Math104A homework2

January 25, 2021

1. Write a computer code to implement the Composite Trapezoidal Rule.

$$T_h(f) = h[\frac{1}{2}f(x_0) + f(x_1) + \cdots + f(x_{N-1}) + \frac{1}{2}f(x_N)].$$

to approximate the definite integral

$$I(f) = \int_a^b f(x)dx$$

[133]:
```
# Our function should have 4 inputs: Function f, Start point a, End point b,␣
↪and the interval length h.
# The output should be the value of CTR
# Yubo 01/25
import numpy as np
import matplotlib.pyplot as plt

def CTR(f, a, b, h):
    N = (b-a)/h
    Count = 0.5*f(a) + 0.5*f(b)
    for i in range(1,int(N)):
        Count += f(a+h*i)
    T = Count*h
    return T
```

[151]:
```
# h = 2/20
f = lambda x: (1/((1+x)**2))
Estimate_1 = CTR(f, 0, 2, 2/20)
f_int = lambda x : -1/(x+1)
Exact_1 = f_int(2)-f_int(0)
Error_1 = abs(Exact_1-Estimate_1)
print('The estimated value is',Estimate_1 )
print('The exact value is',Exact_1 )
print('Thus, the error for h =2/20 is', Error_1)
```

```
The estimated value is 0.6682683087950135
The exact value is 0.6666666666666667
Thus, the error is 0.0016016421283467919
```

[152]:
```
# h = 2/40
f = lambda x: (1/((1+x)**2))
Estimate_2 = CTR(f, 0, 2, 2/40)
f_int = lambda x : -1/(x+1)
Exact_2 = f_int(2)-f_int(0)
Error_2 = abs(Exact_2-Estimate_2)
print('The estimated value is',Estimate_2 )
print('The exact value is',Exact_2 )
print('Thus, the error for h = 2/40 is', Error_2)
```

```
The estimated value is 0.6670676941291324
The exact value is 0.6666666666666667
Thus, the error is 0.0004010274624656196
```

[154]:
```
# h = 2/80
f = lambda x: (1/((1+x)**2))
Estimate_3 = CTR(f, 0, 2, 2/80)
f_int = lambda x : -1/(x+1)
Exact_3 = f_int(2)-f_int(0)
Error_3 = abs(Exact_3-Estimate_3)
print('The estimated value is',Estimate_3 )
print('The exact value is',Exact_3 )
print('Thus, the error for h = 2/80 is', Error_3)
```

```
The estimated value is 0.6667669623471976
The exact value is 0.6666666666666667
Thus, the error for h = 2/80 is 0.00010029568053082638
```

- Now in order to compute the rate of convergence, we simply need to compute the division of the errors and find the relationship with 1/2.

[158]:
```
Rate_1 = Error_2/Error_1
Rate_2 = Error_3/Error_2
print (Rate_1,Rate_2)
```

```
0.25038518615863237 0.25009678866924184
```

- It has a convergence linear trend at the rate of 2.

[217]:
```
# N = 16, 32, 64, 128
g = lambda x: x**(1/2)
g_p = lambda x: (2/3)*(x**(2/3)) #integral
ex = g_p(1)-g_p(0) # real value
es_1 = CTR(g, 0, 1, 1/16) # estimated
er_1 = abs(es_1-ex) # error
es_2 = CTR(g, 0, 1, 1/32)
er_2 = abs(es_2-ex) # error
es_3 = CTR(g, 0, 1, 1/64)
er_3 = abs(es_3-ex) # error
```

```
es_4 = CTR(g, 0, 1, 1/128)
er_4 = abs(es_4-ex) # error
print(er_1,er_2,er_3,er_4)
ratio_1 = er_2/er_1
ratio_2 = er_3/er_2
ratio_3 = er_4/er_3
print(ratio_1,ratio_2,ratio_3)
```

```
0.0030854697894383554 0.0011077303877248257 0.0003958552881599964
0.00014100936984029477
0.3590151462563711 0.35735707221415675 0.3562144401195968
```

$ I don't see a second order convergence to the exact value of the integral. $ $ From the testing, we can see that the second order convergence is not always the case.$ $ We need a different N value to get the second order convergence$

2.

$$q(h) = \frac{T_{\frac{h}{2}}\left(\cos x^2\right) - T_h\left(\cos x^2\right)}{T_{\frac{h}{4}}\left(\cos x^2\right) - T_{\frac{h}{2}}\left(\cos x^2\right)}$$

Using your code, find a value of h for which $q(h)$ is approximately equal to 4.

[254]:
```
# so first we define a function that shows us if two numbers are "approximately␣
↪equal".
# the input is two numbers and the number of significance figures, the default␣
↪is 5, while usually is 2 or 3.
# the output is T or F
def nearly_equal(a,b,sig_fig=5):
    return ( a==b or int(a*10**sig_fig) == int(b*10**sig_fig))
```

[309]:
```
# function k
import random
k = lambda x : np.cos(x**2)
for i in range(1000):
    h = random.uniform(0, 1)
    if (nearly_equal(((CTR(k,0,(np.pi/2)**(1/2),h/2)-CTR(k,0,(np.pi/2)**(1/
↪2),h))/(CTR(k,0,(np.pi/2)**(1/2),h/4)-CTR(k,0,(np.pi/2)**(1/2),h/2))), 4,␣
↪3)== True):
        print(h)
        break
# the sig_fig was set to 3.
```

```
0.10473044287621969
```

[350]:
```
# test
h = 0.10473044287621969
(CTR(k,0,(np.pi/2)**(1/2),h/2)-CTR(k,0,(np.pi/2)**(1/2),h))/(CTR(k,0,(np.pi/
↪2)**(1/2),h/4)-CTR(k,0,(np.pi/2)**(1/2),h/2))
```

[350]: 4.000147040007715

[403]:
```
h = 0.10473044287621969
CTR(k,0,(np.pi/2)**(1/2),h)
```

[403]: 0.9503280714633465

[404]:
```
# by using the calculator, the exact value should be 0.977451424291329743....
# So the error would be
Error_1 = abs(CTR(k,0,(np.pi/2)**(1/2),h)-0.977451424291329743)
print ('The error would be',Error_1)
```

```
The error would be 0.02712335282798317
```

[405]:
```
T_h = CTR(k,0,(np.pi/2)**(1/2),h)
T_05_h = CTR(k,0,(np.pi/2)**(1/2),h/2)
S_h = T_h + (4/3)*(T_05_h - T_h)
# Our new error should be
Error_2 = abs(S_h - 0.977451424291329743)
print('Our new Error is', Error_2)
```

```
Our new Error is 0.00018422782699334395
```

- Explain why $S_h(\cos x^2)$ is more accurate and converges faster to $I(\cos x^2)$ than $T_h(\cos x^2)$

$ The reason is that Simpson's rule uses quadratic approximations instead of linear approximations.$
$In other words, it approximates the curve with a sequence of quadratic parabolic segments instead of straight li
$

Using your code, find a value of h for which $q_1(h)$ is approximately equal to 16

$$q_1(h) = \frac{S_{\frac{h}{2}}\left(\cos x^2\right) - S_h\left(\cos x^2\right)}{S_{\frac{h}{4}}\left(\cos x^2\right) - S_{\frac{h}{2}}\left(\cos x^2\right)}$$

[406]:
```
T_h = CTR(k,0,(np.pi/2)**(1/2),h)
T_05_h = CTR(k,0,(np.pi/2)**(1/2),h/2)
T_025_h = CTR(k,0,(np.pi/2)**(1/2),h/4)
T_0125_h = CTR(k,0,(np.pi/2)**(1/2),h/8)
S_h = T_h + (4/3)*(T_05_h - T_h)
S_05h = T_05_h + (4/3)*(T_025_h - T_05_h)
S_025h = T_025_h + (4/3)*(T_0125_h - T_025_h)
q_1h = ((S_05h-S_h)/(S_025h-S_05h))
q_1h
# Above are our new setup
```

[406]: 0.002828457299368575

[415]:
```
for i in range(10000):
    h = random.uniform(0, 1)
    T_h = CTR(k,0,(np.pi/2)**(1/2),h)
    T_05_h = CTR(k,0,(np.pi/2)**(1/2),h/2)
    T_025_h = CTR(k,0,(np.pi/2)**(1/2),h/4)
    T_0125_h = CTR(k,0,(np.pi/2)**(1/2),h/8)
    S_h = T_h + (4/3)*(T_05_h - T_h)
    S_05h = T_05_h + (4/3)*(T_025_h - T_05_h)
    S_025h = T_025_h + (4/3)*(T_0125_h - T_025_h)
    q_1h = ((S_05h-S_h)/(S_025h-S_05h))
    if nearly_equal(q_1h,16,2):
        print(h)
        break
# the sig_fig was set to 2.
```

```
0.8431715240139113
```

[416]:
```
# test
h = 0.8431715240139113
T_h = CTR(k,0,(np.pi/2)**(1/2),h)
T_05_h = CTR(k,0,(np.pi/2)**(1/2),h/2)
T_025_h = CTR(k,0,(np.pi/2)**(1/2),h/4)
T_0125_h = CTR(k,0,(np.pi/2)**(1/2),h/8)
S_h = T_h + (4/3)*(T_05_h - T_h)
S_05h = T_05_h + (4/3)*(T_025_h - T_05_h)
S_025h = T_025_h + (4/3)*(T_0125_h - T_025_h)
q_1h = ((S_05h-S_h)/(S_025h-S_05h))
print(q_1h)
# close to 16.
```

```
16.00656679075275
```

3. Show that these assertions are not true.

$$e^x - 1 = \mathcal{O}\left(x^2\right) \text{ as } x \to 0$$

Solution:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

$$e^x = 1 + x + \frac{x^2}{2!} + \mathcal{O}\left(x^3\right) \text{ as } x \to 0$$

$$e^x - 1 = x + \mathcal{O}\left(x^2\right) \text{ as } x \to 0$$

Thus the statement is false. Plus, the limit of it goes to infty, not a constant.

$$x^{-2} = \mathcal{O}(\cot x) \text{ as } x \to 0$$

Solution:

$$\lim_{x \to 0} \frac{1}{x^2 \cot x}$$

$$\lim_{x \to 0} \frac{\sin x}{x^2 \cos x} = \lim_{x \to 0} \frac{1}{\cos(x)} \lim_{x \to 0} \frac{\sin(x)}{x^2}$$

$$= \lim_{x \to 0} \frac{\sin x}{x^2}$$

$$= \lim_{x \to 0} \frac{\cos x}{2x}$$

$$= \lim_{x \to 0} \frac{1}{2x} \lim_{x \to 0} \cos x = \infty$$

Thus, the statement is false.

$$\cot x = o\left(x^{-1}\right) \text{ as } x \to 0$$

Solution:

$$\lim_{x \to 0} \frac{x \cos x}{\sin x}$$

$$= \lim_{x \to 0} \cos x \lim_{x \to 0} x \sin x$$

$$= \lim_{x \to 0} \frac{x}{\sin x}$$

$$= \lim_{x \to 0} \frac{1}{\cos x}$$

$$= 1$$

Thus the statement is false.