Homework #5

Name - Rakshya Sharma

Subject - ASTR-119

Assignment - Homework 5

Date - November 3, 2021

Purpose - The purpose of this assignment is to write a jupyter notebook to numerically integrate the given function using the trapezoid, Simpson's method and Romberg integration.

References - Professor Brant's code from Tuesday, October 26 lecture and session 10 slides. Python Data Science Handbook Pg. 55 (Exponents and Logarithms),

In [496]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

Using Trapezoidal, Simpson's Method and Romberg Integration

Let's define the function to integrate

In [497]:
```
def func(x):
    a = -2
    b = 10
    return np.exp(a*x) * np.cos(b*x)
```

Define it's integral so we know the right answer

In [498]:
```
def func_integral(x):
    a = -2
    b = -10
    c = 0

    return np.exp(a*x) *(5* np.sin(10*x) + np.cos(10*x))/52. + c*x
```

Define the core of trapezoid method

In [499]:
```
def trapezoid_core(f,x,h):
    return 0.5*h*(f(x+h) + f(x))
#evaluating in the left side of trapezoid
```

Define the wrapper function to perform the trapezoid method

In [500]:
```
def trapezoid_method(f,a,b,N):
    #f == function to integrate
    #a == lower limit of integration
    #b == upper limit of integration
    #N == number of intervals to use

    #define x values to perform the trapezoid rule
    x = np.linspace(a,b,N)

    h = x[1]-x[0]

    #define value of the integral
    Fint = 0.0

    #perform the integral using the
    #trapezoid method

    for i in range(0,len(x)-1, 1):
        Fint += trapezoid_core(f,x[i],h)

    #return the answer
    return Fint
```

Define the core of simpson's method

In [501]:
```
def simpson_core(f,x,h):
    return h*(f(x) + 4*f(x+h) + f(x+2*h))/3.
```

Define a wrapper for Simpson's method

In [502]:
```
def simpsons_method(f,a,b,N):
    #f == function to integrate
    #a == lower limit of integration
    #b == upper limit of integration
    #N == number of intervals to use
    #note the number of chunks will be N-1
    #so if N is odd, then we don't need to
    #adjust the last segment

    #define x values to perform the simpsons rule
    x = np.linspace(a,b,N)
    h = x[1]-x[0]

    #define the value of the integral
    Fint = 0.0

    #perform the integral using the
    #simpsons method

    for i in range(0, len(x)-2, 2):
        Fint += simpson_core(f,x[i],h)

    #apply simpson's rule over the last interval
    #if N is even
    if((N%2)==0):
        Fint += simpson_core(f,x[-2],0.5*h)

    #return the answer
    return Fint
```

Define Romberg core

In [503]:
```
def romberg_core(f,a,b,i):

    # we need the difference between a and b
    h = b-a

    #interval between function evaluation at refine level i
    dh = h/2. ** (i)

    #we need the cofactor
    K = h/2. ** (i+1)

    #and the function evaluations
    M = 0.0
    for j in range(2**i):
        M += f(a + 0.5*dh + j*dh)

   #return the answer
    return K*M
```

Define a wrapper function

In [504]:
```
def romberg_integration(f,a,b,tol):

    #define an iteration variable

    i = 0

    #define a max number of iteration
    imax = 1000

    #define an error estimate
    delta = 100.0*np.fabs(tol)

    #set an array of integral answers
    I = np.zeros(imax, dtype = float)

    #get the zeroth romberg iteration first
    I[0] = 0.5*(b-a)*(f(a) + f(b))

    #iterate by 1
    i += 1

    #iterate until we reach tolerance
    while(delta>tol):

        #find the Romberg iteration
        I[i] = 0.5*I[i-1] + romberg_core(f,a,b,i)

        #compute a fractional error estimate
        delta = np.fabs( (I[i]-I[i-1])/I[i])

        print(i, I[i], I[i-1], delta)

        if(delta>tol):

            #iterate
            i += 1

            #if we've reach maximum iteration
            if(i>imax):
                print("Max iterations reached.")
                raise StopIteration("Stopping iterations after ", i)

     #return the answer
    return I[i]
```

In [505]:
```
Answer = func_integral(int(np.pi))-func_integral(0)#converting the constant np.pi into int
print(Answer)
print("Trapezoid")
print(trapezoid_method(func,0,int(np.pi),10))
print("Simpson's")
print(simpsons_method(func,0,int(np.pi),10))
print("Romberg")
tolerance = 1.0e-4
RI = romberg_integration(func, 0, int(np.pi), tolerance)
print(RI, (RI-Answer)/Answer, tolerance)
```

```
-0.019458905318788734
Trapezoid
0.05404385479060319
Simpson's
-0.10703440124949726
Romberg
1 0.801019205156373 1.500573526675744 0.8733302735017517
2 0.27474344698584363 0.801019205156373 1.9155170539796211
3 0.12696199913966824 0.27474344698584363 1.1639817334917995
4 0.07097479063964986 0.12696199913966824 0.7888323163117764
5 0.04458189512894176 0.07097479063964986 0.5920092771824386
6 0.03169051424349386 0.04458189512894176 0.40678989259678977
7 0.025315951934776575 0.03169051424349386 0.25180022166026217
8 0.022146142895665744 0.025315951934776575 0.14313142717647687
9 0.02056558720591901 0.022146142895665744 0.07685439146088816
10 0.01977639537121192 0.02056558720591901 0.03990574722509339
11 0.01938207088161097 0.01977639537121192 0.02034480690992998
12 0.019184976489074967 0.01938207088161097 0.010273371596167395
13 0.019086446255581015 0.019184976489074967 0.005162314250361921
14 0.019037185379509415 0.019086446255581015 0.0025876134044805427
15 0.019012556001641267 0.019037185379509415 0.0012954269728921277
16 0.01900024157774892 0.019012556001641267 0.0006481193326914876
17 0.018994084432063463 0.01900024157774892 0.0003241612254319936
18 0.018991005875784985 0.018994084432063463 0.00016210601474267618
19 0.018989466601788175 0.018991005875784985 8.105935933264823e-05
0.018989466601788175 -1.975875378942962 0.0001
```

3) How many iterations does Romberg integration take to reach the specified accuracy?

The Romberg integration depends on the refined width of intervals that we are calculating to the integrals over we get better approximation to the integral. So, function evaluated at 1 is four times more accurate than function evaluated at 0.

4) How many intervals does the trapezoid method need to reach the specified accuracy?

In the trapezoid method each interval has an error that is 3rd order accurate i.e as we decrease the interval between the places we evaluate the function the error on the integral decreases by that width to the third power over the small interval. It's usually 2nd order for the intervals to reach the specified accuracy.

5) How many intervals does Simpson’s method need to reach the specified accuracy?

Simpson's rule is fourth order accurate over a large range of integrals. We decrease the interval by a factor of two and then locally the error improves by a factor of 32.

In [ ]:
```

```