HW5

xh2433 Xuhang He

04/19/2024

1 Question 1

For any Lagrange approximation, the maximum error is $|f(x) - p_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |\pi_{n+1}(x)|$, where $\pi_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)$ and $M_{n+1} = \max_{\epsilon \in [a,b]} \left| f^{(n+1)}(\epsilon) \right|$. In this question, $a = -1$ and $b = 1$. Whether n is odd or even, since $f = sin(x)$, $f^{(n+1)}$ is either $\pm sin(x)$ or $\pm cos(x)$. In either cases, M_{n+1} is upper bounded by 1. For any fixed n, $(n+1)!$ is a constant, so upper bound for $|f(x) - p_n(x)|$ is the upper bound for $\pi_{n+1}(x)$. Notice that $x, x_i \in [-1, 1]$, so $x - x_i \leq 2$, so $|\pi_{n+1}(x)| \leq \left|2^{n+1}\right|$, so the maximum error is upper bounded by $\frac{\left|2^{n+1}\right|}{(n+1)!}$.

2 Question 2

python code:

```python
import numpy as np
import matplotlib.pyplot as plt

def L(val, k, x):
    xk = x[k]
    x_not_k = np.delete(x, k)
    over = val - x_not_k
    under = xk - x_not_k
    res = np.prod(over / under)
    return res

def LPrime(val, k, x):
    xk = x[k]
    xInter = np.delete(x, k)
    coeff = 1 / (xk - xInter)
    vals = np.zeros(xInter.shape)
    for j in range(vals.shape[0]):
        xInter2 = np.delete(xInter, j)
        over = val - xInter2
        under = xk - xInter2
        vals[j] = np.prod(over / under)
    res = np.dot(coeff, vals)
    return res

def Hk(val, k, x):
    return (L(val, k, x) ** 2) * (1 - 2*LPrime(x[k], k, x) * (val - x[k]))

def Kk(val, k, x):
    return (L(val, k, x) ** 2) * (val - x[k])

def P2nPlus1(val, x, y, z):
    outterRes = np.zeros(val.shape)
    for i in range(val.shape[0]):
        res = np.zeros(x.shape)
        for k in range(x.shape[0]):
            res[k] = Hk(val[i], k, x) * y[k] + Kk(val[i], k, x) * z[k]

        outterRes[i] = res.sum()
    return outterRes

def derivativeTanh(x):
    return 1 - np.power(np.tanh(x), 2)

x = np.arange(21)

y = np.tanh(x)

z = derivativeTanh(x)

testX = np.arange(0, 25, 0.5)
trueY = np.tanh(testX)
print(trueY)
myY = P2nPlus1(testX, x, y, z)

plt.plot(testX, trueY, color="red", label="Ground Truth")
plt.plot(testX, myY, color="blue", label="Approximation")
plt.legend()
plt.show()
```

3 Question 3

For $l_0(x)$, it should only be 1 when $x = x_0$ and for all other $x_i \neq x_0$, l_0 should be 0. Similarly for For $l_n(x)$, it should only be 1 when $x = x_n$ and for all other

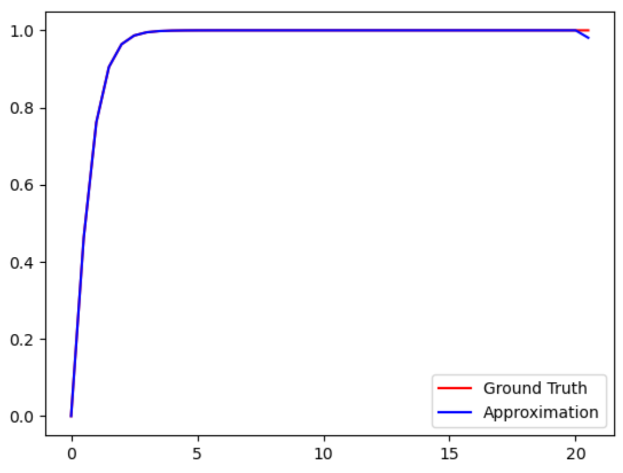

Figure 1: Ploted Image from Question 2

$x_i \neq x_n$, l_n should be 0. $h_i(x)$ should only be 1 when $x = x_i$ and 0 for all other x_k. Need to mention is that we do not care the values of l_0' and l_n' because in this question it is not necessary to interploate $f'(x_0)$ and $f'(x_n)$. $h_i'(x)$ should be 0 for all x_i for $i = 1, ..., n-1$. $k_i(x)$ should always be 0 for all x_i for $i = 1, ..., n-1$, and $k_i'(x) = 1$ when $x = x_i$ and $k_i'(x) = 0$ for all other x_k where $k \neq i$.

Suppose M_i, $M_i(x) = \prod_{k=1,k\neq i}^{n-1} \frac{x-x_k}{x_i-x_k}$. A possible solution for $h_i(x)$ is $M_i(x)L_i(x)(1-(x-x_i)(M_i'(x_i)) + L_i'(x_i))$, which satisfies all the above requirements. A possible solution for $k_i(x)$ is $M_i(x)L_i(x)(x-x_i)$, which satisfies all the above requirements. A possible solution for l_0 is $M_0(x)L_0(x)$, which satisfies all the above requirements. A possible solution for l_n is $M_n(x)L_n(x)$, which satisfies all the above requirements.

4 Question 4

python code:

```
import numpy as np
import matplotlib.pyplot as plt

def trapzium(a, b, func):
    return (b - a) * (func(a) + func(b)) / 2

def simpsons(a, b, func):
    return (b - a) * (func(a) + 4 * func((a+b)/2) + func(b)) / 6
```

```
def func1(x):
    return np.exp(np.power(x, 3)) + np.sin(np.power(x, 2)) + x

def func2(x):
    return np.cos(np.power(x, 5) + 4 * np.power(x, 4) + 3 * np.power(x, 3) + 2*np.power(x, 

h = np.array([1, 0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625])

func1_trap = trapzium(-h, h, func1)
func1_simp = simpsons(-h, h, func1)

plt.plot(h, func1_trap, color="red", label="Trapzium Func1")
plt.plot(h, func1_simp, color="blue", label="Simpsons Func1")
plt.legend()
plt.show()

func2_trap = trapzium(0, h, func2)
func2_simp = simpsons(0, h, func2)

plt.plot(h, func2_trap, color="red", label="Trapzium Func2")
plt.plot(h, func2_simp, color="blue", label="Simpsons Func2")
plt.legend()
plt.show()
```

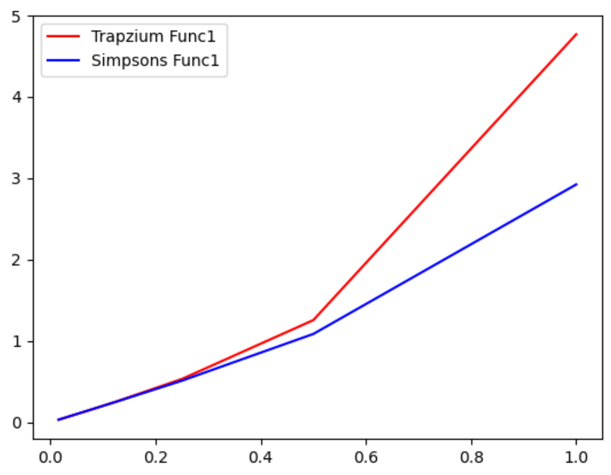

Figure 2: plot for function 1

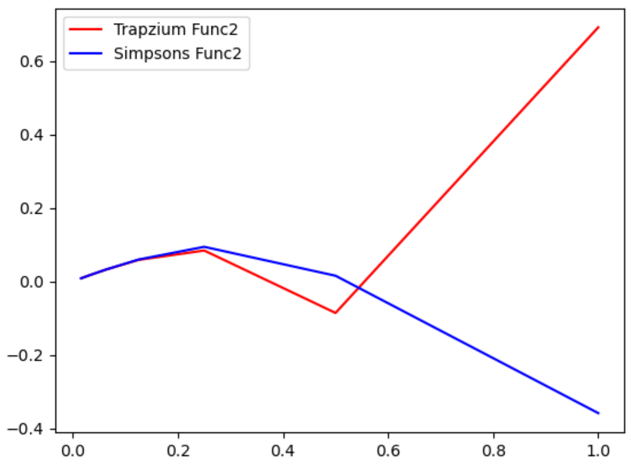

Figure 3: plot for function 2

5 Question 5

5.1 (1)

$L_0 = \frac{-x(x-0.5)(x-1)}{3}$
$L_1 = 2(x+1)(x-0.5)(x-1)$
$L_2 = \frac{-8(x+1)(x-1)x}{3}$
$L_3 = (x+1)(x-0.5)x$

$$P_3 = \frac{x(x-0.5)(x-1)}{3} + \frac{8(x+1)(x-1)x}{6} + (x+1)(x-0.5)x$$

5.2 (2)

Since $P_3(x_i) = f(x_i)$, $w_i = \int_0^1 L_i$. $w_0 = 0, w_1 \approx 0.167, w_2 \approx 0.667, w_3 \approx 0.167$