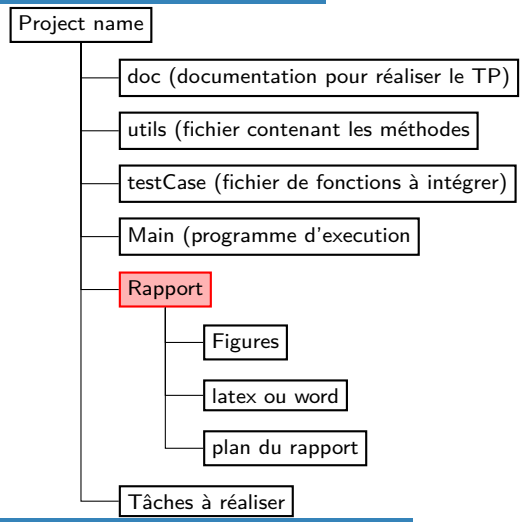
Worksheet for Numerical Integration in Python - Ma121

Feuille de route pour un TP



Utils to define integration methods

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
File: utils.py
Created on Wed Feb 16 11:47:43 2022
@author: H. El-Otmany
@ This file contain all method (RG,RD,MP,T)
@ Input data :
    - function to integrate f,
    - start value a,
    - end value b,
    - number of subdivison n
@ Output : computed value of integral
"""
import numpy as np
import matplotlib.pyplot as plt
#Formule du retcangle à gauche
def method_RG(f, a, b, n):
    h = (b-a)/n
    x = np.linspace(a, b-h, n)
    S = sum(f(x))
    return h*S
#Formule du rectangle à droite
def method_RD(f, a, b, n):
    h = (b-a)/n
    x = np.linspace(a+h,b,n)
    S = sum(f(x))
    return h*S
#Formule du point milieu
def method_MP(f, a, b, n):
    h = (b-a)/n
    x = np.linspace(a+h/2, b-h/2, n)
    S = sum(f(x))
    return h*S
#Formule des trapèzes
def method_T(f, a, b, n):
    h = (b-a)/n
    x = np.linspace(a+h, b-h, n-1)
    S = sum(f(x)) + (f(a) + f(b))/2
    return h*S
"""
@ plotting relative errors
"""
def errors_method(method, f, a, b, I_exact, n_liste):
    number = len(n_liste)
    err_liste = np.empty(number)
    for i in range(len(n_liste)):
        I_app = method(f, a, b, n_liste[i])
        err_liste[i]=abs((I_app-I_exact)/I_exact)
    return err_liste
methods={'Rectangles à gauche': method_RG,
         'Rectangles à droite':method_RD,
         'Point milieu':method_MP,
         'Trapèzes':method_T}
"""
@ You can also use this method for plotting errors
"""
def errors_method2(f, a, b, n_liste, I_ex, err_title):
    error_RG = []
    error_RD = []
    error_PM = []
    error_T = []
    I = abs(I_ex)
    for n in n_liste:
        error_RG.append(abs(I_ex - method_RG(f, a, b, n))/I)
        error_RD.append(abs(I_ex - method_RD(f, a, b, n))/I)
        error_PM.append(abs(I_ex - method_MP(f, a, b, n))/I)
        error_T.append(abs(I_ex - method_T(f, a, b, n))/I)
    plt.loglog(n_liste, error_RG,'o', label = 'Rect. gauche')
    plt.loglog(n_liste, error_RD,'+', label = 'Rect.droite')
    plt.loglog(n_liste, error_PM,'*', label = 'Point milieu')
    plt.loglog(n_liste, error_T,'x', label = 'Trapezes')
    plt.legend()
    plt.title(err_title)
    plt.show()
    return error_RG, error_RD, error_PM, error_T
```

testCase to define test functions

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
File: testCase.py
Created on Wed Feb 16 11:47:43 2022
@ This file contain all test cases
@author: H. El-Otmany
"""
from math import log, exp, sin,pi, cos
from utils import *
def trial_f1(x):
    return 1/x
def trial_f2(x):
    return log(x+1)/x
def trial_f3(x):
    return exp(sin(x))
def trial_f4(x):
```

```
    return cos(x*x)
def trial_g11(x):
    def f(r):
        return exp(-r*r)
    f1 = np.vectorize(f)
    return method_MP(f1,x,2*x,100)
def trial_square(x):
    square = trial_g11(x) **2
    return square
```

mainProg to run

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
File: mainProg.py
Created on Wed Feb 16 11:47:43 2022
@author: Hammou El-Otmany
@ this file used for testing
"""
import sys
import math
import numpy as np
from numpy import inf
from math import log, exp, sin,pi, cos
import matplotlib.pyplot as plt
#print method, value, error
#from prettytable import PrettyTable
#call method defined in scipy.integrate
from scipy.integrate import quad, simps
#importing all the functions
from utils import *
from testCase import *
#define variables a, b, n
a = 1
b = 3
#Exact value of Integral
I_ex= log(3)
"""
@ Call different method
@ we can use lambda x:1/(1+x**2)
#instead of trial_f1
 """
I_RG = method_RG(trial_f1, a, b, 50)
I_RD = method_RD(trial_f1, a, b, 50)
I_MP = method_MP(trial_f1, a, b, 50)
I_T = method_T(trial_f1, a, b, 50)
"""
@ Print method value
"""
print("RG method: ", I_RG,'\n',
      "RD method: ", I_RD,'\n',
      "MP method: ", I_MP,'\n',
      "Trap. method: ", I_T,'\n')
"""
@ Print method value for each N subdivisions
"""
for n in [10,50,1000]:
    I_RG = method_RG(trial_f1, a, b, n)
    I_RD = method_RD(trial_f1, a, b, n)
    I_MP = method_MP(trial_f1, a, b, n)
    I_T = method_T(trial_f1, a, b, n)
    print(" n value:",n,"RG method: ", I_RG,'\n',
      "n value:",n,"RD method: ", I_RD,'\n',
      "n value:",n,"MP method: ", I_MP,'\n',
      "n value:",n,"Trap. method: ", I_T,'\n')
"""
@ Print method value with prettytable
#  """
# MTables = PrettyTable()
# MTables.field_names = ['Methods','Value of I', 'Error']
# MTables.add_row(['RG method', I_RG,abs((I_ex-I_RG)/I_ex)])
# MTables.add_row(['RD method', I_RD,abs((I_ex-I_RD)/I_exx)])
# MTables.add_row(['MP method', I_MP,abs((I_ex-I_MP)/I_ex)])
# MTables.add_row(['Trapeze method',I_T,abs((I_ex-I_T)/I_ex)])
# print(MTables)
"""
@ Compute the integral by using scipy.integrate
 """
h = (b - a)/n
x = np.linspace(a, b, n)
y = trial_f1(x)
I_Tscipy = np.trapz(y,x,h)
print("Trapeze with scipy: ", I_Tscipy)
"""
@ plotting relative errors
"""
n_liste=list(range(10,1000,10))
#+ list(range(10,50))+list(range(50,200,5))+list(range(200,1000,20))
plt.figure()
for m in methods :
    method = methods[m]
    error = errors_method(method, trial_f1, a, b, I_ex, n_liste)
    plt.loglog(n_liste,error,label=m)
plt.legend()
plt.title('Évolution des erreurs relatives en fonction de n')
plt.show()

"""
@ Plotting errors with erors_method2
"""
plt.figure()
n_liste = list(range(10,1000,10))
error_title = 'Évolution des erreurs relatives en fonction de n'
errors_method2(trial_f1, 1, 3, n_liste, I_ex, error_title)
plt.show()
"""
@ Plotting g by using Trapeze's method
"""
plt.figure()
X = []
x = 0.
g = []
pas = 0.01
while x <= 3:
    X.append(x)
    g.append(trial_g11(x))
    x = x + pas
plt.plot(X, g,label='with n = 100')
plt.legend()
plt.title('Plotting g by using Trapeze')
plt.show()

"""
@ Approximated value of I
"""
I_value = method_T(trial_square,1,2,n)
print("value of I:", I_value)
for n in [10,50,100, 150, 250, 500, 800, 1000]:
    I_value = method_T(trial_square,1,2,n)
    print("calcul with n= ",n,"\nvalue of I :", I_value)
```