MTH 373/573:
Scientific Computing

Homework 2

IIIT-Delhi, Monsoon 2022
Kaushik Kalyanaraman

Due by: October 16, 2022 (Sunday) by 23:59 hours IST

Total: 50 points

Guidelines

This assignment should be answered individually.

- You are not allowed to discuss specifics of your solutions, solution strategies, or coding strategies with any other student in this class.
- Please consult with and carefully read through IIIT-Delhi's Academic Dishonesty Policy.

Please don't be a cheater!

- You shall submit your solutions via Google Classroom – there are no printed/handwritten/-paper submissions. Please type, take a picture, or scan your solutions and upload them as a single PDF file and only a single PDF file. You can find many websites that will merge your scanned images into a PDF file or, for the nerdy types, you can find a command line script to do this.
- You may post queries relating to any HW question to `#discussions` on the class's Slack channel.
- Start working on your solutions early and don't wait until close to due date.
- Each problem should have a clear and concise write-up.
- Please clearly show all steps involved in your solution.
- You will need to write a separate code for each computational problem and sometimes for some subproblems too. You should name each such file as `problem_`*n*`.py` where *n* is the problem number. For example, your file names could be `problem_5.py`, `problem_6a.py`, `problem_6b.py`, and so on.
- *Python tip:* You can import `Python` modules as follows:

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import numpy.linalg as npla
import scipy.linalg as spla
```

- We shall only accept `*.py` `Python` files – no Jupyter Notebook submissions are allowed. Please write your code for the machine problems in a text editor and generate appropriate output via Terminal using `Python` (directly) or `IPython` (recommended).

Problem 1: Linear Least Squares Problem Formulation — 10 points

Formulate the following minimization problems as linear least squares problems. That is, for each problem, provide an appropriate matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$ such that the problem can be expressed as $Ax \simeq b \iff \min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2$.

(a) Minimize $x_1^2 + 2x_2^2 + 3x_3^2 + (x_1 - x_2 + x_3 - 1)^2 + (-x_1 - 4x_2 + 2)^2$.

(b) Minimize $(-6x_2 + 4)^2 + (-4x_1 + 3x_2 - 1)^2 + (x_1 + 8x_2 - 3)^2$.

(c) Minimize $2(-6x_2 + 4)^2 + 3(-4x_1 + 3x_2 - 1)^2 + 4(x_1 + 8x_2 - 3)^2$.

(d) Minimize $x^T x + \|Bx - d\|_2^2$ where $B \in \mathbb{R}^{p \times n}$ and $d \in \mathbb{R}^p$ are given.

(e) Minimize $x^T Dx + \|Bx - d\|_2^2$ where $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix with positive diagonal elements, $B \in \mathbb{R}^{p \times n}$ and $d \in \mathbb{R}^p$ are given.

Points breakdown: Each of the above parts is worth 2 points.

Problem 2: Least Squares Solution — 3 points

What is the solution to and Euclidean norm of the minimum residual vector for the following linear least squares problem?

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \simeq \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}.$$

Solve this problem using the necessary condition for the existence of the minimum and verify using the sufficiency condition that the solution is indeed a minimum. Exhibit all steps in your computations.

Points breakdown: 1 point for computation of solution, 1 point for verification of it being a minimum, and 1 point for computing the norm of the minimum residual vector.

Problem 3: Theoretical Problem — 6 points

Let $A \in \mathbb{R}^{m \times n}$, $m \geq n$ with linearly independent columns.

(a) Show that the $(m + n) \times (m + n)$ matrix:

$$\begin{bmatrix} I & A \\ A^T & \mathbb{O} \end{bmatrix}$$

is nonsingular.

(b) Show that the solution $\hat{x}, \hat{y}$ of the set of linear equations:

$$\begin{bmatrix} I & A \\ A^T & \mathbb{O} \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

is given by $\hat{x} = b - Ax$ and $\hat{y} = x$ where x is the solution of $Ax \simeq b$.

Points breakdown: 3 points for each part.

Problem 4: Least Squares and QR Decomposition — 3 points

Let $\hat{x}$ be the solution of the least squares problem: $\min \|b - Ax\|_2^2$, where $A \in \mathbb{R}^{m\times n}$, $m \geq n$ with linearly independent columns and $b \notin \operatorname{im} A$. Consider the following QR factorization:

$$\begin{bmatrix} A & b \end{bmatrix} = QR = \begin{bmatrix} q_1 & q_2 & \cdots & q_n & q_{n+1} \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1n} & R_{1,n+1} \\ 0 & R_{22} & \cdots & R_{2n} & R_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & R_{nn} & R_{n,n+1} \\ 0 & 0 & \cdots & 0 & R_{n+1,n+1} \end{bmatrix},$$

where each $q_i \in \mathbb{R}^m$, $i = 1, \ldots n+1$ are column vectors. Now, using this show the following norms can be computed from the elements of the last column of R:

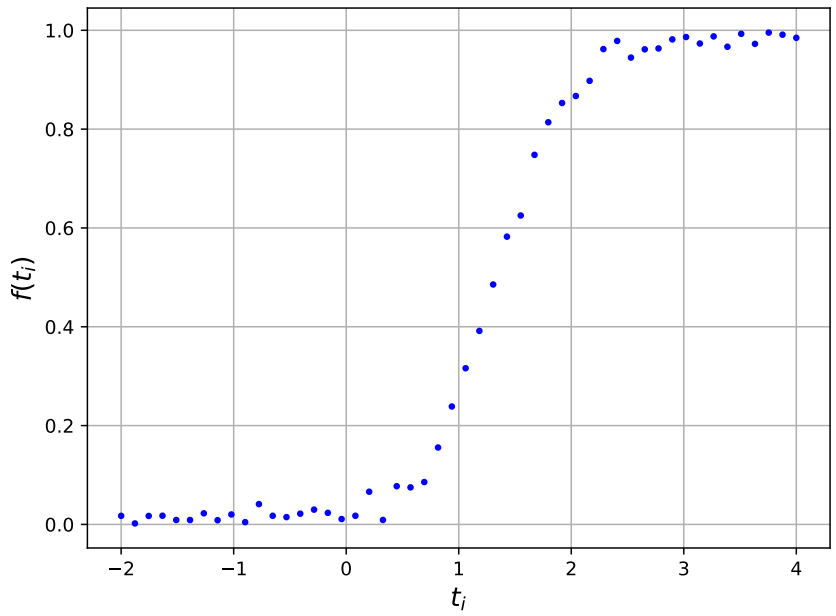
(a) $\|b\|_2$, (b) $\|A\hat{x}\|_2$, (c) $\|b - A\hat{x}\|_2$.

Points breakdown: Each of the parts is worth 1 point.

Problem 5: Linear Least Squares Data Fitting — 8 points

In this problem, you will be fitting a function to sampled data with a small twist which you will have to untangle! So, the figure below shows 50 data points (t_i, y_i), and I claim that these points are well approximated by the function:

$$y = f(t) = \frac{e^{\alpha t+\beta}}{1 + e^{\alpha t+\beta}}.$$



Your tasks for this problem are as follows.

(a) Formulate this problem as a linear least squares problem. That is, set up the problem to find values of the parameters α, β so that:

$$\frac{e^{\alpha t_i+\beta}}{1+e^{\alpha t_i+\beta}} \approx y_i, \quad i = 1, \dots m,$$

where $m = 50$. You can assume that $0 < y_i < 1$ for all $i = 1, \dots, m$.

Hint: The twist here is that this is not readily a linear least squares problem. You can convert it to one though with an appropriate use of a logarithm!

(b) Use the $\{(t_i, y_i)\}_{i=1}^{50}$ data points in the accompanying file `hw2_data_ty.txt` to set up the appropriate matrix A and right hand side vector b for this problem. Compute your solution to this linear least squares problem in two different ways: i) by solving the normal equations using `np.linalg.solve` and ii) by directly solving $Ax \simeq b$ using `np.linalg.lstsq`. (Please read the documentation for the second function to determine how to use it.)

Compute the errors for the fits (that is, the 2-norm of the residual) for each case as well as plot both fitted functions in a single plot.

Note: You can use the following code fragment to read the data into an array. I have also listed in this as to how I plotted these points so that you have some reference on how to go about with generating "reasonable" plots.

```
import numpy as np
import matplotlib.pyplot as plt

t, y = np.loadtxt("hw2_data_ty.txt").T
plt.figure()
plt.plot(t, y, 'bo', ms=2.5)
plt.grid(True)
plt.xlabel("$t_i$", fontsize=14)
plt.ylabel("$f(t_i)$", fontsize=14)
plt.gcf().tight_layout()
plt.show()
```

Note: You will need to provide a written/typeset explanation in part (a) for the problem formulation. In part (b), you will submit your Python code (appropriately in a filename `problem_n.py`) along with reporting the errors and providing your plots in the writeup.

Points breakdown: Part (a) is worth 3 points and part (b) is worth 5 points.

Problem 6: Normal Equations versus QR Factorization — 10 points

In this problem we compare the accuracy of the two methods for solving a least squares problem $Ax \simeq b$. Let us take the following to be A and b:

$$A = \begin{bmatrix} 1 & 1 \\ 10^{-k} & 0 \\ 0 & 10^{-k} \end{bmatrix}, \qquad b = \begin{bmatrix} -10^{-k} \\ 1 + 10^{-k} \\ 1 - 10^{-k} \end{bmatrix}.$$

(a) Write the normal equations for this problem, and solve the resulting 2×2 linear system analytically. That is, you will solve this for *any* k.

(b) Solve the linear least squares problem in for $k = 6, 7, \ldots, 15$ using a QR factorization of A (`np.linalg.qr`) and using the triangular solver available in `scipy.linalg.solve_triangular`. Read the two functions' documentaion to see how to use them by going to the NumPy and SciPy websites as shown in an earlier problem.

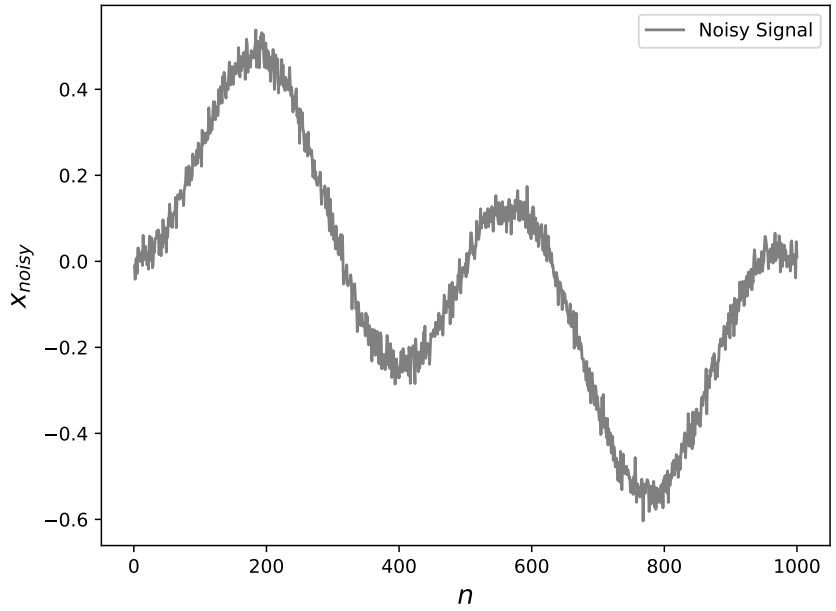
(c) Solve the linear least squares problem in for $k = 6, 7, \ldots, 15$ using normal equations and `np.linalg.solve`. Compare the solution obtained for each k in this case with the analytical solution and that obtained via a QR decomposition solution.

Points breakdown: Part (a) is worth 2 points and parts (b) and (c) are worth 4 points each.

Problem 7: Denoising using Linear Least Squares — 10 points

In this problem, we shall look at an application of linear least squares, namely, denoising a signal. However, our application is what can be termed "model problem" in that this is a model for real-life problems which you may encounter (hopefully!) during the course of your academic or professional career. The wishful thinking is that having seen something like this will then help you solve a future problem or at least understand some black-box denoiser from some other package or software! Like this prelude, this problem description is lengthy but the question that I am asking you to work on itself is rather straightforward – it just looks daunting!

The figure below shows a signal of length 1000, corrupted with *noise*. You will estimate the original signal. This is referred to as signal reconstruction, denoising, or smoothing. In this problem, we shall assume that the noise in the signal is the small and rapidly varying. We shall therefore apply a smoothing method based on least squares



We will represent the corrupted signal as a vector x_{noisy} of size 1000. The estimated signal, that is, the solution to our problem will be represented as a vector x of size 1000.

To reconstruct the signal, we decompose x_{noisy} into two parts $x_{\text{noisy}} = x + n$ where n is small and rapidly varying, and x is close to x_{noisy}: $x \approx x_{\text{noisy}}$, and slowly varying: $x_{i+1} \approx x_i$. We can achieve such a decomposition by choosing x as the solution of the following least squares problem:

$$\min \left\| x - x_{\text{noisy}} \right\|_2^2 + \lambda \sum_{i=1}^{999} \left(x_{i+1} - x_i \right)^2 ,$$

where λ is a positive constant. In the above minimization problem, the first term $\left\| x - x_{\text{noisy}} \right\|_2^2$ measures how much x deviates from x_{noisy}. The second term $\sum_{i=1}^{999} \left(x_{i+1} - x_i \right)^2$ penalizes rapid changes of the signal between two samples. By minimizing a weighted sum of both terms, we obtain an estimate x that is close to x_{noisy} and also varies slowly. The parameter λ is used to adjust the relative weight of both terms (and this is a strategy that is widely used in many optimization problems including in machine learning).

Although it is not readily evident, the above minimization problem is a linear least squares one because it can be expressed as $Ax \simeq b$ where:

$$A = \begin{bmatrix} I \\ \sqrt{\lambda} D \end{bmatrix}, \qquad \begin{bmatrix} x_{\text{noisy}} \\ 0 \end{bmatrix},$$

where D is a 999×1000 matrix defined as:

$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & \cdots & 0 & 0 \\ \vdots & & \ddots & \ddots & & & \vdots \\ 0 & 0 & \cdots & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & -1 & 1 \end{bmatrix}_{999 \times 1000}.$$

The matrix A is thus 1999×1000. We shall finally solve the linear least squares problem by solving the normal equations:

$$\left(I + \lambda D^T D\right) x = x_{\text{noisy}}.$$

Solve the linear least squares problem with the vector x_{noisy} defined in `hw2_data_denoising.txt` for $\lambda = 1, 100, 10000$. Plot the three reconstructed signals x and provide your observations on the effect of λ on the quality of the estimate x.

Note: Here is my code fragment to read and plot the noisy signal.

```
import numpy as np
import matplotlib.pyplot as plt

x_noisy = np.loadtxt("hw2_data_denoising.txt")
plt.figure()
plt.plot(np.arange(1, 1 + len(x_noisy)), x_noisy,
           color=(0.5, 0.5, 0.5), label="Noisy Signal")
plt.xlabel("$n$", fontsize=14)
plt.ylabel("$x_{noisy}$", fontsize=14)
plt.legend(loc="best")
plt.gcf().tight_layout()
plt.show()
```