CHEN 2450

HOMEWORK 2

ERROR ANALYSIS

Submit all your homework using Jupyter notebooks. From this homework set onwards, you are expected to include proper text and discussion for each and every problem including appropriate headings and formatting. You will lose points if your reports are not readable or just include code with a few print out statements. No exceptions.

Problem 2 (35 pts)

We learned in class that approximate relative errors (ε_a) can be used in computing Taylor series to get an idea of how big the error is. For example, you can use this idea to keep adding terms to the Taylor series until the ε_a is less than 1%. There is another way to set a desired accuracy by specifying how many significant digits one wants in a calculation. A significant digit is a digit in a number that we know with very high certainty - the more significant digits a number has, the more certainty we have in that number. One can guarantee at least m significant digits if the approximate relative error is: $\varepsilon_a \leq (0.5 \times 10^{2-m})\%$. For example, if you'd like 4 significant digits, your error must be less than $0.5 \times 10^{-2}\%$.

1. (35 pts) Write a Python script that computes the natural logarithm function using a Taylor series expansion up to a desired number of significant digits. Your code should keep adding terms in the Taylor series until the desired number of significant digits is achieved. The number of significant digits is to be specified by the user. Recall that the Taylor series expansion for the natural logarithm function is

$$\ln(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n} = -x - \frac{x^2}{2} - \frac{x^3}{3} - \cdots \text{ for } -1 < x < 1 \tag{1}$$

   Test your script to compute $\ln(1.1)$ for 1, 2,4,6, and 12 significant digits and report the number of terms required to achieve those accuracies.

Here's a python script to compute the Taylor series for the exponential function using a for loop. You should be able to adapt this script to compute the natural logarithm:

```
from math import factorial
oldval = 0.0
x = 2.0
nterms = 10
for n in range(0,nterms):
```

```
newval = oldval + x**n/ factorial(n)
ea = abs( (newval - oldval)/newval) * 100
oldval = newval
print(n, ea,'%')
```