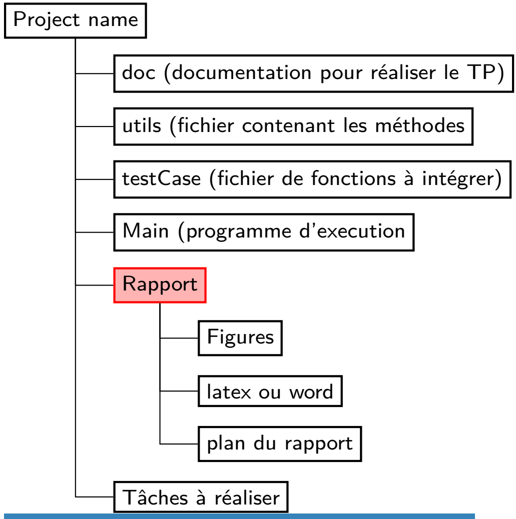
Worksheet for Numerical Integration in Python

Feuille de route pour un TP



Project name

doc (documentation pour réaliser le TP)

utils (fichier contenant les méthodes

testCase (fichier de fonctions à intégrer)

Main (programme d'execution

Rapport

Figures

latex ou word

plan du rapport

Tâches à réaliser

Utils to define integration methods

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 19 18:43:39 2022
@author: H. El-Otmany
@ This file contain all method (RG,RD,MP,S,
@ T,LG2,LG3)
@ Input data :
    - function to integrate f,
    - start value a,
    - end value b,
    - number of subdivision 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
#Formule de Simpson
def method_S(f, a, b, n):
    h = (b-a)/n
    x = np.linspace(a, b-h, n)
    Sx = sum(f(x[1:]))
    Sm = sum(f(x+h/2))
    A = f(a) + f(b) + 2*Sx + 4*Sm
    return h*A/6

#Formule de Gauss à 2 points
def method_LG2(f, a, b, n):
    h = (b-a)/n
    x = np.linspace(a+h/2, b-h/2, n)
    r = h*np.sqrt(3)/6
    coef1 = x-r
    coef2 = x+r
    Sm = sum(f(coef1))
    Sp = sum(f(coef2))
    A = Sm + Sp
    return A*h/2
#Formule de Gauss à 3 points
def method_LG3(f, a, b, n):
    h = (b-a)/n
    x = np.linspace(a+h/2, b-h/2, n)
    r = h*np.sqrt(0.15)
    coef1 = x - r
    coef2 = x - r
    Sm = sum(f(coef1))
    Sx = sum(f(x))
    Sp = sum(f(coef2))
    A = (Sm + Sp)*5 + 8*Sx
    return A*h/18
```

testCase to define test functions

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Feb  8 21:15:05 2022
@ This file contain all test cases
@author: H. El-Otmany
"""
import numpy as np
def trial_f1(x):
    return 1/(1+x**2)
def trial_f2(x):
    return np.exp(x)/x
```

mainProg used for Execution

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 19 18:43:39 2022
@author: Hammou El-Otmany
@ this file used for testing
"""
import sys
import math
import numpy as np
from numpy import inf
import matplotlib.pyplot as plt
from scipy.integrate import quad, simps
from prettytable import PrettyTable
#importing all the functions
from utils import *
from testCase import *
#define variables a, b, n
a = 0
b = 2
n = 100
#Exact value
Iexact= np.arctan(2)
"""
@ Call different method
@ we can use lambda x:1/(1+x**2)
#instead of trial_f1
 """
Irg = method_RG(trial_f1, a, b, n)
Ird = method_RD(trial_f1, a, b, n)
Imp = method_MP(trial_f1, a, b, n)
It = method_T(trial_f1, a, b, n)
Is = method_S(trial_f1, a, b, n)
Ig2 = method_LG2(trial_f1, a, b, n)
Ig3 = method_LG2(trial_f1, a, b, n)
"""
@ Print method value
 """
print("LG3 method:  Ig3 = ", Ig3)
print("RG method:  Irg = ", Irg,'\n',
   "RD method:  Ird = ", Ird,'\n',
   "MP method:  Imp = ", Imp,'\n',
   "Trap. method:  It = ", It,'\n',
   "Simp. method:  Is = ", Is,'\n',
   "LG2 method:  Ig2 = ", Ig2,'\n',
   "LG3 method:  Ig3 = ", Ig3,'\n')
"""
@ Print in table
 """
MTables = PrettyTable()
MTables.field_names = ['Methods', 'Value of I']
MTables.add_row(['RG method', Irg])
MTables.add_row(['RD method', Ird])
MTables.add_row(['MP method', Imp])
MTables.add_row(['Simpson method', Is])
MTables.add_row(['Trapeze method', It])
MTables.add_row(['LG2 method', Ig2])
MTables.add_row(['LG3 method', Ig3])
print(MTables)
"""
@ Compute the integral by using scipy.integrate
 """
h = (b - a)/n
x = np.linspace(a, b, n)
y = trial_f1(x)
It = np.trapz(y,x,h)
print("Trapeze with scipy: It =", It)
"""
@ plotting relative errors
 """
def errors_method(method, f, a, b, I_value, n_li
    number = len(n_liste)
    err_liste = np.empty(number)
    for i in range(len(n_liste)):
        I = method(f, a, b, n_liste[i])
        err_liste[i]=abs((I-Iexact)/Iexact)
    return err_liste
methods={'Rectangles à gauche': method_RG,
          'Rectangles à droite':method_RD,
          'Point milieu':method_MP,
          'Trapèzes':method_T,
          'Simpson':method_S,
          'Gauss Legendre 2':method_LG2,
          'Gauss Legendre 3':method_LG3}

n_liste=list(range(1,50))+list(range(50,200,5))+
    list(range(200,1000,20))
plt.figure()
for m in methods :
    method = methods[m]
    le = errors_method(method, trial_f1, a, b, I
    plt.loglog(n_liste,le,label=m)
plt.legend()
plt.title('Évolution des erreurs relatives en fo
plt.show()
```