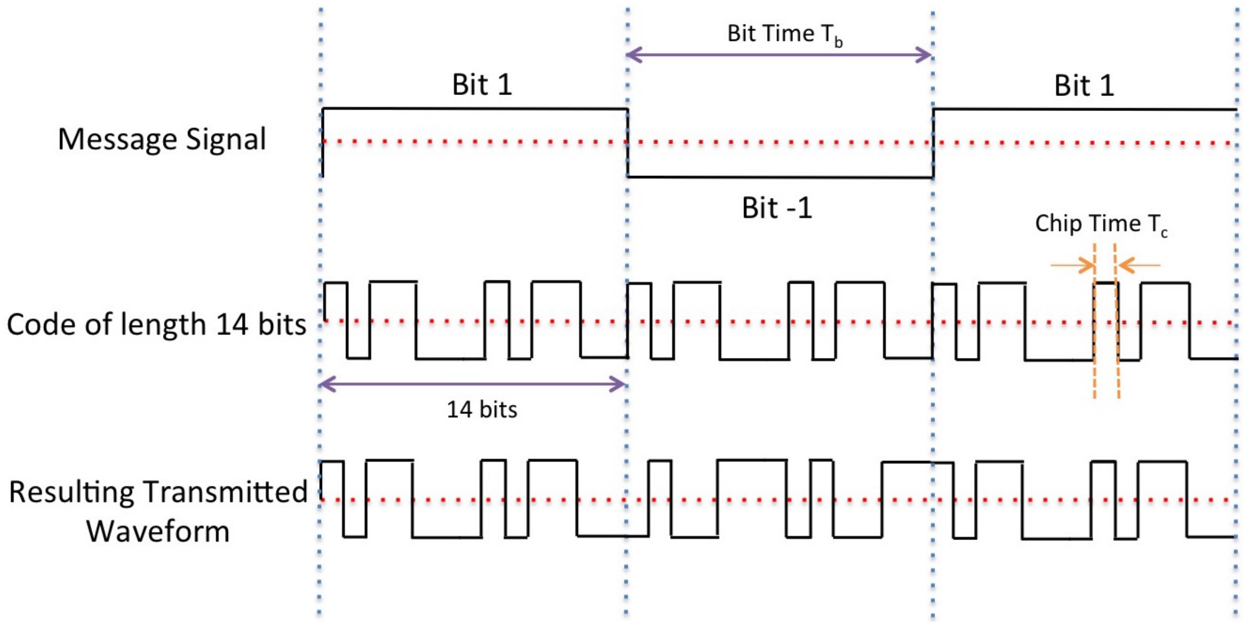
The GPS satellite is constantly transmitting its signature. In addition to identifying itself through its signature, it can also **"modulate" the signature** to communicate more information. Modulating a signature means multiplying the entire signature block by $+1$ or -1, as shown in the figure.

In the figure below, the signature is of length 14, i.e. it is made of 14 ± 1 symbols. 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.

The waveform that is actually transmitted is the multiplication of the message signal with the signature signal. 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 high 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, $T_b = 1023T_c$. (In reality, transmissions can be much faster and have $T_b = 20 \times 1023 \times T_c$.)

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 mins 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.

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

(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?

(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.

(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 is present. Note that the threshold is 800 for this problem.

(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.

(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.

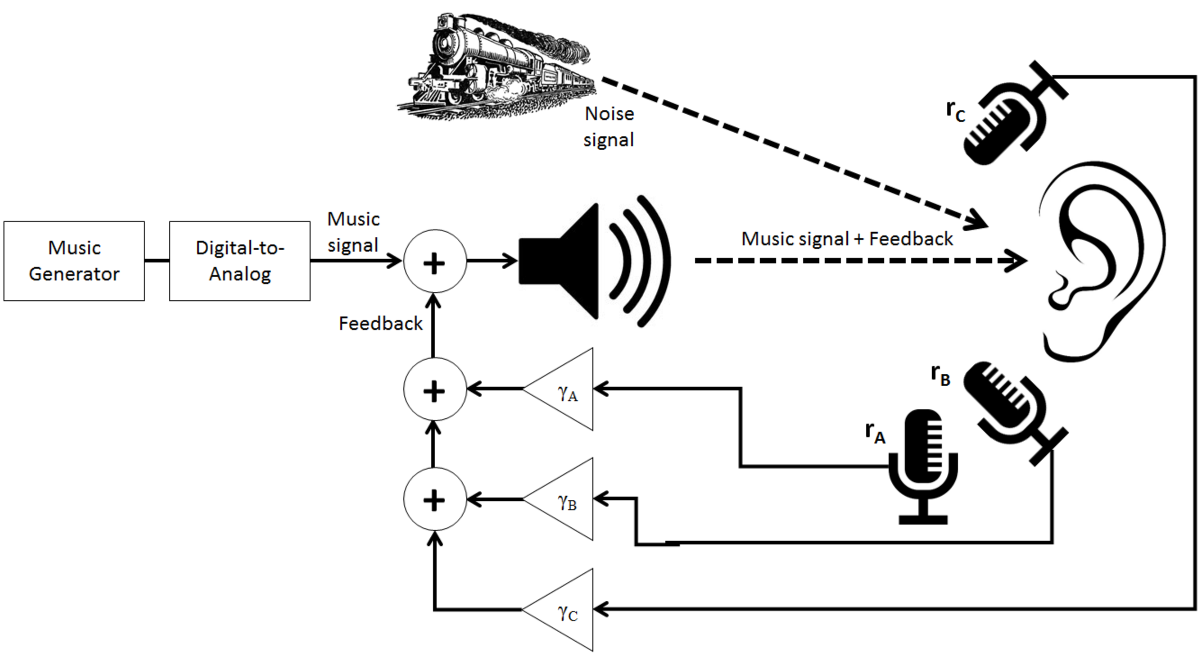
What satellites are you able to see in `data3.npy`? Assume that all satellites begin transmission at time 0. What are the delays of all the satellites that are present? Assume you are told that all the satellites have the same message signal given by $\begin{bmatrix} 1 & 1 & -1 & -1 & -1 \end{bmatrix}$.

5. Noise Canceling Headphones

(Contributors: Aviral Pandey, Lydia Lee, Michael Kellman, Moses Won, Urmita Sikder, Wahid Rahman)

Learning Goal: *This problem will employ least squares method to minimize the effect of additive noise through headphones.*

In this problem, we will explore a common design for noise cancellation using noise-canceling headphones as an example application. We will work with the model shown in the figure below.



A music signal is generated at a speaker and transmitted to the listener's ear. If there is noise in the environment (*e.g.* other people's voices, a train going by), this noise signal will be superimposed (i.e. added) on the music signal and the listener will hear both. In order to cancel the noise, we will try **to record the noise and subtract it directly from the transmitted signal** with the hope that we can achieve perfect cancellation of everything but the music. Since our system is imperfect, we'll have to solve a **least squares problem**.

The gain blocks marked by γ **(Greek "gamma") represent scalar multiplication**, and we will assume that they can take on any real number, positive or negative.

Assume we have an **additive noise signal** noted by $\vec{n}$, which we want to cancel. This represents the extra noise coming from the train.

$$\vec{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{bmatrix}.$$

Let us assume the music signal is represented by

$$\vec{m} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{bmatrix}.$$

In absence of any noise cancellation, the user will hear:

$$\vec{s} = \vec{m} + \vec{n}.$$

However, we want to cancel this noise. For this we use three microphones to record this noise, Mic A, Mic B, and Mic C. However, they cannot perfectly record the noise $\vec{n}$ and have erroneous recordings. Let $\vec{r}_A$, $\vec{r}_B$, and $\vec{r}_C$ represent the **noise that each microphone picks up**:

$$\vec{r}_A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}, \vec{r}_B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}, \vec{r}_C = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}.$$

We can arrange the recordings into a matrix $\mathbf{R}$ and the microphone gains, γ, into a vector $\vec{\gamma}$ to get:

$$\mathbf{R} = \begin{bmatrix} \vec{r}_A & \vec{r}_B & \vec{r}_C \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \\ a_5 & b_5 & c_5 \end{bmatrix}, \vec{\gamma} = \begin{bmatrix} \gamma_A \\ \gamma_B \\ \gamma_C \end{bmatrix}.$$

We want to choose a $\vec{\gamma}$ such that we minimize the impact of the noise.

The listener's ear will hear a total signal of:

$$\vec{s} = \vec{m} + \mathbf{R}\vec{\gamma} + \vec{n},$$

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix} + \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \\ a_5 & b_5 & c_5 \end{bmatrix} \begin{bmatrix} \gamma_A \\ \gamma_B \\ \gamma_C \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{bmatrix}.$$

This problem focuses on how to choose $\vec{\gamma}$ so as to have the signal $\vec{s}$ be as close to $\vec{m}$ as possible.

(a) Ideally, we would want to have a signal at the ear that matches the original music signal $\vec{m}$ perfectly. In reality, this is not possible, so we will aim to minimize the effect of the noise. What quantity would we need to minimize to make sure this happens? Write your answer in terms of the matrix $\mathbf{R}$, the vector of mic gains $\vec{\gamma}$, and the noise vector $\vec{n}$. *Hint: Your answer should be the norm of something.*

(b) We can solve the noise minimization problem by the least squares method. In effect, if we have a problem, $\min_{\vec{x}} \left\|\mathbf{A}\vec{x}-\vec{b}\right\|$, then the $\vec{x}$ that solves this problem is,

$$\hat{\vec{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\vec{b}. \tag{2}$$

Implement this least squares solution in the IPython Notebook helper function `doLeastSquares`. Remember that $\min_{\vec{x}} \left\|\mathbf{A}\vec{x}+\vec{b}\right\| = \min_{\vec{x}} \left\|\mathbf{A}\vec{x}-(-\vec{b})\right\|$.

(c) For the given $\vec{n}$ and the recordings, $\vec{r}_A$, $\vec{r}_B$, $\vec{r}_C$, below, find the γ's that minimize the effect of noise. Use the helper function `doLeastSquares` you made in the last part to solve this.

$$\vec{n} = \begin{bmatrix} 0.10 \\ 0.37 \\ -0.45 \\ 0.068 \\ 0.036 \end{bmatrix}, \vec{r}_A = \begin{bmatrix} 0 \\ 0.11 \\ -0.31 \\ -0.012 \\ -0.018 \end{bmatrix}, \vec{r}_B = \begin{bmatrix} 0 \\ 0.22 \\ -0.20 \\ 0.080 \\ 0.056 \end{bmatrix}, \vec{r}_C = \begin{bmatrix} 0 \\ 0.37 \\ -0.44 \\ 0.065 \\ 0.038 \end{bmatrix}.$$

The next few questions can be answered in the IPython notebook by running the associated cells.

(d) Follow the instructions in the IPython notebook to load a **music signal** and some **noise signals**. Listen to the music signal `music_y` and the two noise signals `noise1_y` and `noise2_y`. Which ones are full of static and which ones are not?

Also listen to the **noisy music signal** `noisyMusic`, generated by adding the music signal `music_y` and the first noise signal `noise1_y`. What do you hear?

(e) For this part we assume the noise signal $\vec{n}$ is given by `noise1_y` only. $\mathbf{R}$, i.e. the matrix of recorded noise from 3 microphones, is simulated with the `recordAmbientNoise` function. Calculate a vector $\vec{\gamma}$ that minimizes the effect of noise using `doLeastSquares`.

(f) Use you answers from the previous part to find the **noise-cancellation signal $\mathbf{R}\vec{\gamma}$**. Add this noise-cancellation signal ($\mathbf{R}\vec{\gamma}$), the music signal $\vec{m}$ (`music_y`), and the noise signal $\vec{n}$ (`noise1_y`), in order to generate a **noise-cancelled signal**. Play the noisy signal from part (e) and the noise-cancelled signal. Can you hear a difference?

(g) **[Optional]** Try adding the other noise signal `noise2_y` to the music signal to generate a new noisy signal. Don't re-calculate new values for $\vec{\gamma}$ i.e. don't solve the least squares problem again. Recalculate $\mathbf{R}$ and find the new noise-cancellation signal $\mathbf{R}\vec{\gamma}$ using $\vec{\gamma}$ from part (f). Add this noise-cancellation signal to your noisy signal. Comment on the quality of the resulting noise-cancelled signal. Is it perfect or are there artifacts?

Note: We want to find out how the $\vec{\gamma}$ we calculated work for any type of noise.

6. Image Analysis

(Contributors: Amanda Jackson, Ava Tan, Aviral Pandey, Lydia Lee, Michael Kellman, Panagiotis Zarkos, Spencer Kent, Titan Yuan, Urmita Sikder)

Learning Goal: *This problem introduces a method of fitting a non-linear model through a set of measured data points using the least squares method.*

Applications in medical imaging often require an analysis of images based on the image's pixels. For instance, we might want to count the number of cells in a given biological sample. One way to do this is to take a picture of the cells and use the pixels to determine their locations and how many there are. Automatic detection of shape is useful in image classification as well (e.g. consider a robot trying to find out autonomously where a mug is in its field of vision).

Let us focus back on the medical imaging scenario. You are interested in finding the exact position and shape of a cell in an image. You will do this by finding the **equation of the circle or ellipse** that bounds the cell relative to a given coordinate system in the image. Your collaborator uses edge detection techniques to find **a bunch of points that are approximately along the edge of the cell**. We assume that the origin of the coordinate system is in the center of the image with standard axes (x, y) and your collaborator gives you the following points that approximately bound the cell:

$(0.3, -0.69)$, $(0.5, 0.87)$, $(0.9, -0.86)$, $(1, 0.88)$, $(1.2, -0.82)$, $(1.5, 0.64)$, $(1.8, 0)$.

Recall that an equation of the form

$$a_1x^2 + b_1xy + c_1y^2 + d_1x + e_1y = 1$$

can be used to represent an ellipse (if $b_1^2 - 4a_1c_1 < 0$), and an equation of the form

$$a_1(x^2 + y^2) + d_1x + e_1y = 1$$

is a circle if $d_1^2 + e_1^2 + 4a_1 > 0$. Notice that the circle has fewer parameters.

(a) How can you find the equation of a *circle* that surrounds the cell by fitting the data points? First, provide a setup and formulate a minimization problem to do this, i.e. a least squares problem minimizing the squared error $\left\|\mathbf{A}\vec{v} - \vec{b}\right\|^2$, where you attempt to find the **unknown coefficients** a_1**,** d_1**, and** e_1 from your data points. Here your unknown vector $\vec{v} = \begin{bmatrix} a_1 \\ d_1 \\ e_1 \end{bmatrix}$. *Hint: The quantities* $(x^2 + y^2)$*, x, and y can be thought of as known values calculated from your data points.*

You do not need to simplify the numerical values for the matrix elements; just writing out the matrix with numerical expressions will suffice.

(b) How can you find the equation of an ellipse (instead of a circle) that surrounds the cell? Provide a setup and formulate a minimization problem similar to that in part (a). Now the unknown vector $\vec{v}$ will be different from the earlier parts.

You do not need to simplify the numerical values for the matrix elements; just writing out the numerical expressions will suffice.

(c) In the IPython notebook, write a short program that uses least-squares to fit a circle to the given points. A helper function `plot_circle` is provided. What is $\frac{\|\vec{e}\|}{N}$, where $\vec{e} = \mathbf{A}\vec{v} - \vec{b}$ and N is the number of data points? Plot your points and the best fit circle in IPython.

(d) In the IPython notebook, write a short program that uses least-squares to fit an ellipse to the given points. A helper function `plot_ellipse` is provided. What is $\frac{\|\vec{e}\|}{N}$, where $\vec{e} = \mathbf{A}\vec{v} - \vec{b}$ and N is the number of data points? Now the unknown vector $\vec{v}$ will be different from the earlier parts. Plot your points and the best fit ellipse in IPython. How does this error compare to the one in the previous subpart? Which technique is better, and why?

7. (PRACTICE, OPTIONAL) Labeling Patients Using Gene Expression Data

(Contributors: Aviral Pandey, Moses Won, Urmita Sikder)

Learning Goal: *This problem aims to design a predictive model using the least squares method on a set of training data and test the efficacy of the model using a set of test data.*

Least squares techniques are useful for many different kinds of prediction problems. Numerous researchers have extensively further developed the core ideas that we have learned in class. These ideas are commonly used in machine learning for finance, healthcare, advertising, image processing, and many other fields. Here, we'll explore how least squares can be used for classification of data in a medical context.

Gene expression data of patients, along with other factors such as height, weight, age, and family history, are often used to predict the likelihood that a patient might develop a certain disease. This data can be combined into a vector that describes each patient. This vector is often referred to as a feature vector.

Many scientific studies examine mice to understand how gene expression relates to diabetes in humans. Studies have shown that the expression of the ***tomosin2*** and ***ts1*** genes are correlated to the onset of diabetes in mice. How can we predict whether or not a mouse will develop diabetes based on data about this expression as well as other factors of the mouse? We will use some (fake) data to explore this.

We are given feature vectors for each mouse as:

$$\begin{bmatrix} \text{age} \\ \text{weight} \\ \text{tomosin2} \\ \text{ts1} \\ \text{chn1} \end{bmatrix}$$

Age and weight in the vector above are represented by real numbers, while the presence or absence of the expression of the genes *tomosin2*, *ts1*, and *chn1* is captured by $+1$ and -1 respectively. For example, the vector $\begin{bmatrix} 2 & 20 & 1 & -1 & -1 \end{bmatrix}^T$ means a 2 month old mouse, that weighs 20 grams, expresses the genes *tomosin2*, but not *ts1* or *chn1*.

We would like the following expression to be **positive if the mouse has diabetes and negative if the mouse does not have diabetes**:

$$f(\text{age}, \text{weight}, \text{tomoson2}, \text{ts1}, \text{chn1}) = \alpha_1(\text{age}) + \alpha_2(\text{weight}) + \alpha_3(\text{tomosin2}) + \alpha_4(\text{ts1}) + \alpha_5(\text{chn1}). \quad (3)$$

(a) We wish to set up a linear model for the problem in the format $\mathbf{A}\vec{x} = \vec{b}$. Here, $\vec{b}$ will be a vector with $+1, -1$ entries where a 1 represents that the mouse is diabetic and -1 represents that the mouse is not diabetic. The feature vectors of each mouse will be included in each row of the matrix $\mathbf{A}$. For example, if the first row of $\mathbf{A}$ contains the data of mouse #1 and the first entry of $\vec{b}$ is $+1$, that means Itchy the mouse #1 has diabetes.

Set up the problem by writing $\mathbf{A}$, $\vec{x}$, and $\vec{b}$ in terms of the variables in the feature vectors, α_i, and any other variables you define. What are your unknowns?

(b) Training data is data that is used to develop your model. The matrix $\mathbf{A}$ and vector $\vec{b}$, provided in `gene_data_train.npy` and `diabetes_train.npy` respectively, represent the (fake) *training* data. Use the data to find the optimal model parameters $\alpha_1, \ldots, \alpha_5$ for the given data set. Find the optimal parameter values using least squares method and the provided IPython notebook.

(c) Now it is time to use the model you have developed to make some predictions! It is interesting to note here that we are not looking for a real number to model whether each mouse has diabetes or not; we are looking for **a binary label**. Therefore, we will use the **sign of the expression from Equation (3)** to assign a ± 1 value to each mouse. Each mouse characteristics are represented by each row of `gene_data_test.npy`.

Predict whether each mouse with the characteristics in the *test* data set `gene_data_test.npy` will get diabetes. There are four mice/ rows in the test data set. Calculate the ± 1 prediction vector $\vec{b}_{calc}$ that shows the prediction whether a mouse will be diabetic or not. Observe the ± 1 vector $\vec{b}_{test}$ from `diabetes_test.npy` that indicates whether or not the mice *actually* have diabetes.

What is the prediction accuracy (number of correct predictions divided by total number of predictions) of your model?

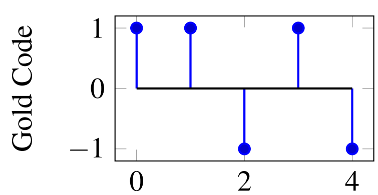
8. (PRACTICE, OPTIONAL) *Gold*en Positioning System (Fall '18, Final Exam)

(Contributors: Aviral Pandey, Ava Tan, Michael Kellman, Sang Min Han, Urmita Sikder, Wahid Rahman)

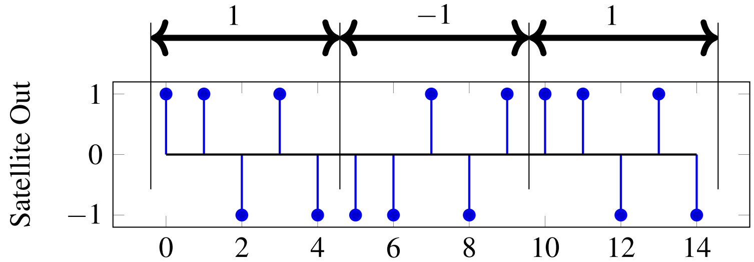
Learning Goal: *This past final exam problem is meant to help practice computing cross-correlation. It also covers the concept of trilateration.*

In this problem we will explore how real GPS systems work, and touch on a few aspects of implementing GPS receivers.

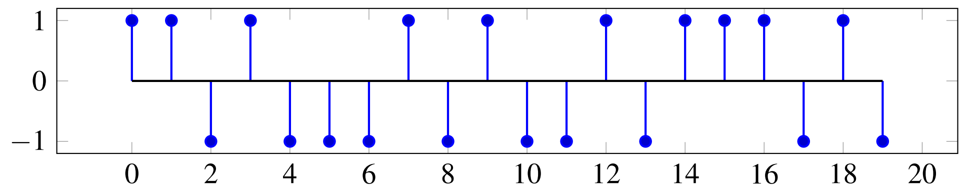
A Gold code is a sequence of 1's and -1's that has a high autocorrelation at a shift of 0, and small autocorrelations otherwise. Every GPS satellite has a unique Gold code assigned to it, and users are aware of the Gold code used by each satellite. The plot below shows a Gold code of length 5.



Each GPS satellite has a message that it transmits by modulating the Gold code. When the satellite is transmitting a 1, it sends just the Gold code sequence. When the satellite is transmitting a -1, it sends -1 times the Gold code. For example, if a satellite were transmitting the message [1, -1, 1], it would transmit the following (just as you have seen in the GPS homework problem):



(a) Suppose you receive the following from a GPS satellite that has the same Gold code as above. **What message is the satellite transmitting?**



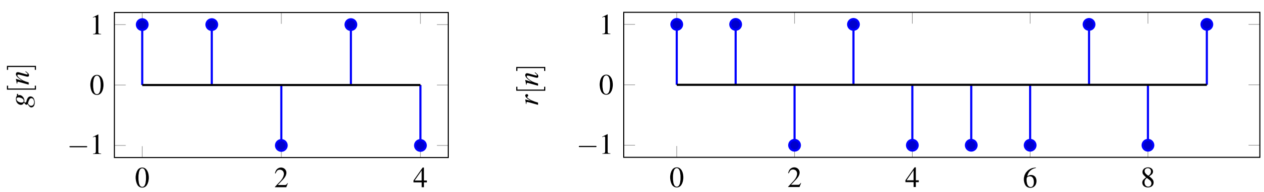
(b) In order to find the message being sent by the satellite, the receiver will find the linear cross-correlation of the received signal with a replica of the satellite Gold code.

We need to find the **linear cross-correlation** of the signals shown below given by

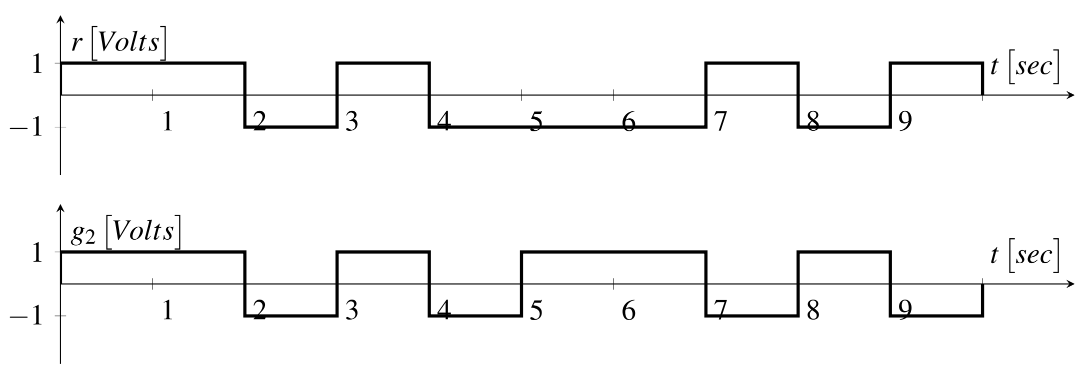
$$\text{corr}_{\vec{r}}(\vec{g})[k] = \sum_{i=-\infty}^{\infty} r[i]g[i-k]$$

where $r[n]$ is the received signal and $g[n]$ is the Gold code sequence. *Note that neither of these signals is periodic in this part.*

Plot the values of $\text{corr}_{\vec{r}}(\vec{g})[k]$ for $-1 \leq k \leq 7$. What is the significance of the peaks in the linear cross-correlation?



(c) Real GPS receivers have specialized hardware to perform cross-correlation using circuits. However, since these transmissions are continuous signals instead of discrete values, we will model the received signal $r(t)$ and the Gold code signal $g_2(t)$ as square waves, as shown in the plot below. Notice that $g_2(t)$ shows two periods of the Gold code.



An essential hardware block to implementing a GPS correlator is *Multiply and Integrate*. The Multiply and Integrate block takes in two inputs, then integrates the product of the two inputs over time. For example, the output of the *Multiply and Integrate* block given the above two inputs would be:

$$y(t) = \int_0^t r(\tau)g_2(\tau)d\tau$$

where $y(t)$ is the circuit output at time t. **Draw $y(t)$ as a function of time, for $t = 0$ to $t = 10$ sec.**

(d) Receivers also need to use the received data to calculate the position of the satellite. Each receiver will receive data from k satellites. Each satellite transmits the time, S_i, at which it started sending the message, where i is the index of the satellite, and $1 \leq i \leq k$. The receiver knows the time, T_i, at which each message arrives. You may assume the receiver and transmitter clocks are synchronized perfectly. Let c represent the speed of the signal.

Find an expression for d_i, the distance between the receiver and the i^{th} satellite, in terms of S_i, T_i, and other relevant parameters.

(e) Each satellite's position in 3D space is (u_i, v_i, w_i), where $1 \leq i \leq k$. The receiver position is given by (x, y, z). We need a linear system of equations the receiver can use to solve for its position, $\vec{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$.

Due to limitations of the hardware, the receiver can only handle **linear** systems of equations. **How many satellites must the receiver get data from to solve for its position?**

9. Study group survey

Please fill out the survey available here `https://forms.gle/nPCFKu1jQYzqtU1V7` to help us get feedback on how the study groups went so we can improve the experience for future semesters. We plan to implement an improved version of this system in multiple other classes next semester as well, so we really appreciate the input.

10. 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.