HW1

November 18, 2024

1 Homework 1

1.1 Deadline Nov 25, 1000 hours

1.1.1 Instructions

- It is OK to discuss among peers, but it is **not** OK to copy each other's work or each other's functions.
- The entire work should be done on a Jupyter notebook. Upload the notebook into the assignment submission section of the class moodle. You will need to use markdown cells for typing your answers and code cells to define functions and their run results.
- In addition to the PDF of the HW, I have also uploaded the notebook for you to take a look at markdown formatting in case you are not sure how to do it.
- Create a private git repository on github. Name your git repository with the following convention for better understanding. Share your git repository with the TAs. This is for practice, but the TAs will keep track of who has done it properly before the deadline. There will be some additional credits for successfully setting up the git repository and adding collaborators.
  - YOUR_FULL_NAME_numerical_methods_2024
  - Within this git repository, create a *.ipynb* file with the name YOUR_FULL_NAME_HW1_solution.ipynb

Q. 1. (4 marks) Consider that you want to build a high-performance CPU cluster with a peak performance of 100 Tera FLOPS. You decide to buy the Intel Xeon Platinum 8168 processors for your computing solution. Find the total number of connected processors you need to achieve the desired peak performance in you cluster. Clearly show and describe your calculations. You may need to search for the performance metrics for the processor. Include the source links for this information.

Q. 2. (2+3 marks) I want to print the table of 13. I have written the following code.

```
a = [i for i in range(1,11)]
b = a*13
for i in range(1,11):
    print (i, "x 13", "=", b[i])
```

- Will it properly work?
- If not, fix it and write the corrected code.

Q. 3. (5+3+4+3 marks)

1. Write a code that reads the file **HW1_data.txt** and stores the data using *with open*.
2. Make a scatterplot for BH mass vs Companion mass in Solar mass. Pay attention to axis labels, legend, and an appropriate scale for the axes.
3. Create a histogram showing the probability distribution function of orbital periods. Pay attention to axis, units, labels, legend, linestyle, etc.
4. Make a scatterplot showing age and eccentricity.

Do not use external modules such as pandas or numpy. Only use matplotlib.pyplot and native Python routines.

Q. 4. (5+10+5 marks)

1. Write a Python function that can convert a decimal integer number given by the user into a binary.
2. Write a Python function that can convert a real decimal number given by the user into its machine floating point representation by returning s, e, and f. Pay attention to the special cases.
3. Write a Python function that can convert a machine floating point representation given by the user into a real decimal number.

Q. 5. (3+2 marks)

1. What is the smallest possible real non-zero positive floating point number in double precision?
2. How is this represented in IEEE754 system in your machine?

Q.6. (5+8+2+10+2+2+10+10+2 marks) Here we will write a code that can ccalculate the exponential function and analyse the various aspects of it. 1. Write a function that can calculate $n!$ for any user-given integer n.

2. Write a function that calculates the exponential using the appropriate Taylor series by brute force. The compulsory variable to be given to the function is the power x of e you want to calculate. As a criteria for termination, use $\sum_0^n \frac{x^n}{n!} == \sum_0^n \frac{x^{n-1}}{(n-1)!}$. I.e., stop when adding further terms does not change the answer.

3. Use the *time* module to estimate the time taken by your code to calculate $x = 0.1$ and $x = 20$ (i.e., $e^{0.1}$ and e^{20}).

4. Notice that there is no reason to actually calculate $\frac{x^n}{n!}$ and sum them. You can simply use the recurrence relation that the nth term $X_n = \frac{x}{n} \times X_{n-1}$. Write another function that exploits this recurrence relation using the same termination criteria. Write this code in a way such that in addition to the final answer, you also can get the individual terms in the series and the cumulative sum for each step as you proceed along the series.

5. Estimate the time taken by the function written for *Q.6.4* to calculate e^x for $x = 0.1$ and $x = 20$. Compare these results with the corresponding times taken by the function written for *Q.6.2*.

6. Use the *numpy* exponential function to find the results of e^x for $x = 0.1$ and $x = 20$. At which decimal place do you have the discrepancy in each case?
7. Plot the individual terms in the series X_n as a function of n for e^{20}. It should increase initially, then decrease. Roughly at which n do you get a turning point?
8. Evaluate e^x for $x = -20$ using the function written for *Q.6.4*. Compare the result with that given using the *numpy* exponential function for $x = -20$. How accurate is your code assuming *numpy* function is correct? Why is it performing worse compared to your calculation for $x = 20$?
9. Is there a better way to use your function to evaluate e^x for $x = -20$?