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 1 (30 pts)

To represent a floating point real number on a computer, it must first be converted to binary form. For example, the number 0.1 has an (exact) binary representation of 0.0001100110011001100110011001100110011... Such a number cannot be stored exactely on a computer due to the computer's limited number of bits to store the digits of a number. In such a case, the number is either rounded or chopped. For example, on a 8 bit computer that chops numbers, the first 8 digits of the binary representation of 0.1 are retained (i.e. 0.0001100). This obviously incurs an error since $0.0001100 \approx 0.09375$ and not 0.1.

The patriot missile system that we discussed in class runs a clock that counts time in increments of 0.1s (called clock cycle) and stores those in a 24bit binary register. The 24 bit representation of 0.1 is equivalent to the number 0.09999990463256836.

1. (10 pts) Calculate the absolute true error incurred in a single clock cycle by the Patiot system.
2. (10 pts) If the Patriot rocket battery had been running for 100 hours, what is the error (in seconds) due to roundoff?
3. (10 pts) When a missile is detected, the patriot system takes the difference in time between two radar pulses to measure the speed and location of the detected missile. For the error you calculated in question 2, what would be the distance calculated by the patriot system if it detects a scud missile travelling at 1676 m/s? The patriot system will not engage if the detected missile is outside a range of 167 m. Will this patriot system engage this scud?

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 Exponential 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 sine function is

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \text{ for all } x \tag{1}$$

   Test your script to compute $\sin(1.5)$ 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:

```
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,'%')
```

Problem 3 (35 pts)

The first derivative of a function can be approximated numerically as

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{2}$$

This is the called the forward difference formula and introduces a truncation error. Your goal in this problem is to evaluate this trunction error and see how it reacts to changing h.

1. (10 pts) For $f(x) = e^x$, compute the relative approximate error ε_a for $f'(2)$ using the following values of $h = (1, 0.5, 0.25, 0.125, 0.0625)$. Present your results in a table.

2. (5 pts) From the results you just computed, what happens to the error when h is reduced in half? (Hint: Is the error reduced? and by how much? compare how the error changes as h changes)

3. (10 pts) Now use the following approximation for the derivative

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{3}$$

to compute $f'(2)$. Use $h = (1, 0.5, 0.25, 0.125, 0.0625)$. Present your results in a table.

4. (5 pts) From the results you just computed, what happens to the error when h is reduced in half?

5. (5 pts) Which of these two methods to compute first derivatives would you rather use? (consider the calculation cost and error in your answer).

Hint: Define a function in python for equation (2) and equation (3):

```
def df(f, x, h):
        return (f(x+h) - f(x))/h
```

and then call each of these defined functions as needed. For example, to approximate the first derivative for the sine function at $x = 2$, use:

```
from math import sin
x = 2
result = df(sin, 2, 0.1)
```