Universidade Federal do Amazonas

Instituto de Ciências Exatas

Departamento de Física

Integração Numérica via algoritmo de Monte-Carlo

Relatório para a disciplina de Cálculo Numérico

por

Aluno: Micael Davi Lima de Oliveira

Orientador: Prof. Dr. José Francisco de Magalhães Neto

Manaus, Dezembro de 2019

Micael Davi Lima de Oliveira

Integração Numérica via algoritmo de Monte-Carlo

Relatório como requisito de nota parcial para a disciplina de Cálculo Numérico apresentado durante o 4º Período pelo discente de matrícula *21851626* da turma *FB01* ao Curso de Bacharelado em Física pela Universidade Federal do Amazonas.

Orientador: Prof. Dr. José Francisco de Magalhães Neto

Manaus, AM

2019

Agradecimentos

Agradeço muito a Deus, por me dar forças e esperança em viver. Ainda que eu não mereça tamanho amor, me fortalece nos dias de tristeza.

Aos meus pais, pelo carinho e apoio.

Ao professor José Reginaldo, cujo conhecimento transmitido foi imprescindível para a realização deste trabalho.

Lista de Figuras

Figura 1 Saída dado pelo algoritmo à solução de $\int_0^\pi \sin(x)dx$. 5

Figura 2 Saída gerada pelo algoritmo num intervalo de 1 hora após o primeiro teste. 6

Figura 3 Saída gerada pelo algoritmo mediante 1 milhão de pontos aleatórios. 7

Figura 4 Saída correspondente à solução aproximada da Integral de Fresnel. 7

Figura 5 Solução aproximada à integral acima no intervalo de $[0, \pi/2]$. 8

Sumário

1 **ANÁLISE DOS TESTES EFETUADOS NO ALGORITMO** 5

2 **APÊNDICE A** 9

2.1 Algoritmo da integração numérica de Monte-Carlo 9

1 Análise dos testes efetuados no algoritmo

Primeiramente, solucionou-se como caso de teste o exemplo da integral definida $\int_0^\pi \sin(x)dx$ mediante a utilização de 200 pontos aleatórios. O algoritmo da função *random* é interno à biblioteca *numpy*. O compilador adotado foi o servidor do Google Colaboratory Research em linguagem Python 3. Deve-se destacar que o algoritmo não é de autoria própria, todos os créditos são reservados a *Michael Guerzhoy*, professor alocado no Center for Statistics and Machine Learning at Princeton University.

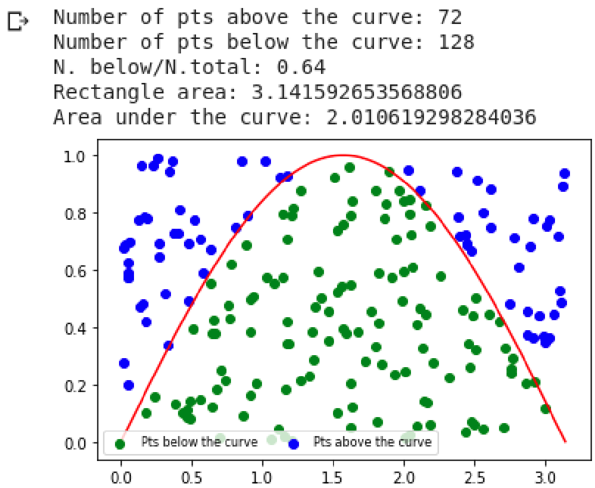

Figura 1: *Saída dado pelo algoritmo à solução de* $\int_0^\pi \sin(x)dx$.

Após a solução numérica mostrada acima, será apresentado a função integral mediante os métodos do Cálculo Diferencial e Integral.

$$\int_0^\pi \sin(x)dx = [-\cos(x) + C]_0^\pi \tag{1.1}$$

$$\int_0^\pi \sin(x)dx = (-\cos(\pi) + C) - (-\cos(0) + C) \tag{1.2}$$

$$\int_0^\pi \sin(x)dx = (1 + C) - (-1 + C) = 2 \tag{1.3}$$

E portanto, o erro encontrado ao método de Monte-Carlo foi de:

$$E(\%) \approx (2 - 2.0106)/(2) \approx -0.5300\% \tag{1.4}$$

A mesma função integrante foi solucionada novamente com 200 pontos aleatórios cerca de 1 hora após a primeira saída.

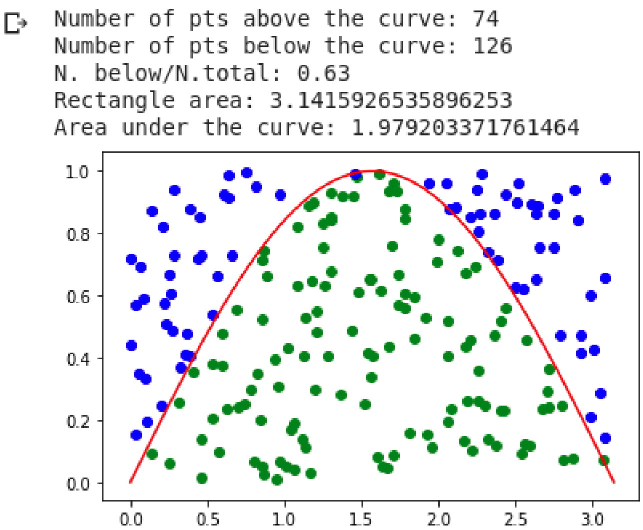

Figura 2: *Saída gerada pelo algoritmo num intervalo de 1 hora após o primeiro teste.*

Contata-se então, uma geração aleatória de pontos, e que para intervalos de tempo distintos obtém-se resultados estatisticamente independentes entre si, caracterizando-se uma função aleatória de números. Novamente, será calculado o erro percentual do resultado:

$$E(\%) \approx (2 - 1.9792)/(2) \approx -1.0400\% \tag{1.5}$$

O próximo teste consistirá em solucionar a mesma função, desta vez, com um total de 1 milhão de pontos gerados aleatoriamente. O objetivo será entender quais parâmetros influenciam de forma significativa no resultado obtido pelo algoritmo.

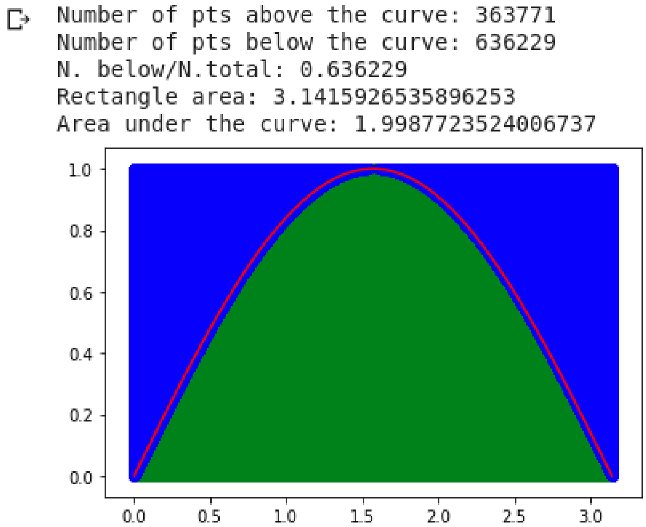

Figura 3: *Saída gerada pelo algoritmo mediante 1 milhão de pontos aleatórios.*

Por fim, será analisado o erro percentual gerado pelo algoritmo.

$$E(\%) \approx (2 - 1.9988)/(2) \approx -0.0600\% \tag{1.6}$$

Percebe-se então, que aparentemente quanto maior for o número de pontos gerados aleatoriamente, aumenta-se a probabilidade de obter-se um resultado de maior precisão. Outro teste a ser executado no algoritmo, será a solução numérica aproximada da integração de uma função que não possui solução exata no conjunto dos números Reais.

$$\int_0^{2\pi} \sin(x^2)dx = ? \tag{1.7}$$

(Integral de Fresnel)

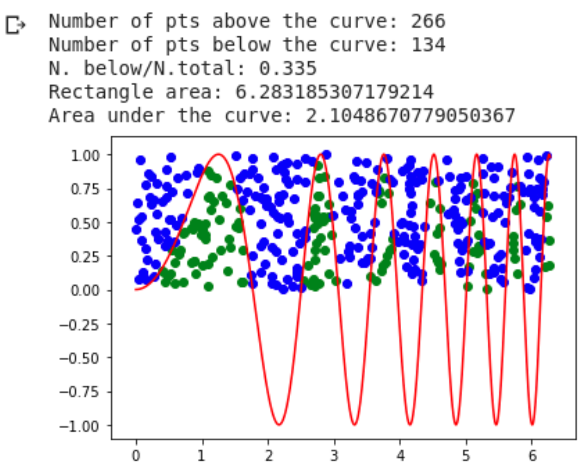

Figura 4: *Saída correspondente à solução aproximada da Integral de Fresnel.*

A integral solucionada não possui valor exato dentro do conjunto dos números Reais. Todavia, mediante o método de integração numérica de Monte-Carlo foi possível obter uma solução aproximada. Desta forma, torna-se impossível calcular o erro do resultado. Sobretudo, percebeu-se uma certa oscilação nos resultados de saída, variando entre ≈ 2.0539 e ≈ 2.3373.

É importante ressaltar que o método de Monte-Carlo obedece a uma distribuição de probabilidade, conhecida como distribuição Normal ou Gaussiana. Desta forma, ainda que os pontos sejam gerados aleatoriamente, os resultados de saída possuem um certo grau de probabilidade de aparecerem, de tal maneira, que o valor mais próximo do resultado real tende a ter um maior número de ocorrências. Portanto, ao construir um histograma, a média limite seria a solução aproximada da integral definida.

O último teste a ser realizado, consistirá em solucionar via análise numérica uma integral definida que possui uma dificuldade muito grande para obter a função integral exata.

$$\int_0^{\pi/2} \frac{x \cdot \cos(x)}{1+\sin^2(x)} dx = \frac{ln^A(\sqrt{B}+C)}{D} \tag{1.8}$$

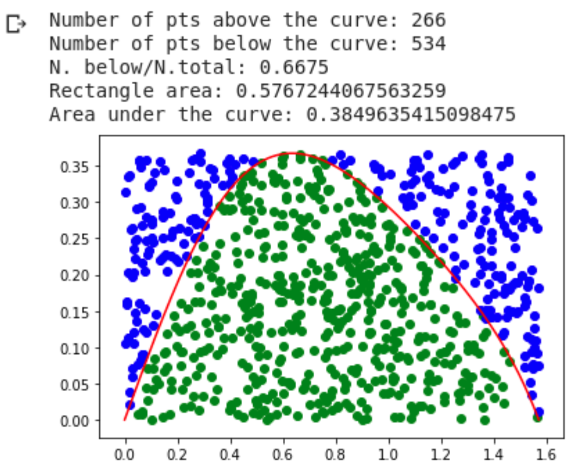

Figura 5: *Solução aproximada à integral acima no intervalo de* $[0, \pi/2]$.

Sendo assim, percebe-se a tamanha importância dos métodos de análise numérica na solução de inúmeros problemas que possuem uma solução exata de difícil determinação, ou mesmo, limitada pelos atuais conhecimentos em matemática pura e aplicada. O método de Monte-Carlo, pode ser inclusive utilizado em pesquisas teóricas de novos fármacos.

2 Apêndice A

2.1 Algoritmo da integração numérica de Monte-Carlo

O algoritmo realizado não é de minha autoria, foi escrito pelo professor Michael Guerzhoy da Universidade de Princeton. Contudo, algumas sutis adaptações foram realizadas por mim.

Código-fonte original:

http://www.cs.toronto.edu/~guerzhoy/

Código-fonte adaptado:

https://colab.research.google.com/drive/1XWlT0iD2pw-21Pdi53Tbz8QOL4tgl_FX

```python
# This algorithm it' not my own, the truth author can be found at:
# http://www.cs.toronto.edu/~guerzhoy/180/lectures/W10/lec2/MonteCarloInt.html

# Algoritmo para a disciplina de Cálculo Numérico 2019/2 lecionada pelo professor
# José Reginaldo doscente do Instituto de Computação da Universidade Federal do
# Amazonas.

# Aluno: Micael Davi Lima de Oliveira
# Matrícula: 21851626
# Turma: FB01

import numpy as np
```

```
import matplotlib.pyplot as plt

def f1(x):
    '''Return sqrt(1-x**2). If x is an array, perform the operation
    elementwise (whence the np.sqrt)'''
    return (np.sin(x))

def definite_integral_show(f, x0, x1, N):
    """Approximate the definite integral of f(x)dx between x0 and x1
   using
    N random points

    Arguments:
    f -- a function of one real variable, must be nonnegative on [x0, x1
   ]
    N -- the number of random points to use


    """
    #First, let's compute fmax. We do that by evaluating f(x) on a grid
    #of points between x0 and x1
    #This assumes that f is generally smooth. If it's not, we're in
   trouble!
    x = np.arange(x0, x1, 0.000001)
    y = f(x)
    f_max = max(y)


    #Now, let's generate the random points. The x's should be between
    #x0 and x1, so we first create points beterrm 0 and (x1-x0), and
    #then add x0
    #The y's should be between 0 and fmax
    #
    #                  0...(x1-x0)
    x_rand = x0 + np.random.random(N)*(x1-x0)
    y_rand = 0 +  np.random.random(N)*f_max

    #Now, let's find the indices of the poitns above and below
    #the curve. That is, for points below the curve, let's find
```

```python
    #   i s.t. y_rand[i] < f(x_rand)[i]
    #And for points above the curve, find
    #   i s.t. y_rand[i] >= f(x_rand)[i]
    ind_below = np.where(y_rand < f(x_rand))
    ind_above = np.where(y_rand >= f(x_rand))

    #Finally, let's display the results
    plt.plot(x, y, color = "red")
    plt.scatter(x_rand[ind_below], y_rand[ind_below], color = "green")
    plt.scatter(x_rand[ind_above], y_rand[ind_above], color = "blue")

    print("Number of pts above the curve:", len(ind_above[0]))
    print("Number of pts below the curve:", len(ind_below[0]))
    print("N. below/N.total:", len(ind_below[0])/N)
    print("Rectangle area:", f_max*(x1-x0))
    print("Area under the curve:", f_max*(x1-x0)*len(ind_below[0])/N)

definite_integral_show(f1, 0, 3.1415926535897932384, 800)
```

Listing 2.1: *(Algoritmo adaptado) Implementação na linguagem Python 3 do método de integração numérica via Monte-Carlo.*

Bibliografia

[1] Guerzroy, Michael. Monte Carlo Integration. Acesso em: 01 de dezembro de 2019. Disponível em: http://www.cs.toronto.edu/~guerzhoy/