ORF 363/COS 323	Problem set 5	Fall 2024, Princeton University
Instructor: A.A. Ahmadi		
TAs: Aydin, Budway, Hua, Tziampazis		
Due on Thursday, November 7, 2024, at 1:30pm EST, on Gradescope		

For all problems that involve coding, please include your code.

Problem 1: Theory-applications split in a course. (Courtesy of Stephen Boyd)
A professor teaches a course with 24 lectures, labeled $i = 1, \ldots, 24$. The course involves some interesting theoretical topics, and many practical applications of the theory. The professor must decide how to split each lecture between theory and applications. Let T_i and A_i denote the fraction of the ith lecture devoted to theory and applications, for $i = 1, \ldots, 24$. (We have $T_i \geq 0$, $A_i \geq 0$, and $T_i + A_i = 1$.)
A certain amount of theory has to be covered before the applications can be taught. We model this in a crude way as

$$A_1 + \cdots + A_i \leq \phi(T_1 + \cdots + T_i), \quad i = 1, \ldots, 24, \tag{1}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is a given nondecreasing function. We interpret $\phi(u)$ as the cumulative amount of applications that can be covered, when the cumulative amount of theory covered is u. We will use the simple form $\phi(u) = a \max\{0, u - b\}$ with $a, b > 0$, which means that no applications can be covered until b lectures of the theory is covered; after that, each lecture of theory covered opens the possibility of covering a lectures on applications.
The theory-applications split affects the emotional state of students differently. We let s_i denote the emotional state of a student after lecture i, with $s_i = 0$ meaning neutral, $s_i > 0$ meaning happy, and $s_i < 0$ meaning unhappy. Careful studies have shown that s_i evolves via a linear recursion (dynamics)

$$s_i = (1 - \theta)s_{i-1} + \theta(\alpha T_i + \beta A_i), \quad i = 1, \ldots, 24,$$

with $s_0 = 0$. Here α and β are parameters (naturally interpreted as how much the student likes or dislikes theory and applications, respectively), and $\theta \in [0, 1]$ gives the emotional volatility of the student (i.e., how quickly he or she reacts to the content of recent lectures). The student's *cumulative emotional state* (CES) is by definition $s_1 + \cdots + s_{24}$. This is a measure of his/her overall happiness throughout the semester.
Now consider a specific instance of the problem, with course material parameters $a = 2$, $b = 3$, and three groups of students, with emotional dynamics parameters given as follows:

	Group 1	Group 2	Group 3
θ	0.05	0.1	0.3
α	-0.1	0.8	-0.3
β	1.4	-0.3	0.7

Your job is to plan (four different) theory-applications splits that respectively maximize the CES of the first group, the CES of the second group, the CES of the third group, and, finally, the minimum of the cumulative emotional states of all three groups. (Hint: you would need to appropriately reformulate constraint (1) to end up with a convex optimization problem.) Report the numerical values of the CES for each group, for each of the four theory-applications splits (i.e., fill out the following table):

	Group 1	Group 2	Group 3
Plan 1			
Plan 2			
Plan 3			
Plan 4			

For each of the four plans, plot T_i as well as the emotional state s_i for all three groups, versus i. (So you should have four figures with four curves on each.) These plots show you how the emotional states of the students change as the amount of theory varies.

Problem 2: Estimating $\sqrt{20}$.

You are stuck in a place with no electronic devices allowed and asked to compute $\sqrt{20}$. How can you use Newton's method for root finding to approximate $\sqrt{20}$ using only the operations $+, -, \times, \div$? Once you write down the right function, apply two iterations of Newton's method for root finding starting from $x_0 = 5$ by hand. Report your final estimate of $\sqrt{20}$ as a rational number.

Now you get home and can get your hands on your computer. Run Newton's method for root finding starting at $x_0 = 5$ on your machine. How many iterations does it take to get $\sqrt{20}$ correct to 15 digits after the decimal point?

Problem 3: Newton fractals

The sensitivity of Newton's method to initial conditions is beautifully demonstrated using plots over the complex plane known as *Newton fractals*. You may have seen a picture of Newton fractals in lecture notes, and now your task in this problem is to produce the Newton fractal associated with the critical points of $f(z) = z^5 - 5z$. The steps below are only meant to help you do this — there is no grade assigned to them.

1. Note that z is a complex number throughout this exercise. Verify that the critical points of f, i.e., the roots of f' are $z_1 = 1, z_2 = -1, z_3 = i, z_4 = -i$.

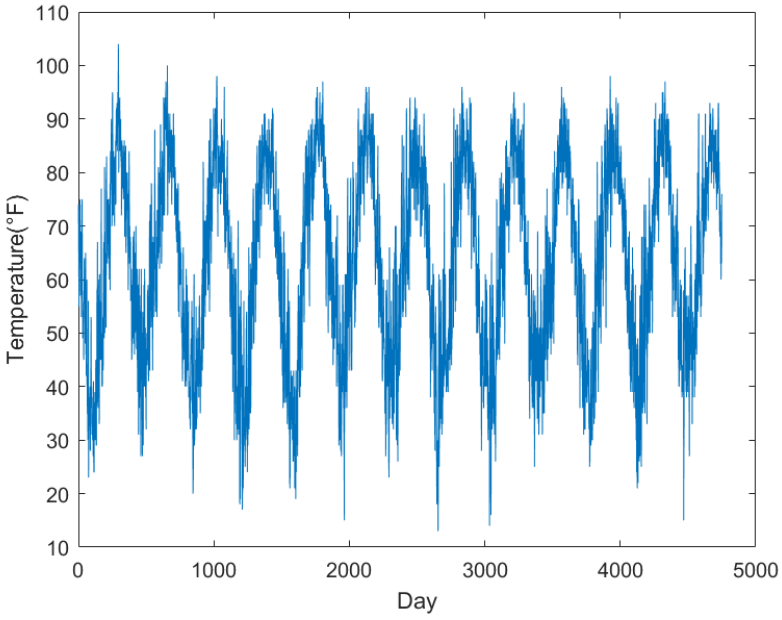
2. Discretize $[-1, 1] \times [-1, 1]$ using intervals of length 0.0031. We recommend that you define the sequence of points $x = -1 : 0.0031 : 0.999$, and $y = -0.999 : 0.0031 : 1$ to avoid certain numerical issues. For each point (x_j, y_l) in your discrete grid, apply Newton's method with $z_{jl} = x_j + iy_l$ as its initial point.
   Hint: Consider using the `meshgrid` function to create your grid (which will contain 645^2 points on the complex plane). To run the Newton method, we recommend using matrix operations in MATLAB instead of for-loops. Finally, you may set the maximum number of iterations for Newton's method to 200 for simplicity.

3. Map each of the critical points of f to some color code; e.g., $z_1 \leftrightarrow 1, z_2 \leftrightarrow 2, z_3 \leftrightarrow 3, z_4 \leftrightarrow 4$. Then, to each initial condition (i.e., to each (x_j, y_l) on the grid) assign one of the four color codes based on the root that the iterations are converging to (up to some tolerance error, say, $\epsilon = 0.01$). Depending on how you discretize, for some initial conditions the algorithm may not converge. In that case, assign color code 0 to that particular (x_j, y_l). You will obtain a 645×645 matrix of color codes representing your Newton fractal. Plot it using the `imagesc` function in MATLAB or the `imshow` function in Python (matplotlib).

Submit a print out of your code and plot. To get credit, your code must produce the plot.

Problem 4: Orbit of the Earth around the Sun and Temperatures in NYC

In the file `TemperatureNewYork.csv`, you are given the maximum daily temperatures in Central Park, New York over 13 years (starting on 10/01/2010).[1] To load the data, you can use the function `readmatrix` in Matlab or `np.loadtxt` in Python. This is what the data looks like:



The goal of the question is to estimate the duration of the Earth's orbit around the Sun, using only temperature data in New York City. To do this, we wish to fit the following model to the data

$$T(t) = a \sin\left(\frac{2\pi}{T_E} t + \Phi\right) + bt + c,$$

where $T(t)$ is the temperature on day t, and our parameters to be found are a, b, c, Φ and T_E. The last parameter T_E corresponds to the length of the Earth's orbit in days. Essentially, we are postulating that the temperature is a periodic function of time, with a given phase and period, to which we have added an offset and growth term. We saw in class that the problem of fitting such a model to data is a nonlinear least-squares problem, for which the Gauss-Newton method is a popular algorithm.

Write down the nonlinear least squares problem that minimizes the sum of the squares of the residuals between the data and the prediction from the model over the 13 years. Apply 50 iterations of the Gauss-Newton method to the problem starting with the initial condition

[1] This data was obtained from `http://www.ncdc.noaa.gov/cdo-web/`.

$(a, T_E, \Phi, b, c) = (120, 360, \pi, 0, 0)$.[2] Plot the data we gave you and the best fit that you found on the same figure.

Hint: You may want to use *function handles* in Matlab for the implementation of the Gauss-Newton algorithm. These are used to create and evaluate functions in a simple and efficient manner. Here is an example:

```
>>f=@(x,y,z) x^2+2*y+1/z;
>>f(1,1,1)
ans=
4
```

In Python, you can use *lambda functions* to create function handles, but it is totally fine if you write normal functions. Here is an example:

```
f = lambda x, y, z: x**2 + 2*y + 1/z
```

(a) What is the period T_E^* that you obtain for the revolution of the Earth around the Sun? How far is it from the true value? (You can use Wikipedia e.g. to get this number.)

(b) What do this particular model and dataset tell you about global warming?[3]

[2] As the problem is nonconvex, the success of the algorithm is highly sensitive to the way the initial conditions are chosen. This is why we start the algorithm with a reasonable guess for the values of our parameters. Feel free to play around with these initial conditions.

[3] You should be careful not to form serious opinions about an important issue like global warming based on one limited experiment.