Homework2_writeup_Jupyter

January 21, 2023

1 Homework 2 writeup template

1.1 Name: Aqua Karaman

1.2 Section: C

1.3 Problem 1

(Make sure your code is somewhere)

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

y1 = 0
term = 0.1
for k in range(100000):
    y1 = y1 + term

print(y1)
A2 = y1

y2 = 0
term = 0.1
for k in range(100000000):
    y2 = y2 + term

print(y2)
A3 = y2

# And y3
y3 = 0
term = 0.25
for k in range(100000000):
    y3 = y3 + term

print(y3)
A4 = y3

y4 = 0
```

```python
term = 0.5
for k in range(100000000):
    y4 = y4 + term

print(y4)
A5 = y4

x1 = np.abs(10**4 - y1)
x2 = np.abs(y2 - 10**7)
x3 = np.abs(2.5 * 10**7 - y3)
x4 = np.abs(y4 - 5 * 10**7)

print(x1,x2,x3,x4)
```

```
10000.000000018848
9999999.98112945
25000000.0
50000000.0
1.8848368199542165e-08 0.018870549276471138 0.0 0.0
```

1.3.1 Part a

In Problem 2 of the coding portion of the homework, I found the following values for x_1, x_2, x_3, and x_4.

x_1	x_2	x_3	x_4
1.8848368199542165e-08	0.018870549276471138	0.0	0.0

1.3.2 Part b

The resulting values from the code are definitely surprising. I wasn't expecting any of them to be zero, but interestingly the last two are. I will say I am unsurprised by x_1 and x_2, as they have approximately the same error just to varying degrees ($\approx 1.88 \times 10^{\text{a certain power}}$), which is understandable since the code to get those values is the same but also to a varying degree. However, x_3 and x_4 are output as perfect 0's, despite the preceding code for each x_n being the same structure with slightly different base values.

1.3.3 Part c

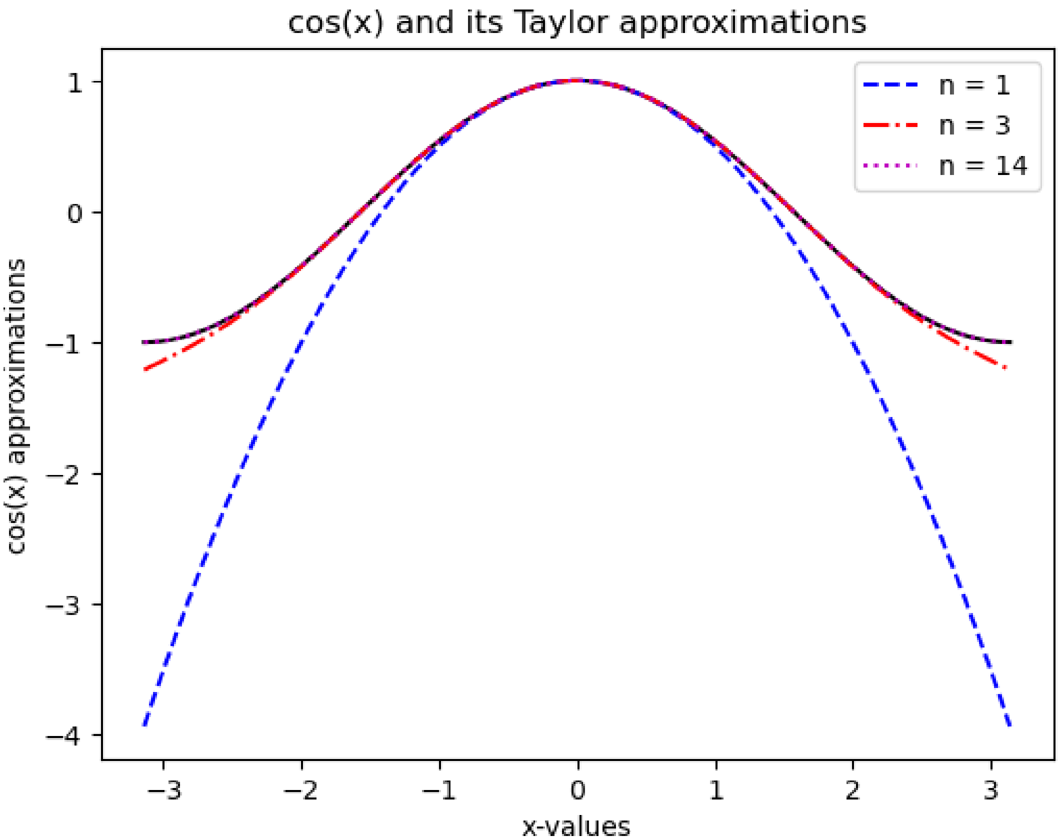
x_3 and x_4 are exactly zero. I can only assume is has something to do with the value of the base as an exponential value of 2 (binary system). Since 0.5 is exactly equal to 2^{-1}, and there's no exact, storable decimal answer that solves the equation $2^n = 0.1$ (symbolic calculation/CAS would be necessary to do this on a computer), the computer's answer is subject to truncation error over larger intervals (such as 10^6).

1.4 Problem 2

[ ]:
```
x = np.linspace(-np.pi,np.pi,100)
print(len(x))

T1 = np.zeros(100)
for k in range(2):
    T1 = T1 + (-1)**k / np.math.factorial(2*k) * x**(2*k)

T3 = np.zeros(100)
for k in range(4):
    T3 = T3 + (-1)**k / np.math.factorial(2*k) * x**(2*k)

T14 = np.zeros(100)
for k in range(15):
    T14 = T14 + (-1)**k / np.math.factorial(2*k) * x**(2*k)

print(len(T1),len(T3),len(T14)) # double checking the length lol

plt.plot(x, np.cos(x), 'k')
plt.plot(x, T1, '--b', label="n = 1")
plt.plot(x, T3, '-.r', label="n = 3")
plt.plot(x, T14, ':m', label="n = 14")

plt.xlabel('x-values')
plt.ylabel('cos(x) approximations')
plt.title('cos(x) and its Taylor approximations', fontsize=12)

plt.legend()
```

```
100
100 100 100
```

[ ]:
```
<matplotlib.legend.Legend at 0x1c405b8e520>
```


cos(x) and its Taylor approximations
n = 1
n = 3
n = 14
cos(x) approximations
x-values
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3