This is a 12 hour take-home midterm with 3 problems. Please submit your solutions to Gradescope at most 12 hours after you receive the exam.

- You may use any books, notes, or computer programs (e.g., MATLAB), but you may not discuss the exam with others until Nov. 9, after everyone has taken the exam and the solutions are posted. The only exception is that you can ask the course staff for clarification, by emailing to the staff email address ee263-fall1819-staff@lists.stanford.edu (please include the question number in your email subject, e.g., “Q1 EE263 Midterm ...”). We have tried pretty hard to make the exam unambiguous and clear, so we’re unlikely to say much. Please do not post any exam related questions on Piazza.

- Since you have 12 hours, we expect your solutions to be legible, neat, and clear. Do not submit your rough notes, and please try to simplify your solutions as much as you can. We will deduct points from solutions that are technically correct, but much more complicated than they need to be.

- Please check your email a few times during the exam, just in case we need to send out a clarification or other announcements. It’s unlikely we will need to do this, but you never know.

- If a problem asks for some specific answers, make sure they are obvious in your solutions. You might put a box around the answers, so they stand out from the surrounding discussion, justification, plots, etc.

- When a problem involves some computation (say, using MATLAB), we do not want just the final answers. We want a clear discussion and justification of exactly what you did, the well-commented MATLAB source code that produces the result, and the final numerical result or plots. Be sure to show us your verification that your computed solution satisfies whatever properties it is supposed to, at least up to numerical precision. For example, if you compute a vector $x$ that is supposed to satisfy $Ax = b$ (say), show us the matlab code that checks this, and the result. (This might be done by the matlab code norm(A*x-b); be sure to show us the result, which should be very small.) We will not check your numerical solutions for you, in cases where there is more than one solution.

- In the portion of your solutions where you explain the mathematical approach, you cannot refer to MATLAB operators, such as the backslash operator. (You can, of course, refer to inverses of matrices, or any other standard mathematical construct.)
• Some of the problems are described in a practical setting, such as computational Biology or machine learning. *You do not need to understand anything about the application area to solve these problems.* We have taken special care to make sure all the information and math needed to solve the problem is given in the problem description.

• Some of the problems require you to download and run a matlab file to generate the data needed. These files can be found at the URL
  
  http://ee263.stanford.edu/exams/mterm18/mterm18data.zip

• Please respect the honor code. Although we encourage you to work on homework assignments in small groups, *you cannot discuss the midterm with anyone*, with the exception of EE263 course staff, until Nov. 9, when everyone has taken it and the solutions are posted online.
1. *Bacteria farming [25 points]*

You are in the business of selling a special species of bacteria that is used in the development of a pharmaceutical drug. There are \( n \) different strains of the bacteria that have variable effectiveness in drug development. Further, when combined together in a batch, each bacteria strain affects the others’ growth rates, such that the overall growth of the system from week to week can be described by the discrete-time system
\[
x(k+1) = Ax(k),
\]
where \( A \in \mathbb{R}^{n \times n} \) and \( x_i(k) \) represents the quantity of strain \( i \) at the start of week \( k \). Each strain’s selling price, which is directly related to the value in drug development, is described in the vector \( r \), such that \( r_i x_i \) is the amount of money you would get for selling \( x_i \) amount of strain \( i \).

The Bacteria Farming Exposition is in \( T \) weeks, where you will get the opportunity to sell off your stock of bacteria. As a highly skilled bacteria trader, you know the bacteria market is very sensitive, and have projected that if you sell more than \( p \) total of bacteria \( (p \in \mathbb{R}) \), the market will crash. Your goal, then, is to sell exactly \( p \)-worth of bacteria after \( T \) weeks.

Your initial quantities of strains is given by \( x(0) \in \mathbb{R}^n \). Luckily, you have connections with the bacteria black market, which has a secret storage of particular strains of bacteria. You are able to borrow and return these particular strains from the black market. At the end of every week, you can make one trade, \( u(k) \), where you can take or give as much bacteria from the black market as you want. Note, \( u(k) \in \mathbb{R}^n \) and \( u_i(k) \) represents the amount of strain \( i \) you trade at time \( k \) (with \( u_i(k) > 0 \) implying borrowing and \( u_i(k) < 0 \) implying giving back). Unfortunately, you will accrue debt at a weekly rate \( \alpha \in \mathbb{R} \), where \( \alpha > 1 \), across all borrowed bacteria from the black market. So, if one week you bought 1.3 units of strain \( i \) and 1.4 units of strain \( j \), you will owe 1.3\( \alpha \) units of strain \( i \) and 1.4\( \alpha \) units of strain \( j \) to the black market next week. You will need a second linear system, \( y(k) \), that will keep track of your debt for each strain. You can assume \( y(0) = 0 \). You may also assume negative debt accrues at the same rate as the positive. That is, if you give bacteria to the black market, it will owe you at a weekly rate of \( \alpha \).

Finally, to be specific, on the day of the expo, you will be selling \( x(T) \) bacteria and you will have \( y(T) \) debt. When it’s time to sell your bacteria, *you must have zero debt on all bacteria*.

In summary, here are two typical weeks at your farm:

- At the start of week \( k \), you have \( x(k) \) bacteria and \( y(k) \) debt
- Your bacteria grows according to \( A \)
- You add \( u(k) \) to your grown bacteria
- Your debt increases by \( u(k) \)
- The week is over
• At the start of week $k+1$, your debt has grown by rate $\alpha$
• You now have $x(k+1)$ bacteria and $y(k+1)$ debt, and the cycle repeats.

(a) Before we get started, write your problem in the following form:

$$x(k + 1) = Ax(k) + Bu(k)$$
$$y(k + 1) = Cy(k) + Du(k)$$

Explicitly state $B$, $C$, and $D$. Note, $A$ is already given as part of the problem data, and it is possible that some of $B$, $C$, and $D$ will be a function of $\alpha$. We don’t need justification here, but if you happen to be wrong, we will give more partial credit if the justification makes sense. One to two sentence justification per matrix, maximum.

(b) Now, let’s use our $x(k + 1)$ equation to rewrite $x(T)$ in terms of only $x(0)$ and $u_{tot} = [u(0)\top \ u(1)\top \ldots u(T - 1)\top]\top$.

State the final $x(T) = Fx(0) + Gu_{tot}$ equation, where you show clearly how to build $F$ and $G$ from things we know.

(c) In order to have no debt at the end of $T$ weeks, we need to satisfy the constraint $y(T) = 0$. Reformulate this as a linear equality constraint on $u_{tot}$ (i.e. $Hu_{tot} = d$ for some appropriate $H$ and $d$). Clearly explain how $H$ and $d$ are constructed.

(d) We are finally ready to solve our problem! As a somewhat law-abiding citizen, you don’t like trading bacteria with the black market. Therefore, your goal is to minimize the net amount of bacteria you trade with the black market over the course of $T$ weeks. In other words, your goal is to minimize $\|u_{tot}\|_2$. You need to do this while hitting your profit target $\$p$ exactly, and ensuring no accumulated debt at the end of $T$ weeks. To clarify again, you will be selling $x(T)$ bacteria, and your zero debt must be $y(T) = 0$. First clearly state how you will setup and solve this problem to find $u_{tot}$, then apply your method to the data in bacteria data. Report proof that you get $Hu_{tot} = d$ (i.e. $\|Hu_{tot} - d\|$ is close to enough to 0 within some small tolerance, such as $10^{-5}$).

Once you’ve done that, using the formulas

$$x(k + 1) = Ax(k) + Bu(k)$$
$$y(k + 1) = Cy(k) + Du(k)$$

Make the following two plots: (i) your bacteria strain amounts from week to week for each strain (ii) your bacteria-debt amounts for each strain. Also, report $x(T)$. 

4
2. A sojourn in computer vision [40 points]

In computer vision, points can be naturally represented by vectors \( v \in \mathbb{R}^3 \), and linear transforms can be represented as matrix vector products. This allows us to easily manipulate points and groups of points which define objects. In this question, we will explore some of these manipulations.

One useful operation is a rotation. In class we saw a 2d rotation matrix. Here, we will consider a general 3d rotation matrix \( R \). Define \( R \in \mathbb{R}^{3 \times 3} \) such that the following properties hold:

- \( R \) is orthogonal
- The determinant of \( R \) is 1

(a) Prove that any rotation matrix \( R \in \mathbb{R}^{3 \times 3} \) has the following two properties
  
  i. \( R \) is magnitude preserving, i.e., \( \|Rv\|_2 = \|v\|_2, \ \forall v \in \mathbb{R}^3 \)
  
  ii. \( R \) is angle preserving, i.e., \( \angle(v, u) = \angle(Rv, Ru), \ \forall v, u \in \mathbb{R}^3 \)

(b) Sometimes we define rotation with respect to a coordinate axis (\( x \), \( y \), or \( z \) axis in \( \mathbb{R}^3 \)); however, sometimes it is useful to define rotation with respect to a certain vector. For example, we might want to rotate \( v \in \mathbb{R}^3 \) by \( \theta \) degrees around \( w \in \mathbb{R}^3, \ |w|_2 = 1 \), which we can express as a matrix-vector product \( v \text{rot} = R_w(\theta)v \).

\[ Figure 1: \text{Problem 2b setup} \]

i. First, express \( v \) as \( v = v_w + v_{w\perp} \) where \( v_w = \mu w \) for some \( \mu \in \mathbb{R} \) and \( v_{w\perp} = Aw \) for some \( A \in \mathbb{R}^{3 \times 3} \) (\( v_{w\perp} \) is orthogonal to \( w \)). Explicitly find \( \mu \) and \( A \) in terms \( v \) and \( w \).

Consider the subspace \( \mathcal{W}^\perp \) containing all vectors perpendicular to \( w \). Assume we have found an orthonormal basis for this subspace \( \{u_1, u_2\} \). Note that any vector
in $W^\perp$ can be written as a linear combination of $\{u_1, u_2\}$. Therefore, if we know we are working in this basis, we can write a vector $t \in W^\perp$ as

$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = t_1e_1 + t_2e_2 + t_3e_3 = \alpha_1u_1 + \alpha_2u_2 = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}_u = tu$$

Note that $t$ is defined in the standard basis $\{e_1, e_2, e_3\}$ whereas $tu$ is defined in the basis $\{u_1, u_2\}$. This notation is convenient, as it allows us to treat $t$ as a vector in $\mathbb{R}^2$ as long as we stay within this subspace. For example, if we want to translate $t$ within the subspace, we can first transform $t$ into the basis $\{u_1, u_2\}$, perform the translation there, then transform $t$ back to the standard basis.

ii. Write an expression for $v_{rot}$ in terms of $w, u_1, u_2, \theta,$ and $v$. What is $R_w(\theta)$? You do not need to simplify.

Another operation of interest is a perspective transform, which takes a vector in $\mathbb{R}^3$ and transforms it onto a plane in $\mathbb{R}^2$ (for example, a camera taking a picture). Consider a camera that looks down the positive $x$ axis, as shown in the diagram below. This camera will take a point $[x \ y \ z]^T \in \mathbb{R}^3$ and transform it onto the image plane, located at $x = 1$ and parallel to the $yz$ plane. In other words, the perspective transform does the following operation

$$v = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ y/x \\ z/x \end{bmatrix} = \frac{1}{x} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \tilde{v}$$

which we can also think of as scaling the vector $v = [x \ y \ z]^T$ by $\frac{1}{x}$ (Figure 2).

![Figure 2: The camera transforms a point $v$ to $\tilde{v}$, such that $\tilde{v}_1 = \tilde{x} = 1$](image-url)

Suppose the camera takes a picture with a point $v \in \mathbb{R}^3$ in it, and then the camera rotates by a small angle about the $z$ axis and takes another picture, again with $v$ in...
the field of view. The point \( v \) would have two different representations on the image plane, as the camera’s coordinate system rotates so that it always is looking down the \( x \) axis. However, \( v \) has not actually moved. As a result, we will introduce the notion of absolute coordinates and camera coordinates. In this case, the basis we use to define camera coordinates is the standard basis under some rotation.

(c) Consider a point \( v \) defined in absolute coordinates. You view \( v \) with a camera that has image plane at \( x = 1 \) in camera coordinates as described above (i.e., \( \tilde{v} \) in camera coordinates has \( \tilde{v}_1 = 1 \)). If the camera has been rotated by some angle \( \theta \) around the \( z \) axis relative to the absolute coordinate system, find an expression for \( \tilde{v} \) in terms of \( v \) and \( \theta \). Note: this \( \tilde{v} \) has undergone rotation as well as a perspective transform, as opposed to the \( \tilde{v} \) in Figure 2, where only a perspective transform has occurred.

**Hint:** the rotation matrix \( R_z(\theta) \) for rotation about the \( z \) axis by \( \theta \) might be helpful here:

\[
R_z(\theta) = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Now that you know how to match up camera coordinates with absolute coordinates, you figure that you can use a camera to locate something in the real world. You decide to build an autonomous Roomba\(^1\) to clean your bedroom. This Roomba can move in the \( x \) and \( y \) direction and can rotate around the \( z \) axis. We will call the direction that it is facing the \( x \) axis of the camera. Its own coordinate system is thus a rotation and translation of the absolute coordinate system. For simplicity, we’ll assume the Roomba sits on the absolute \( xy \) plane (i.e., its \( z \) position in absolute coordinates is always 0). Since the space of operation is constrained, you decide to develop the following system so that the Roomba knows where it is:

- You place beacons around the room at known positions \( p_i \in \mathbb{R}^3 \) in the absolute coordinate system.
- You put a camera on the Roomba that projects points onto an image plane at \( x = 1 \) in the Roomba’s coordinate system. Thus, it sees each \( p_i \) in the field of view as \( q_i = [1 \ q_{i,2} \ q_{i,3}]^T \)
- You label the beacons such that the Roomba always knows which one is which. (For example, maybe they are lights of varied wavelengths.) Assume \( N \) beacons \( p_1, p_2, \ldots, p_N \) are in the field of view.

We will try to estimate the position of the Roomba \( r = [x \ y \ 0]^T \) in absolute coordinates

---

\(^1\) A robot on wheels
and direction it is facing $\theta$ that minimizes the squared error:

$$E = \sum_{i=1}^{N} \|q_i - \hat{q}_i\|_2^2$$

where $\hat{q}_i = f(r, \theta, p_i)$ is the point on the image plane in camera coordinates that would result from our estimates of $r, \theta$. Note that $f$ is a nonlinear function.

(d) Explain how you would solve the problem of minimizing $E$. Clearly state all steps of your approach and write simplified expressions for all functions, matrices, vectors, etc. that you will be using in your approach. You should simplify your math to the point that it would be very easy to implement in code.

**Hint:** When taking the Jacobian of a complex function, remember the chain rule:

$$D_x f(g(x)) = \begin{bmatrix} \frac{\partial f_1}{\partial g_1} & \cdots & \frac{\partial f_1}{\partial g_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial g_1} & \cdots & \frac{\partial f_m}{\partial g_p} \end{bmatrix} \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_p}{\partial x_1} & \cdots & \frac{\partial g_p}{\partial x_n} \end{bmatrix},$$

where $f : \mathbb{R}^p \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^p$.

(e) How many beacons ($N$) must you see at a minimum for your method to work?

(f) Apply your method in part (2d) to the data given in `comp_vis_data`. To avoid numerical issues, consider your solution converged if you achieve an $E \leq 10^{-2}$.

Report the location and rotation of the Roomba. You should submit plots of (i) the value of $E$ per iteration, and (ii) the $(x, y)$ location of your guess for each iteration of your algorithm (this will look like a trajectory from your initial guess to the final). Note that the room is 10 $\times$ 10, so we will use $r(0) = [5 5 0]^T$ and $\theta(0) = 0$; that is, we begin by guessing that the Roomba is in the center of the room facing in the positive $x$ axis direction.

(g) Redo (2f) adding a regularization term of $\mu\|\hat{q}_i - \hat{q}_i^{(k-1)}\|_2^2$ to your objective function at the $k$th step of your algorithm (i.e. when solving for $\hat{q}_i^{(k)}$). Note that $\hat{q}_i^{(k-1)}$ is the $(k - 1)$th iterate of $\hat{q}_i$ and you should use $\mu = 10$. Provide plots of (i) the value of $E$ per iteration, and (ii) the $(x, y)$ location of your guess for each iteration. Compare these plots to those in (2f).
3. System power consumption optimization [35 points]

In an ordinary least squares problem, we are given $A \in \mathbb{R}^{m \times n}$ (skinny and full rank) and $y \in \mathbb{R}^m$, and we choose $x \in \mathbb{R}^n$ in order to minimize

$$\|Ax - y\|_2^2 = \sum_{i=1}^{m} (\hat{a}_i^T x - y_i)^2.$$  

Note that the penalty that we assign to a measurement error does not depend on the sensor from which the measurement was taken. However, this is not always the right thing to do: if we believe that one sensor is more accurate than another, we might want to assign a larger penalty to an error in the measurement from the more accurate sensor. We can account for differences in the accuracy of our sensors by assigning sensor $i$ a weight $w_i > 0$, and then minimizing

$$\sum_{i=1}^{m} w_i (\hat{a}_i^T x - y_i)^2.$$  

By giving larger weights to more accurate sensors, we can account for differences in the precision of our sensors.

(a) Weighted least squares. Explain how to choose $x$ in order to minimize

$$\sum_{i=1}^{m} w_i (\hat{a}_i^T x - y_i)^2,$$

where the weights $w_1, \ldots, w_m > 0$ are given.

(b) Iteratively reweighted least squares for $\ell_1$-norm approximation. Consider a cost function of the form

$$\sum_{i=1}^{m} w_i (x)(\hat{a}_i^T x - y_i)^2.$$  \hspace{1cm} (1)

One heuristic for minimizing a cost function of the form given in (1) is iteratively reweighted least squares, which works as follows. First, we choose an initial point $x^{(0)} \in \mathbb{R}^n$. Then, we generate a sequence of points $x^{(1)}, x^{(2)}, \ldots \in \mathbb{R}^n$ by choosing $x^{(k+1)}$ in order to minimize

$$\sum_{i=1}^{m} w_i (x^{(k)})(\hat{a}_i^T x^{(k+1)} - y_i)^2.$$  

Each step of this algorithm involves updating our weights, and solving a weighted least squares problem. Suppose we want to use this method to solve minimizing the $\ell_1$-norm approximation error, which is defined to be

$$\|Ax - y\|_1 = \sum_{i=1}^{m} |\hat{a}_i^T x - y_i|,$$
where the matrix $A \in \mathbb{R}^{m \times n}$ and the vector $y \in \mathbb{R}^m$ are given. How should we choose the weights $w_i(x)$ to make the cost function in (1) equal to the $\ell_1$-norm approximation error?

We now consider an important special case of the multi-objective least squares least squares problems. In piece-wise constant fitting problems, we start with a signal represented by a vector $x \in \mathbb{R}^n$ that is almost piece-wise constant. The entries $x_i$ correspond to the value of some function of time, evaluated (or sampled, in the language of signal processing) at equal time intervals. Such piece-wise constant signals are quite common in biological and medical systems, economics and electrical systems. In this problem we consider signals in one dimension, but the same ideas can be applied to signals in two or more dimensions, (e.g., images or video signals.)

Our measurement of the signal $x$ is corrupted by an additive noise $v$:

$$x_{\text{noisy}} = x + v.$$ 

The noise can be modeled in many different ways, but here we simply assume that it is unknown, small, and, unlike the signal, rapidly varying. The goal is to form a piece-wise constant estimate $\hat{x}$ of the original signal $x$, given the noisy measurement $x_{\text{noisy}}$. This process is called piece-wise constant fitting or piece-wise constant signal reconstruction.

One simple formulation of this fitting problem is the bi-objective problem

$$\min \left( \|\hat{x} - x_{\text{noisy}}\|_2^2, \phi(\hat{x}) \right),$$

where $\hat{x}$ is the optimization variable and $x_{\text{noisy}}$ is the problem data. The function $\phi : \mathbb{R}^n \to \mathbb{R}$ is a special regularization function. It is meant to measure how close the estimate $\hat{x}$ is to a piece-wise constant function. Hence, the optimization problem (2) seeks signals that are close (in $\ell_2$-norm) to the corrupted signal, and that are close to piece-wise constant, i.e., for which $\phi(\hat{x})$ is small. We can find the optimal trade-off curve for the optimization problem (2) by minimizing the weighted-sum objective for different values of $\mu \geq 0$:

$$\min \|\hat{x} - x_{\text{noisy}}\|_2^2 + \mu \phi(\hat{x}).$$

Now finding a good measure for piece-wise constantness of a signal is tricky business! Let’s define the first difference of the vector $\hat{x}$ as follows:

$$D\hat{x} = \begin{bmatrix} \hat{x}_2 - \hat{x}_1 \\ \hat{x}_3 - \hat{x}_2 \\ \vdots \\ \hat{x}_n - \hat{x}_{n-1} \end{bmatrix}.$$
A piece-wise constant \( \hat{x} \) will only have a few nonzero entries in its first difference \( D\hat{x} \). Therefore, the ideal measure for piece-wise constantness of a signal would be the number of nonzero entries in its first difference, and we can have:

\[
\phi(\hat{x}) = \text{card}(D\hat{x}),
\]

where the card function returns the number of nonzero entries in a vector. The problem with using this definition for our regularization function \( \phi(\hat{x}) \) is that it makes the optimization problem (3) almost impossible to solve. It turns out that in most practical applications, the \( \ell_1 \)-norm function is an extremely good substitute for the card function, and makes the optimization problem completely tractable. Therefore, we will be working with the below regularization function in this problem:

\[
\phi(\hat{x}) = \|D\hat{x}\|_1.
\]

Note that strictly speaking, replacing (5) with (6) is a heuristic, but it is a heuristic that works extremely well and is used almost always in practice.

To summarize, the problem of finding a piece-wise constant fit \( \hat{x} \) to a noisy signal measurement \( x_{\text{noisy}} \) can be cast into the following bi-objective optimization problem:

\[
\text{minimize } \|\hat{x} - x_{\text{noisy}}\|_2^2 + \mu \|D\hat{x}\|_1.
\]

(c) Explicitly specify the entries of matrix \( D \) used in (7) and its dimension.

(d) In this part we will use the above method for piece-wise constant data fitting to model the power consumption of different subsystems in a health monitoring device. Imagine a health monitoring implant device that consists of three subsystems: A microcontroller unit (MCU), a low energy Bluetooth unit (BLE) and a glucose sensing unit (GSU). The device is programmed to wake up every 100ms, measure the blood glucose of a patient, transmit the data to a base station over Bluetooth and go back to sleep. Hence, the power consumption for each of the three subsystems will have a periodic pattern with a period of 100ms. In addition, we know that the true power consumption for each subsystem is best modeled by a piece-wise constant signal over the course of a 100ms. In data file \textit{sys_power_model_data}, you are given \( x_{\text{mcu-noisy}}, x_{\text{ble-noisy}}, x_{\text{gsu-noisy}} \in \mathbb{R}^{1000} \), which are noisy measurements of the power consumption (in milliwatts \([\text{mW}]\)) for the three subsystems over 100ms. The data points for each signal are collected at 0.1ms intervals (i.e. a sampling rate of 10kHz). For each noisy measurement, sweep the parameter \( \mu \) in (7) and plot the optimal trade-off curve between \( \|\hat{x} - x_{\text{noisy}}\|_2 \) and \( \|D\hat{x}\|_1 \). Based on the optimal trade-off curve for each case, choose the value of \( \mu \) that results in a good piece-wise constant fit to signals \( x_{\text{mcu}}, x_{\text{ble}}, x_{\text{gsu}} \) (Note that best choice of \( \mu \) does not have to be the same for all three cases). Report your choices of \( \mu \) for the three cases.

Provide plots of (i) the noisy signal and the piece-wise constant fit to it on the same figure for each case (3 separate figures) and (ii) the optimal trade-off curve for each case (3 separate figures).
Hint: First you need to use the results of parts (a) and (b) to reformulate the bi-objective problem (7) as an iteratively reweighted least squares problem. You can choose your \( \hat{x}^{(0)} \) for each case to be equal to \( x_{\text{noisy}} \) and then solve the problem for each value of \( \mu \) in a few iterations. Generally, 5 to 10 iterations should be enough to get you decent convergence. Note that since our approach to this problem is a heuristic, in some cases it will not lead to a perfect piece-wise constant fit. Nevertheless, your results should be very close to piece-wise constant fits. Also note that when you are solving for \( w_i(x^{(k)}) \), you might encounter cases where it goes to infinity (i.e. \( w_i(x^{(k)}) = 1/0 \)). In such cases, do not update the weight and reuse the old iterate (i.e. \( w_i(x^{(k)}) = w_i(x^{(k-1)}) \)).

If your implementation is efficient, it should take only a few minutes to run your code for each case.

(e) The manufacturers of all three subsystems have provided ideal power consumption profiles for a 100ms time period. These ideal power consumption profiles are provided as \( x_{\text{ble-ideal}}, x_{\text{mcu-ideal}} \) and \( x_{\text{gsu-ideal}} \) (in mW) in data file \texttt{sys_power_model_data}. Our goal then is to scale the piece-wise power consumption patterns of the subsystems obtained in part (d) in order to match the ideal power consumption profiles as closely as possible. In addition, our overall average power consumption of the device should exactly be \( P_{\text{total}} = 25 \mu W \). To optimize the overall power consumption, you as the power engineer have the capability of scaling the power consumption for each subsystem by a constant factor (by changing the current supplied to each unit), which is effectively scaling the power consumption profile up or down. If the corresponding scaling factors for microcontroller, Bluetooth and glucose sensing units are \( \alpha \), \( \beta \), and \( \gamma \), then the overall average power consumption of the device would be given by:

\[
P_{\text{total}} = \alpha P_{\text{mcu}} + \beta P_{\text{ble}} + \gamma P_{\text{gsu}},
\]

where \( P_{\text{mcu}}, P_{\text{ble}} \) and \( P_{\text{gsu}} \) are the average power consumptions of MCU, BLE and GSU units, respectively. Find the scaling factors \( \alpha \), \( \beta \), and \( \gamma \) for each subsystem that ensure \( P_{\text{total}} = 25 \mu W \) (\textbf{Hint}: find average power consumption of one subsystem by taking average over piece-wise fit from part (d)), and make the scaled power piece-wise consumption profiles \( \alpha \hat{x}_{\text{mcu}}, \beta \hat{x}_{\text{ble}} \) and \( \gamma \hat{x}_{\text{gsu}} \) as close as possible to the ideal profiles provided by the manufacturers. For each subsystem, provide a plot of the ideal power consumption profile as well as your scaled fit on the same figure (3 separate figures).

\textbf{Partial credit:} If you were not able to finish part (d) to obtain the piece-wise constant fits \( \hat{x}_{\text{mcu}}, \hat{x}_{\text{ble}} \) and \( \hat{x}_{\text{gsu}} \), you can use the noisy power consumption profiles in \texttt{sys_power_model_data} in this part for partial credit. However, you should clearly state that you are doing so at the beginning of your solution.