Nick Wurzer

V00958568

CSC349a

Rich Little

September 26, 2021

Assignment 1

For this assignment I used Python. All the functions which would normally be .m files are contained in the file a1_functions.py. I wrote a script called a1_script.py which prints relevant information for the questions and calls functions euler, euler2, mclauren1 and mclauren2. Below is the output from a1_script.py

-------------------------------Answers to question 1------------------

Question 1, part b:

Euler approximation using equation 1:

values of t	approximations of v(t)
0	0
0.60	5.88600000
1.20	11.23755690
1.80	16.10319762
2.40	20.52704287
3.00	24.54920726
3.60	28.20616302
4.20	31.53107073
4.80	34.55408005
5.40	37.30260305

6.00	39.80156282
6.60	42.07361946
7.20	44.13937557
7.80	46.01756301
8.40	47.72521286
9.00	49.27780976
9.60	50.68943236
10.20	51.97288099
10.80	53.13979374
11.40	54.20075193
12.00	55.16537615

Question 1 part c:

Euler approximation using equation 1:

values of t	approximations of v(t)
0	0
0.60	5.29800000
1.20	10.11494673
1.80	14.49451936
2.40	18.47643104
3.00	22.09678900
3.60	25.38842196
4.20	28.38117784
4.80	31.10219438
5.40	33.57614525
6.00	35.82546378
6.60	37.87054637

7.20	39.72993744
7.80	41.42049760
8.40	42.95755653
9.00	44.35505200
9.60	45.62565624
10.20	46.78089084
10.80	47.83123126
11.40	48.78620179
12.00	49.65446192

Question 1 part d:

The true value for Q1b: 54.27894360

The relative error for Q1b is: 0.0163

-------------------------------Answers to question 2------------------

Question 2 part b:

Euler approximation using equation 2:

values of t	approximations of v(t)
0	0
0.60	5.88600000
1.20	11.71311190
1.80	17.36591023

2.40	22.73930603
3.00	27.74640196
3.60	32.32382157
4.20	36.43386451
4.80	40.06356090
5.40	43.22129839
6.00	45.93201363
6.60	48.23194968
7.20	50.16376730
7.80	51.77249057
8.40	53.10247525
9.00	54.19537413
9.60	55.08894972
10.20	55.81653749
10.80	56.40696589
11.40	56.88476843
12.00	57.27056124

Question 2 part c:

The true value at t=12 for Q2b is: 58.5724

The relative error for Q2b is: 0.0222

-------------------------------Answers to question 3------------------

Question 3 equation 1:

McLauren approximation using equation 1

Degree of polynomial	Approximations of e^x
0	1.00000000
1	-1.75000000
2	2.03125000
3	-1.43489583
4	0.94807943
5	-0.36255697
6	0.23815138
7	0.00215882
8	0.08328126
9	0.05849385
10	0.06531039
11	0.06360625
12	0.06399678

The true value of e^-2.75 is: 0.06392786

The relative error for Q3a is: 0.00107809

Question 3 equation 2:

McLauren approximation using equation 2

Degree of the polynomial	Approximations of e^x
0	1.00000000
1	0.26666667
2	0.13278008
3	0.09093062

4	0.07473634
5	0.06806885
6	0.06539489
7	0.06440100
8	0.06406630
9	0.06396472
10	0.06393684
11	0.06392988
12	0.06392828

The true value of e^-2.75 is: 0.06392786

The relative error for Q3b is: 0.00000655

We can conclude that equation 2 approaches the true value of e^(-2.75) faster than equation 1. If I had to guess, this is because equation 1 'wastes' part of the next iteration by jumping over the true value. Equation 2 is not an alternating series and no part of the next iteration is destructive in getting closer to the true value.

a1_functions.py

```
#!/usr/bin/python
#Author: Nick Wurzer
#V00958568
#University of Victoria
#CSC349a
#Rich Little
```

```python
import math as m


def euler(m, c, g, t0, v0, tn, n):
    print("\nEuler approximation using equation 1:\n")
    print(f'{"values of t":16}{"approximations of v(t)":16}')
    print(f"{t0:<16}{v0:<16}")
    h = (tn-t0)/n
    t = t0
    v = v0
    results = []
    for i in range(n):
        v = v + (g - c / m * v) * h
        results.append(v)
        t = t + h
        print(f"{t:<16.2f}{v:<16.8f}")
    print()
    return results

def euler2(m, k, g, t0, v0, tn, n):
    print("\nEuler approximation using equation 2:\n")
    print(f'{"values of t":16}{"approximations of v(t)":16}')
    print(f"{t0:<16}{v0:<16}")
    h = (tn-t0)/n
    t = t0
    v = v0
    results = []
    for i in range(n):
        v = v + (g - k / m * (v**2)) * h
        results.append(v)
```

```
        t = t + h
        print(f"{t:<16.2f}{v:<16.8f}")
    print()
    return results

def mclauren1(x, n):
    print("\nMcLauren approximation using equation 1\n")
    print(f'{"Degree of polynomial":32}{"Approximations of e^x":32}')
    appx = 0
    results = []
    for i in range(n+1):
        appx = appx + (((-1)**(i)) * ((-x)**(i))) / m.factorial(i)
        print(f"{i:<32}{appx:<32.8f}")
        results.append(appx)
    return results

def mclauren2(x, n):
    print("\nMcLauren approximation using equation 2\n")
    print(f'{"Degree of the polynomial":32}{"Approximations of e^x":32}')
    denominator = 0
    results = []
    for i in range(n+1):
        denominator = denominator + (-x)**i / m.factorial(i)
        appx = 1/denominator
        print(f"{i:<32}{appx:<32.8f}")
        results.append(appx)
    return results
```

a1_script.py

```python
#!/usr/bin/python

#Author: Nick Wurzer
#V00958568
#University of Victoria
#CSC349a
#Rich Little



"""This is a script for assignment 1.  It calls functions from a1_functions.py.

Run this script with A1_functions.py in the same folder to see printed answers to questions from assignment 1.

I've commented out a graph which I thought would be fun to code."""

import a1_functions as A1F
import math as m
import matplotlib.pyplot as plot
import numpy as np

def main():
	print("\n------------------------------Answers to question 1------------------------------\n")
	#A1F.euler(68.1, 12.5, 9.81, 0, 0, 12, 6)

	print("Question 1, part b:\n")
	_1b = A1F.euler(82.6, 12.5, 9.81, 0, 0, 12, 20)
```

```
print("Question 1 part c:\n")
A1F.euler(82.6, 12.5, 8.83, 0, 0, 12, 20)

print("Question 1 part d:\n")
true_value = 9.81*82.6/12.5*(1 - m.exp((-12.5)*12/82.6))
print(f"\nThe true value for Q1b: {true_value:.8f}")
relative_error =  abs(((_1b[-1])-true_value)/true_value)
print(f"\nThe relative error for Q1b is: {relative_error:.4f}")



print("\n\n------------------------------Answers to question 2------------------------------\n")

print("Question 2 part b:\n")
_2b = A1F.euler2(82.6, .234, 9.81, 0, 0, 12, 20)

print("Question 2 part c:\n")
true_value = ((9.81*82.6/.234)**(1/2))*m.tanh((9.81*.234/82.6**(1/2))*12)
print(f"\nThe true value at t=12 for Q2b is: {true_value:.4f}")
relative_error = abs(((_2b[-1])-true_value)/true_value)
print(f"\nThe relative error for Q2b is: {relative_error:.4f}")
#Uncomment these lines to see a graph of results from Q2b
#note: right now graph does not start at (0,0), first point is (0,5.88) and therefore needs a
translation by 1 on the x-axis.
#y = np.array(_2b)
#x = np.arange(0,12,(12/20))
#print(y)
#print(x)
#plot.plot(x, y)
#plot.show()
```

```
    print("\n-------------------------------Answers to question 3-------------------------------\n")

        print("Question 3 equation 1:\n")
        _3a = A1F.mclauren1(-2.75, 12)
        true_value = m.exp(-2.75)
        relative_error = abs(((_3a[-1])-true_value)/true_value)
        print(f"\nThe true value of e^-2.75 is: {true_value:.8f}")
        print(f"\nThe relative error for Q3a is: {relative_error:.8f}")

        print("\nQuestion 3 equation 2:\n")
        _3b = A1F.mclauren2(-2.75, 12)
        true_value = m.exp(-2.75)
        relative_error = abs(((_3b[-1])-true_value)/true_value)
        print(f"\nThe true value of e^-2.75 is: {true_value:.8f}")
        print(f"\nThe relative error for Q3b is: {relative_error:.8f}")

        print("\nWe can conclude that equation 2 approaches the true value of e^(-2.75) faster than equation 1.\nIf I had to guess, this is because equation 1 'wastes' part of the next iteration by jumping over the true value.\nEquation 2 is not an alternating series and no part of the next iteration is destructive in getting closer to the true value.")


if __name__ == "__main__":
        main()
```