Integração Numérica

FCUP
Análise Numérica (M2018) 2018/2019
Trabalho de Grupo 4
Ângelo Gomes – 201703990 – MIERSI
Eduardo Morgado – 201706894 – MIERSI
Simão Cardoso – 201604595 – MIERSI
Sónia Rocha – 201704679 – MIERSI

1. Considerações Iniciais:

Na realização deste trabalho, foi utilizada como linguagem de implementação Python(3.6.8), onde os pontos de virgula flutuante são sempre de precisão dupla.

O programa pode ser encontrado no Github no repositório da equipa, https://github.com/thejoblessducks/Trabalho4AN.git, no entanto, irá ser também aqui apresentado o código para o programa.

Para esta implementação recorreu-se a algumas bibliotecas externas às incluídas no Python para facilitar a resolução do problema, sendo essas bibliotecas: *numpy,matplotlib,sicpy* e *prettytable*.

```
from __future__ import division
import warnings

from scipy.misc import derivative
from scipy.optimize import fminbound as fmin

import matplotlib.pyplot as plt
from prettytable import PrettyTable as PT
from time import localtime, strftime

import numpy as np
import decimal as dm

warnings.filterwarnings("ignore")

'''------------------------------------------------------------------------
Functions: f(x),f'(x) and f^4(x) and error function 10^-x
------------------------------------------------------------------------'''
f = lambda x : (x+np.log( np.sin((x**2)+np.arctan(np.sqrt(1+(x**3))))+1))*x
f_1 = lambda x: derivative(f,x,n=1)
f_4 = lambda x: derivative(f,x,n=4,order=5)

error = lambda x: 10**(-x)


'''------------------------------------------------------------------------
M calculation, max|f'(x)| or max|f^4(x)| for xE]a,b[
------------------------------------------------------------------------'''
def M(lower,upper,simpson=True):
    if simpson:
        return abs(f(fmin(lambda x: -f_4(x),lower,upper)))
    else:
        return abs(f(fmin(lambda x: -f_1(x),lower,upper)))
```

Figura 1-Código

```
'''-------------------------------------------------------------------------
Simpson Rule for Integration Approximation
-------------------------------------------------------------------------'''
#Organize data and display
def simpson(a,b,errors):
    table = PT()
    table.title = "Numerical Approximation for Integration using Simpson Rule"
    table.field_names = ["Error","N","Approximation M="+str(M(a,b))]

    for e in errors:
        s,n = simpsonRule(a,b,e+1)
        table.add_row([str(error(e+1)),str(n),str(dm.Decimal(s))])
    print(table)
    print("\n"*3)

#Determine n for error e
def determineNSimpson(a,b,e):
    m = M(a,b)
    v = (np.power((b-a),5)*m)/(e*180)
    v = np.power(v,(1./4.))
    m = int(v)+2 if int(v)%2==0 else int(v)+1
    return m

#Actual method
def simpsonRule(a,b,e):
    n = determineNSimpson(a,b,error(e))
    val = lambda x: 2 if (i%2==0) else 4
    h = (b-a)/n
    s = f(a)+f(b)
    for i in range(n):
        s += val(i)*f(a+i*h)
    return (h/3)*s,n
```

```
'''-------------------------------------------------------------------------
Rectangle Rule for Integration Approximation
-------------------------------------------------------------------------'''
#Organize data and display
def rectangle(a,b,k,d=True):
    #if d is true,it will show the graph of approximation for every k
    m = M(a,b,simpson=False) #m calculation~

    table = PT()
    table.title = "Numerical Approximation for Integration using Rectangles"
    table.field_names = ["N","Approximation M="+str(m),"Error"]

    for i in range(1,k+1):
        s,e = rectangleRule(a,b,i,m,d);
        er = '%.2E' %dm.Decimal(str(e))
        table.add_row([str(i),str(dm.Decimal(s)),er])
    print(table)
    print("\n"*3)

#Rectangle rule method
def rectangleRule(a,b,n,m,d=True):
    area = lambda c,d: (d-c)*(f(c)+f(d))/2#function for 2 point area

    v = np.linspace(a,b,n)
    s = 0.0
    for i in range(1,len(v)):
        s += area(v[i-1],v[i])
    #error calculation
    e = (np.power((b-a),2)/(n*2))*m

    if d:#draw option
        draw(a,b,n);
    return s,e
```

```
#Draw f and approximation rectangles
def draw(a,b,n):
    v = np.linspace(a,b,n)
    plt.plot(v,f(v),color='red')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Numerical approximation: Rectangular')
    for i in range(1,len(v)):
        c = v[i-1]
        d = v[i]
        plt.plot([c,c],[0,f((c+d)/2)],color='blue')
        plt.plot([d,d],[0,f(d)],color='blue')
        plt.plot([c,d],[f((c+d)/2),f((c+d)/2)],color='blue')
    plt.savefig("./Images/RectangleDraw_N"+str(n)+"_"+strftime("%Y%m%d_%H%M%S", localtime())+".png",dpi=300, bbox_inches="tight")
    plt.cla()
    plt.clf()
    plt.close()
```

```
#-------------------------------------------------------------------------
disp = input("Display Rectangles Graph? (yes/no)")
disp = disp == 'yes'
print
simpson(0,1,[7,12])#apply simpson to error 7 and 12 in [0,1]
rectangle(0,1,20,d=disp) #apply rectangle in [0,1] in n points n=1(1)20
```

Figura 2-Código

$$I = \int_0^1 f(x)\,dx$$

$$f(x) = e^{\left(x+\ln\left(\sin\left(x^2+\arctan\left(\sqrt{1+x^3}\right)\right)+1\right)\right)x}$$

Figura 3-Integral a calcular para f(x)

2. Aproximar pela regra de Simpson:

Pretende-se calcular um valor aproximado de $\int_a^b f(x)dx$, $a = 0, b = 1$, pela regra de Simpson com erro majorado inferior a $e = 10^{-7}$ ou $e = 10^{-12}$ casa decimais corretas, para isso é necessário, antes de aplicar a regra, determinar o valor de n para o número de partições de igual amplitude a fazer em $[a, b]$.

Uma vez que, o erro absoluto da aproximação é $|E_n^S| \leq \frac{h^4}{180}(b-a)M \equiv |E_n^S| \leq \frac{(b-a)^5}{180n^4}M$ onde $M = max|f^4(c)|\ c \in [a,b]$, o valor de n será, $n \leq \sqrt[4]{\frac{(b-a)^5 M}{e*180}}$ onde e será o erro (10^{-7} ou 10^{-12}), o valor de n deverá ser par. Para o integral e $f(x)$ apresentados na *Figura 3* o valor de M será $M = 0.3726153095780505..$ (sendo este valor retornado pelo programa).

A *Tabela 1* apresenta os resultados da execução deste programa (o programa apresenta os resultados em formato *double*, no entanto a tabela apresenta os resultados com 16 casa decimais para facilitar a leitura).

Erro	N	Aproximação
10^{-8}	22	0.6596945515830136..
10^{-13}	380	0.6596950503634053..

Tabela 1-Resultados de aproximação de integral para erros de 10^{-8} e 10^{-13}

3. Aproximar pela regra dos Retângulos:

Pretende-se calcular o valor aproximado de $\int_a^b f(x)dx$, $a = 0, b = 1$, pela regra dos retângulos, fazendo partições do intervalo de integração em $n_k = 2^k$, $k = 1,..,20$ subintervalos de amplitude igual.

A *Tabela 2* apresenta os resultados da execução deste programa (o programa apresenta os resultados em formato *double*, no entanto a tabela apresenta os resultados com 16 casa decimais para facilitar a leitura).

N	Aproximação	Erro
2	0.6408381551074519..	8.67E-02
4	0.6552654309208094..	4.34E-02
8	0.6586061355593629..	2.17E-02
16	0.6594239894325171..	1.08E-02
32	0.6596273583236167..	5.42E-03
64	0.6596781319352013..	2.71E-03
128	0.6596908210467112..	1.35E-03
256	0.659693993056327..	6.77E-04
512	0.6596947860419631..	3.39E-04
1024	0.6596949842873245..	1.69E-04
2048	0.6596950338485997..	8.47E-05
4096	0.6596950462389144..	4.23E-05
8192	0.6596950493364918..	2.12E-05
16384	0.6596950501108882..	1.06E-05
32768	0.6596950503044867..	5.29E-06
65536	0.6596950503528862..	2.65E-06
131072	0.6596950503649871..	1.32E-06
262144	0.6596950503680101..	6.61E-07
524288	0.6596950503687705..	3.31E-07
1048576	0.6596950503689571..	1.65E-07

Tabela 2-Resultados de aproximação do integral por regra dos retângulos

Para este método, $M = max|f'(c)|\ c \in [a,b], M = 0.34681378218322184$

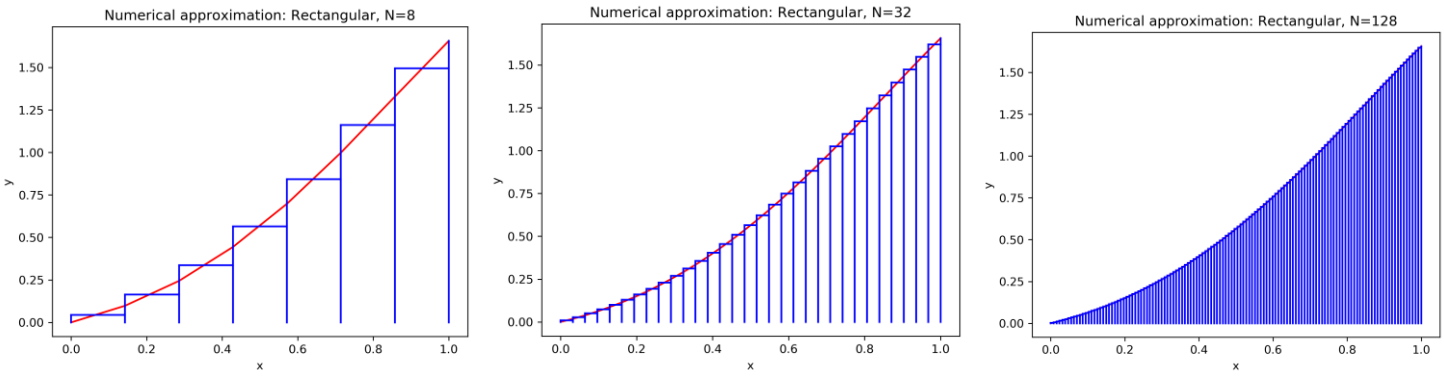

Figura 4- Alguns gráficos das aproximações pela regra de retângulos para diferentes ns

4. Conclusões:

Por observação da *Tabela 2* e da *Figura 4*, a regra dos retângulos demonstra ser uma regra muito lenta, ou seja, são necessários muitos mais ponto para alcançar uma precisão dentro dos parâmetros da regra de Simpson, por exemplo, para a regra dos retângulos atingir uma precisão de ordem próxima à de Simpson para n=22, são necessários 2^{18} pontos.

Através da *Figura 4* pode-se observar a existência do erro para a regra dos retângulos, quanto mais pequeno for o valor de n, maior o erro obtido, uma vez que, os retângulos a considerar irão ter dimensões cada vez maiores, ultrapassando em muitos dos casos, o traço da função, ou seja, com o aumento do n iremos diminuir a área dos retângulos fora da área da função diminuindo assim o erro.

De ambos os métodos apresentados neste relatório, o método/regra que melhor aproxima o valor do integral de uma dada função é a regra de Simpson.