Phys 256 — Homework 2

Student Name

Instructions

- Submit written report in PDF format (with inlined code snippets, if it is needed or required). Report must be submitted to Gradescope 'hw02report' assignment.
- Submit Python code to Gradescope 'hw02code' assignment.
- When required, all relevant code for a given problem should be in file named as `pN.py` where 'N' stands for problem number, e.g. `p1.py`, `p2.py`, or `p10.py`. All code files should be contained in one 'zip' archive and uploaded for grading to Gradescope.
- The class web page will have templates with function definitions to which you must adhere!
- Do not forget to discuss test cases in your report.

Prerequisites

Read help about `plt.plot` from `matplotlib` package, `np.linspace` and `np.arange` from `numpy` package, and about the built-in `print` command.

Problem 1: (2 points)

Plot the function $f(x) = \exp(-x^2/10) * \sin(x)$ for 400 linearly spaced points of x in the region from 0 to 2π. Points should be joined with solid lines.

Do not use any loops (for, while). Inline the code listing and include the resulting figure in your report.

No additional code required for separate submission.

Problem 2: (2 points)

Plot the functions x^2 and $x^3/2 + 0.5$ for 100 linearly spaced points of x in the region from -1 to $+1$. x^2 should be a red solid line and the other function should be a black dashed line.

Do not use any loops (for, while). Include the resulting figure and inline code listing in the body of your report.

No additional code required for separate submission.

Problem 3: (3 points)

Write a function that calculates the sum below

$$S_N = \sum_{k=1}^{N} a_k \tag{1}$$

for any integer $N < 10^6$ where $a_k = 1/k^{2k}$ for odd k and $a_k = 1/k^{3k}$ for even k. The function should return 0 if $N < 1$.

Use a `while` loop for the function implementation. Is it better to start summing from high values $k = N$ downward toward $k = 1$? Or is it better to do the conventional sum starting from $k = 1$ and going upwards to $k = N$? Justify your answer. Implement the way which gives the most precise numerical answer.

Hint: you may find `np.mod` function useful to check for even and odd numbers.

Problem 4: (3 points)

Write a function that calculates

$$1 + \sum_{i=1}^{N} \frac{1}{x^i} \tag{2}$$

for any integer $N < 10^6$ and floating-point number x. The function should return 0 if $N < 1$. The function should return `np.inf` if $x = 0$.

Do not use loops for this problem. Instead use `np.array` and relevant operations with it to obtain the final result. Use `np.sum` for the final summation. Note: unfortunately we have no control how summation is done by the `numpy` package, so we cannot do the same trick as above.

Problem 5: (5 points)

Code file submission is required for this part.

Write a function `mycos(x, N)` that calculates the value of $\cos(x)$ at the given point x via the Taylor series up to N terms. x could be any number from -20π to 20π.

Only the figure and inlined code is needed for the part below.

How well does your code handle the situation with large x values? Take $x = 10\pi$ for example.

Once you have your `mycos(x, N)` function, make a plot of its error vs the stock `np.cos(x)` function for $x = \pi/4$ at different values of N as large as 10000. I.e. plot `np.abs ( mycos( x, N)- np.cos(x))` for the relevant set of N. I **strongly** suggest to use `plt.loglog` for this case. Does the error decrease as N grows or is there a strange saturation? Explain your observations.

How far do you need to expand the Taylor series to get absolute precision of 10^{-4}? What value of N do you find reasonable (no need to state it beyond one significant digit). Why is this so?

Problem 6: (5 points)

Download the data file "hw02dataset.csv" from the class webpage. It represents the result of someone's attempt to find the resistance of a sample via measuring a voltage drop (V) (1st column) and current (I) passing through the resistor at the same conditions (2nd column). Judging by the number of samples it was an automated measurement.

- Using Ohms law $R = V/I$ find the resistance (R) of this sample (no need to print it out for each point at this step).

- Estimate the resistance of the sample (i.e. find the average resistance) and estimate the error bars of this estimate (i.e. find the standard deviation). Do not make the fitting with straight line (this is next week's material).

To perform these tasks you must write functions in your code submission to calculate the sample mean (`mymean(x)`) and the sample standard deviation (`mystd(x)`) that take x as `np.array`.

For post-processing and plotting it is sufficient to have inlined code listings included in your report.

As a reminder, the standard deviation is defined as

$$\sigma(x) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{3}$$

where x is the set (vector) of data points, $\bar{x}$ is the average (mean), and N is the number of points in the set.

Additional requirements:

1. **Do not use** standard built-in `np.mean` and `np.std` functions in your solution. You need to make your own code to calculate them. But feel free to test against `python numpy` functions.

2. Your implementation of functions should be general i.e. they should be able to work with arrays of any length bigger than one. I.e. do not make it specific to the provided data file.

3. If the standard deviation function receives an array with only one data point it must return `np.nan`. The `len` function is your friend to find the number of elements in the array.

Note: read help for `np.std`, it has many options, and you might want to know about them. In particular, the default value of `ddof` option is not what you want it to be in order to check your work.