homework5

February 11, 2024

Introduction to Computer Programming for the Physical Sciences

Joseph F. Hennawi

Winter 2024

Open a new Jupyter notebook

Name your notebook with your name and Homework 1

Open a Markdown cell at the top and write your name and Homework 1

Open a Markdown cell before each problem and write e.g. Problem 1, Problem 2(a), etc.

Please abide by the Policy and Guidelines on Using AI Tools

Once you finish the problems: 1) Restart the Python kernel and clear all cell outputs. 2) Rerun the notebook from start to finish so that all answers/outputs show up. 3) Save your notebook as a single .pdf file and upload it to Gradescope on Canvas by the deadline. No late homeworks will be accepted except for illness accompanied by a doctor's note.

For parts of problems that require analytical solutions you can perform your calculations using a pencil and paper. Then photograph your work and convert the photograph to a .pdf file using an online tool. Homework assignments can only be submitted as a single .pdf file, so you will also need to figure out how to concatenate your photo .pdf file and your notebook .pdf file into a single .pdf file that you can submit. Online websites can do this for you. Alternatively, you can code up the analytical solution to your problems in a notebook Markdown cell using the LaTeX mathematical rendering language. This is harder but a chatbot can help you learn it.

1 Homework 5

1.1 Problem 1: Riemann Sums

In an upcoming lecture we will learn how to numerically integrate functions in Python. You are all familiar with the concept that an integral

$$I = \int_a^b f(x)dx$$

is simply the area under the curve $f(x)$ between $x = a$ and $x = b$. You are probably also familiar with the definition of a Riemann integral that approximates this area as a finite sum of vertical rectangular strips, and then taking the limit as the width of the strips goes to zero (you can read

more at the Wikipedia article on Riemann sums). If we use n strips of equal width $\Delta x = (b-a)/n$, then the Riemann integral can be defined as

$$I \approx \sum_{i=1}^{n} f(a + i\Delta x/2)\Delta x$$

where $f(a + i\Delta x/2)$ is the value of the function $f(x)$ at the midpoint of the i th strip. In other words, we are using the so-called **midpoint** rule whereby one evaluates the function, $f(x)$, at the midpoint of each interval, which determines the height of each rectangle.

a) Write a function to evaluate Riemann sums that is consistent with the docstring below. **No loops are allowed anywhere in your implementation!**

```
def riemann_sum(func, a, b, n=15, plot=True):
    """
    Compute the Riemann sum of func(x) over the interval [a, b] with n subintervals using the

    Parameters
    ----------
    func : callable
        The function to integrate, with calling sequence func(x).
    a : float
        The lower limit of integration.
    b : float
        The upper limit of integration.
    n : int, optional
        The number of subintervals to use. Default is 15.
    plot : bool, optional
        If True, plot the function func(x) over the interval [a, b] (using a finely spaced arr
        used to compute the Riemann sum and the midpoints.  Default is True.

    Returns
    -------
    answer : float
        The estimate of the integral of func(x) over the interval [a, b].

    """
```

b) Use your `riemann_sum` function to numerically evaluate the following two definite integrals:

$$\int_0^{\pi} \sin(x)dx = 2 \qquad \text{and} \qquad \int_{-1}^{2} x^3 dx = \frac{15}{4}.$$

for the default value of `n=15`. For each integral, print the numerical result, the exact result, and the relative error between the two the screen. Run your function with `plot=True` for both cases. An example of how the plot should look for the case of $\int_{-1}^{1} x^2 dx = \frac{2}{3}$ is shown below. Make sure your plot looks the same!

c) Obviously the accuracy of the Riemann sum depends on the number of subintervals chosen n. One can show that the *absolute error*, $\epsilon_{\text{abs}}(n)$, of an integral approximated with Riemann sums

scales like:

$$\epsilon_{\rm abs}(n) \equiv \left| I(n) - \int_a^b f(x)dx \right| \propto n^\alpha$$

where α is a constant. For the case of $\int_0^\pi \sin(x)dx = 2$, use your function to compute the Rieman sum for $n = 10, 100, 1000, 10000, 100000$ and store your results in a `numpy` array or a list. I strongly recommend that you set `plot=False` in your calls to `riemann_sum` for this step! Make a **scatter** plot (i.e. the data should be shown as points, not a curve!) of the absolute error $\epsilon_{\rm abs}(n)$ as a function of n with logarithmic x and y axes. From your plot of the data, estimate the value of α and overplot a curve with $g(n) = An^\alpha$, where the constant A is chosen to roughly go through your data points. Make sure you label your coordinate axes and include a legend. You are allowed to use a loop over n for this step.

1.2 Problem 2: Blackbody Radiation

A *blackbody* is a physics idealization of a body in which electromagnetic radiation has come into perfect thermal equilibrium with the body, both of which can therefore be characterized by a temperature T, assumed to be measured on the absolute (Kelvin) scale. Your own body radiates mostly infrared photons that are approximately blackbody in nature, and the Sun radiates mostly optical photons that are also approximately blackbody in nature. The frequency spectrum of the radiation emitted by a blackbody of temperature T is given by

$$F(\nu) = \frac{2\pi h\nu}{c^2} \frac{1}{\exp(h\nu/k_{\rm B}T) - 1}$$

where h is Planck's constant, ν is the photon frequency, c is the speed of light, and $k_{\rm B}$ is Boltzmann's constant. The quantity $F(\nu)d\nu$ is the radiation flux (power per unit area) emitted within the frequency range ν to $\nu + d\nu$. Therefore $F(\nu)$ has SI units of $\mathrm{W/m^2/Hz}$.

a) By Taylor expanding the exponential term in the denominator of the above expression, show that in the low frequency limit $h\nu \ll k_{\rm B}T$ the radiation flux has a power-law dependendce on frequency. What is that power-law index? This is known as the *Rayleigh-Jeans* spectrum, and is what classical physics predicted for the spectrum before the advent of quantum mechanics. You should see that Planck's constant does not appear in your answer (hence there is no quantum physics in this description).

b) Write a Python function that takes as input the temperature T and a frequency ν and returns the radiation flux $F(\nu)$. Given the large dynamic range of frequency, ν, that you will need to visualize in part **c)**, you should evaluate this function using a `numpy` array of **logarithmically spaced** ν values.

c) Plot the spectrum (i.e. $F(\nu)$ vs. ν) for the case of $T = 310.15\,\mathrm{K}$ (human body temperature) and $T = 5778\,\mathrm{K}$ (Sun's surface temperature) *on the same plot.* Your plot should have the following set of characteristics: - Different colors for the curves illustrating the two different temperatures. - Labels on the x and y axes (specifying units). - A legend indicating the two different curves. - Clearly illustrate the behavior at low and high frequencies. This requires a sufficiently large range of frequencies and logarithmic axes for both the x and y axes. - A secondary x-axis at the top of the plot that shows the wavelength $\lambda = c/\nu$ in microns, μm. Recall that $1\mu\mathrm{m} = 10^{-6}\mathrm{m}$. - A reference curve shown as a black dashed-line that shows the Rayleigh-Jeans power-law behavior that you derived for the low frequency limit in part **a)**. Assume the Sun's surface temperature, $T = 5778\,\mathrm{K}$, for plotting this Rayleigh-Jeans power-law curve.

d) Is the low-frequency behavior of the spectrum consistent with the Rayleigh-Jeans power-law you derived in part **a)**? Does the human body spectral flux ever exceed that of the Sun at any frequency?

2 Problem 3: Distribution of Household Size and Income in the US

The American Community Survey (ACS) is a yearly survey conducted by the United States Census Bureau that collects information about the country and its inhabitants. Download the file census_income_data.csv at this link: https://github.com/enigma-igm/Phys29/blob/main/Phys29/homework/HW4/data/census_income_data2022.csv This is a comma-separated values or **csv** file containing household income data from the ACS.

As you can see from the header, it has three columns, `WGTP`, `NPF`, and `HINCP`: - `WGTP` is the weight of the household, or the number of such households in the US population - `NPF` is the number of people in the household - `HINCP` is the household income

The `WGTP` is a statistical weight that is assigned to each record (line of the file) that indicates how many people in the population are represented by that record. In yearly surveys like the ACS, it's not feasible to collect data from every individual in the population, so a subset of the population is surveyed instead. Each record in the datafile thus represents (or "stands in for") a certain number of similar individuals in the population. The WGTP value tells you how many individuals that record represents.

For example a line in the file like:

```
WGTP, NPF, HINCP
127, 5, 75000
```

indicates that in the USA there are 127 households with 5 members per household in them which earn a yearly household income of $75,000.

In this problem we want to analyze the distribution of household size and household income in the U.S. population. Given the way that the data is collected, we need to use the WGTP values to get accurate estimates of the distribution across the population. For example, if you want to calculate the total number of households:

$$N_{\rm households} = \sum_i {\rm WGTP}_i$$

where the sum is over all records in the datafile.

Similarly, to calculate the total population of the USA, you would sum up the product of the WGTP (# of similar housholds in USA) and NPF (# of family members per household) values:

$$N_{\rm people} = \sum_i {\rm WGTP}_i \times {\rm NPF}_i.$$

a) Read in the datafile into a `pandas` DataFrame `df`. Run the `df.describe()` method on the DataFrame to get a sense of the values in the data.

b) Assign the `WGTP`, `NPF`, and `HINCP` columns to `numpy` arrays. Calculate the total number of households and the total number of people in the US population using the formulas above. Print the results to the screen. Do the numbers you get make sense?

c) According to your result from `df.describe()`, the household income column, `HINCP`, contains negative values. These correspond to households that had a net loss in 2022. For example, suppose a household member owns a business that suffered a loss in 2022 (the business expenses exceeded business income), this would corresopnd to a net negative income. Use `numpy` boolean array indexing to create a new filtered dataset that only includes the records with a net positive household income, i.e. `HINCP > 0`. Recompute the total number of households and the total number of people in the US population using this filtered dataset and print the results to the screen. Use this filtered dataset for all the remaining parts of this problem.

d) In lecture we learned how `matplotlib` histograms can be used to visualize the distribution in a dataset. Use the `matplotlib` library to create a histogram of the number of persons per household, `NPF` in the dataset. Choose linearly spaced bins of unit width from the minimum value of the `NPF` to the maximum value (make sure your bins include the minimum and maximum value). The y-axis of your histogram **should be the percentage of the total number of housholds** contained in each bin, where the total number of households is the number you calculated in part **c)**. Note that you cannot simply run `plt.hist(NPF, bins=bins)` or `np.histogram(NPF, bins=bins)` because the `WGTP` values need to be taken into account. The `plt.hist` and `np.histogram` functions have a `weights` argument that you can use to construct weighted histogram and thus factor in the weights. Use the arrays returned by the histogram function to explicitly verify that the sum of the percentages over all the bins is equal to 100%, and print this to the screen.

e) Create and plot a histogram of the household income, `HINCP` in the dataset, again taking proper account of the `WGTP` weights. Choose linearly spaced bins of width $\$20,000$ spanning from \$0 to $\$2,500,000$. As in part **d)**, the y-axis should be the percentage of the total number of housholds contained in each bin. Choose a linear scaling for both the x-axis and y-axis. Again, verify that the sum of the percentages over all the bins is equal to 100%, and print this to the screen.

f) Remake the plot from part **e)** but using a logarithmic scale for the y-axis. Which scale do you think is more informative, linear or log?

g) For the household income histogram, write code that uses the arrays returned to print to the screen the bin index (zero based), the left edge of the bin, the right edge of the bin, and the percentage (up to six decimal places) of the total number of households contained in each bin.

f) From the `numpy` array that contains the household income histogram output (i.e. what `np.histogram` or `plt.hist` return), compute the **total percentage** of households with an income **greater than or equal to** $\$500,000$ and print this to the screen.

g) From the `numpy` arrays that contain the US census data (i.e. `WGPT`, `HINCP` etc.), directly compute the the **total percentage** of households with an income **greater than or equal to** $\$500,000$ and print this to the screen. **Note that you are not allowed to use your histogram outputs for this step**, i.e. your code needs to operate on the census data `numpy` arrays directly. Verify that that the two percentages you computed in parts **f)** and **g)** are in agreement with each other.