EP3 - Cálculo Numérico

Nicholas Funari Voltani - Nº USP: 9359365

17 de Outubro de 2019

Abaixo seguem as tabelas das rotinas dos exercícios; o exercício 1 foi feito em C++, os exercícios 2 e 3 foram feitos em Python, e os gráficos 1 e 2 foram feitos com o Pyplot. Os respectivos códigos estão nos apêndices.

1 Exercício 1: Método de Simpson

1.1 Precisão Simples

Tabela 1: Valores de $\int_0^1 (6 - 6\,x^5)\,dx$ com método de Simpson, precisão simples ($I_{analitica} = 5$).

p	$N_{tentativas} = 2^p$	I_{num}	Erro $= \lvert I_{num} - I_{analitica} \rvert$
1	2	4.875	0.125
2	4	4.99219	0.0078125
3	8	4.99951	0.000488281
4	16	4.99997	3.05176e-05
5	32	5	1.43051e-06
6	64	5	0
7	128	5	9.53674e-07
8	256	5	1.43051e-06
9	512	5	4.76837e-07
10	1024	5	4.76837e-07
11	2048	5	0
12	4096	5.00001	8.58307e-06
13	8192	5	2.86102e-06
14	16384	5.00001	1.23978e-05
15	32768	5.00005	4.76837e-05
16	65536	5.00012	0.000121593
17	131072	5.00007	6.91414e-05
18	262144	5.00045	0.000452995
19	524288	5.00232	0.00232077
20	1.04858e+06	5.00631	0.00631094
21	2.09715e+06	5.00805	0.0080471
22	4.1943e+06	5.0248	0.0248032
23	8.38861e+06	5.1367	0.136697
24	1.67772e+07	4.81554	0.184465

1.2 Precisão Dupla

Tabela 2: Valores de $\int_0^1 (6 - 6\,x^5)\,dx$ com método de Simpson, precisão dupla ($I_{analitica} = 5$).

p	$N_{tentativas} = 2^p$	I_{num}	Erro $= \lvert I_{num} - I_{analitica} \rvert$
1	2	4.875	0.125
2	4	4.99219	0.0078125
3	8	4.99951	0.000488281
4	16	4.99997	3.05176e-05
5	32	5	1.90735e-06
6	64	5	1.19209e-07
7	128	5	7.45058e-09
8	256	5	4.65661e-10
9	512	5	2.91038e-11
10	1024	5	1.81988e-12
11	2048	5	1.16351e-13
12	4096	5	2.66454e-15
13	8192	5	2.30926e-14
14	16384	5	2.30926e-14
15	32768	5	8.88178e-15
16	65536	5	3.55271e-15
17	131072	5	1.86517e-14
18	262144	5	9.23706e-14
19	524288	5	8.88178e-15
20	1.04858e+06	5	1.35003e-13
21	2.09715e+06	5	2.93987e-13
22	4.1943e+06	5	1.24345e-13
23	8.38861e+06	5	8.79297e-13
24	1.67772e+07	5	4.80505e-13

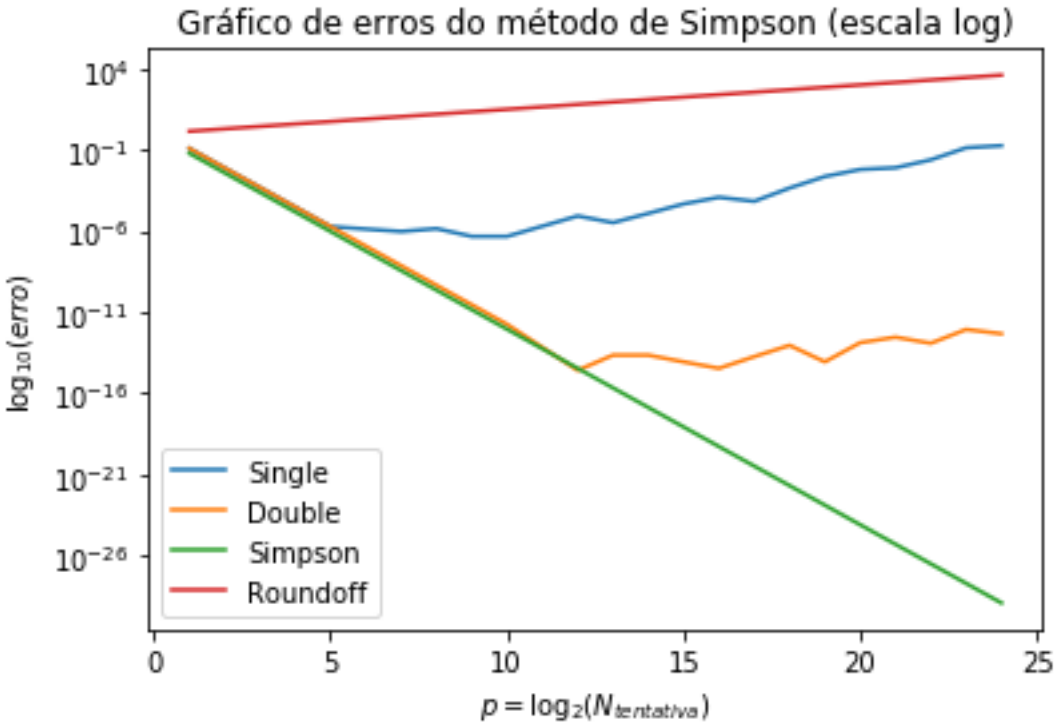

Figura 1: Gráfico dos erros no método de Simpson (em escala logarítmica no eixo vertical).

O gráfico dos erros do método de Simpson tem a forma acima, pois depois de certo número de iterações, o erro de *roundoff* torna-se relevante em relação ao erro inerente ao método ($\mathcal{O}(h^4) = \mathcal{O}(N_{tentativa}^{-4})$), forçando os erros a tenderem assintoticamente ao gráfico dos erros de *roundoff* ($\mathcal{O}\left(\sqrt{N_{tentativa}}\right)$).

2 Exercício 2: Método dos Trapézios

Assumiu-se que $l = 1m$ (comprimento do pêndulo) e $g = 9.81m/s^2$ (para obter os valores da tabela).

Dado θ_0, tem-se que $T = 4\sqrt{\frac{l}{g}} \int_0^{\pi/2} \frac{1}{\sqrt{1-k^2\sin^2(\phi)}} d\phi$, onde $k^2 = \frac{(1-cos(\theta_0))}{2}$.

Tabela 3: Tabela da variação do período T para diferentes ângulos iniciais θ_0 entre 0 e π.

Ângulo Inicial θ_0	Período T
0.0	2.006066680710594
0.15707963267948966	2.0091646679239057
0.3141592653589793	2.0185115351377463
0.47123889803846897	2.0342686110340145
0.6283185307179586	2.0567137595244342
0.7853981633974483	2.086255872614371
0.9424777960769379	2.1234573998662873
1.0995574287564276	2.169067688554444
1.2566370614359172	2.2240716565686345
1.413716694115407	2.2897612190123233
1.5707963267948966	2.367841947576239
1.727875959474386	2.460596725804436
1.8849555921538759	2.571146180414268
2.0420352248333655	2.703882981466589
2.199114857512855	2.865240708892742
2.356194490192345	3.065164746629215
2.5132741228718345	3.320234232385449
2.670353755551324	3.66133681656972
2.827433388230814	4.157352965790445
2.9845130209103035	5.026670340633692

Sabendo que $T_{Galileu} = 2\,\pi\sqrt{\frac{l}{g}}$, podemos analisar como $\frac{T}{T_{Galileu}}$ muda com θ_0:

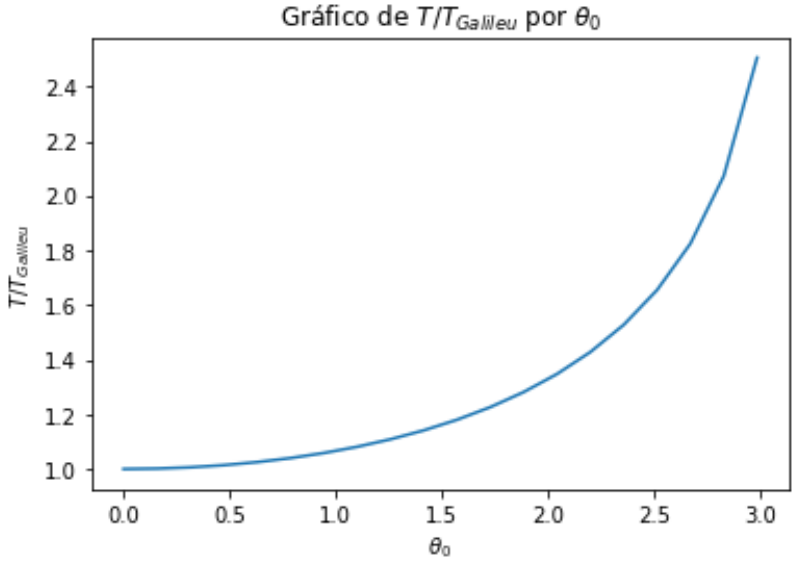

Figura 2: Gráfico de $T/T_{Galileu}$ por θ_0.

3 Exercício 3: Método de Monte Carlo

Tabela 4: Valor de π para N_{tent} tentativas, cada uma com 100 pontos $(x, y) \in (0, 1) \times (0, 1)$.

Tentativas N_{tent}	Valor médio da integral I_m	$\sigma = \frac{1}{\sqrt{N_{tent}-1}}\sqrt{\sum_{i=1}^{N_{tent}}(I_i - I_m)}$	$\sigma_m = \frac{\sigma}{\sqrt{N_{tent}}}$
1	2.92	–	–
2	3.08	0.02720000000000005	0.11661903789690611
4	3.1142857142857143	0.023586394557823118	0.07678931331543329
8	3.146666666666667	0.019555555555555566	0.04944132324730443
16	3.1535483870967744	0.02436439819632325	0.03902274833055974
32	3.13968253968254	0.018587303212802428	0.024100896775847903
64	3.148031496062992	0.02307485265129259	0.018988011288085086
128	3.1466666666666665	0.020932983377077854	0.01278823414836547
256	3.1517025440313113	0.021924180887967888	0.00925426018618585
512	3.1423655913978497	0.024981302952571397	0.006985098949137443
1024	3.1422960429897415	0.026551329387784567	0.005092055832888949
2048	3.1427399267399267	0.027008986159917123	0.003631526059853768
4096	3.1425369307776827	0.026891997257739893	0.0025623093134910357
8192	3.1425599707013365	0.026756660416814133	0.0018072614361339318
16384	3.1429169591357162	0.027146496247197966	0.0012872026414235509
32768	3.1421554894331276	0.02722501289848515	0.0009115050510467034
65536	3.1417299021141214	0.02697135311656084	0.0006415217751065071
131072	3.14164253861442	0.026945946653054907	0.00045341069466232415

A Código: Método de Simpson (exercício 1)

Code 1: 3a) Código para método de Simpson (simples).

```
#include <iostream>
#include<cmath>
using namespace std;

float f_single(float x){ // funcao em precisao single/float
    float y;
    y = 6-6*float(pow(x,5));
    return y;
}

int main(void){
    float I = 5;          // valor analitico da integral
    int p;                // para tamanho dos intervalos
    float N;              // numero de intervalos
    int it;               // para iteracoes de Simpson
    float I_numsingle;    // valores numericos das integrais
    float x_single;
    float h;              // passo do programa
    for(p=1; p<=24;++p){
        N=pow(2,p);
        h = 1/N;
        x_single = 0;
        I_numsingle = (h/3)*f_single(x_single); // primeiro termo igual a 0
        for(it=1;it<N;++it){
            x_single+=h;
            if(it%2==1){                        //y_1, y_3,... (termos "pares", se comecasse em n=1)
                I_numsingle+=(h/3)*4*f_single(x_single);}
            else{                               //y_2,y_4,... (termos "impares", se comecasse em n=1)
                I_numsingle+=(h/3)*2*f_single(x_single);}
        }
        x_single+=h;
        I_numsingle+=(h/3)*f_single(x_single);     // para ultimo ponto da iteracao

        cout<<p;cout<<" & "; cout<<N;cout<<" & ";
        cout<<I_numsingle; cout<<" & ";cout<<abs(I-I_numsingle);cout<<" & \\\\\n";
    }
}
```

B Código: Método dos Trapézios (exercício 2)

Code 2: 3a) Código para o método dos trapézios.

```
import numpy as np
from matplotlib import pyplot as plt

l=1 ## Talvez mudar
g=9.81
T_Galileu=2*np.pi*np.sqrt(l/g)
## Para analise grafica
x=[]
y=[]

def k(theta): ## k do codigo <=> k^2 do enunciado
    return (1-np.cos(theta))/2

def f(theta,phi):
    return 4*np.sqrt(l/g)*(1/np.sqrt(1-k(theta)*np.sin(phi)**2))

N=1500 ## Numero de pontos no intervalo (0,pi/2) (No de Trapezios = N-1)
h=(np.pi/2 - 0)/N ## Passo

print("Integracao por Metodo dos Trapezios\n")
print("\\theta_0 & T\\\\\n") ## Prints para a tabela em Latex, no EP
## Para 20 theta_0s, entre 0 e pi
for n in range(0,20):
    theta_0 = n*np.pi/20
    ## Primeiro termo da soma
    phi=0
    T = (h/2)*f(theta_0,phi)
    ## Termos do meio da soma
    for i in range(1,N):
        phi += h
        T += h*f(theta_0,phi)
    ## Ultimo termo da soma
    phi+=h
    T += (h/2)*f(theta_0,phi)
    print("{} & {}\\\\\n".format(theta_0,T))
    ## Para analise grafica
    x.append(theta_0)
    y.append(T/T_Galileu)

## Analise grafica
plt.xlabel("$\\theta_0$")
plt.ylabel("$T/T_{Galileu}$")
plt.title("Gr fico de $T/T_{Galileu}$ por $\\theta_0$")
plt.plot(x,y)
```

C Código: Método de Monte-Carlo (exercício 3)

Code 3: 3a) Código para o método de Monte Carlo.

```
import numpy as np
def random(Zn):
    a=16807
    m = 2147483647
    Z=(a*Zn)%m
    return Z
## U_i = random(Z_i)/m
## U_i o valor pseudo-aleatorio
m = 2147483647
Zn = 9359365
x=[]
y=[]
I=[]
## Letra b) Primeira tentativa
## (ou p=0, p do proximo for)
for k in range(200):
    Zn=random(Zn) ## esta em Z_{n+1}
    Un=Zn/m       ## esta em U_{n+1}
    if k%2==0:
        x.append(Un)
    else:
        y.append(Un)
Nin=0
Nout=0
for k in range(len(x)):
    if x[k]**2+y[k]**2<1:
        Nin+=1
    else:
        Nout+=1
I.append(4*Nin/(Nin+Nout))
print("Primeira tentativa: Im = {}".format(I[0]))

## Letra c) Tentativas
for n in range(1,18):
    Ntent=2**n
    for tent in range(0,Ntent): ## quantas tentativas serao feitas
        x=[]
        y=[]
        for i in range(0,200): ## 100 pontos x, 100 pontos y
            Zn = random(Zn)
            Un = Zn/m
            if i%2==0:
                x.append(Un)
            else:
                y.append(Un)
        Nin=0
        Nout=0
        ## contando pontos dentro e fora do semicirculo
        for p in range(len(x)):
            if x[p]**2+y[p]**2<1:
                Nin+=1
            else:
                Nout+=1
        I.append(4*Nin/(Nin+Nout))

    Im=np.mean(I)
    sigma2=0
    for l in range(Ntent):
        sigma2+=(1/(Ntent-1))*(I[l]-Im)**2
    sigmam = np.sqrt(sigma2/Ntent)
    print("{} & {} & {} & {}\\\\".format(Ntent,Im,sigma2,sigmam))
```