UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto

Métodos Computacionais

Curso de Física Biológica.

Relatório 3:

Integração por Método de Monte Carlo

Discentes:
André Luís Dias
Diego Henrique Conchal

Docentes:
Prof. Dr. Davi Rubinho Ratero
Monitor Murilo Nogueira Sanches

30 de Junho, 2023

Sumário

Figura 1 – Código do executado no SageMath.

```
import numpy as np
import matplotlib.pyplot as plt
import random
from math import e
from sympy import limit, Symbol, oo

def ex1(x):
    return np.sqrt(1 - x**2)

def val_ex1():
    cod = 1
    a = 1
    b = 0
    n = 10**4
    y = [0, 1]
    print("Exercício - 1)\n")
    print("Valor esperado = 3.141593")

    return a, b, y, n, cod

def ex2(x):
    return x**2

def val_ex2():
    cod = 2
    a = 1
    b = 0
    n = 10**4
    y = [0, 1]
    print("Exercício - 2)\n")
    print("Valor esperado = 0.3333")

    return a, b, y, n, cod

def ex3(x):
    return x**3

def val_ex3():
    cod = 3
    a = 2
    b = 0
    n = 10**4
    y = [0, 8]
    print("Exercício - 3)\n")
    print("Valor esperado = 4")

    return a, b, y, n, cod


def ex4(x):
    return 1/x

def val_ex4():
    cod = 4
    a = e
    b = 1
    n = 10**4
    y = [0, 1 ]
    print("Exercício - 4)\n")
    print("Valor esperado = 1")
```

Fonte: Autoria própria.

Figura 2 – Código do executado no SageMath.

```
    return a, b, y, n, cod

def ex5(x):
    return np.sin(x)

def val_ex5():
    cod = 5
    a = pi
    b = 1
    n = 10**4
    y = [0, 1]
    print("Exercício - 5)\n")
    print('Valor esperado = ',float(1+cos(1)))

    return a, b, y, n, cod

def ex6(x):
    return 1/(1+x**2)

def val_ex6():
    cod = 6 # Retorna o codigo da Eq.
    a = 1
    b = 0
    n = 10**4
    y = [0, 1]
    print("Exercício - 6)\n")
    print("Valor esperado = 0.785")

    return a, b, y, n, cod

def ex7(x):
    return 1 / np.sqrt(1 - np.cos(x))


def val_ex7():
    cod = 7
    a = 2*(np.pi)
    b = 0
    n = 10**4
    y = [0, 800]
    print("Exercício - 7)\n")
    print("Valor esperado = 50.0")

    return a, b, y, n, cod


def ex8(x):
    return 1/(x**2+1)

def val_ex8():
    cod = 8 #
    a = 42
    b = 0
    n = 10**4
    y = [0, 1]
    print("Exercício - 8)\n")
    print("Valor esperado = 1.570797")

    return a, b, y, n, cod

def ex9(x):
```

Fonte: Autoria própria.

Figura 3 – Código do executado no SageMath.

```
    return e^x/ (1+e^x)

def val_ex9():
    cod = 9
    a = 1
    b = -1
    n = 10**4
    y = [0, 0.731059]
    print("Exercício - 9)\n")
    print("Valor esperado = 1")
    return a, b, y, n, cod

def ex10(x):
    return 1/sqrt(x)

def val_ex10():
    cod = 10
    a = 1
    b = 0
    n = 10**4
    y = [0, 10]
    print("Exercício - 10)\n")
    print("Valor esperado = 2")
    return a, b, y, n, cod

def nome_eq(num):
    x = Symbol('x')
    equação = ["f(x)=x**2",
               "f(x)=x**3",
               "f(x)=1/x",
               "f(x)=np.sin(x)",
               "f(x)=1/(1+x**2)",
               "f(x)=1/np.sqrt(1-np.cos(x))",
               "f(x)=1/(x**2+1)",
               "f(x)=e^x/ (1+e^x)",
               "f(x)=1/sqrt(x)"]

    return equação[num-2]


def calculadora(a, b, ym, Eq, y1, y2, cod):
    XD = np.linspace(float(a), b, 100)
    plt.figure()
    plt.plot(XD,Eq(XD))
    teste = 0
    underworld = 0
    for i in range(n):
        x = random.uniform(a, b)
        y = random.uniform(y1, y2)
        teste += 1
        print(str('Contador de interações ') + str(teste), end='\r')

        if y <= Eq(x):
            underworld = underworld + 1
            plt.scatter(x, y, color='b', s=2, alpha=0.5)
        else:
            plt.scatter(x, y, color='r', s=2, alpha=0.5)

    if cod == 1.0:
        solv = 4*(underworld/n) * (a - b) * (y2 - y1)
    else:
```

Fonte: Autoria própria.

Figura 4 – Código do executado no SageMath.

```
        solv = (underworld/n) * (a - b) * (y2 - y1)

    print("\nValor calculado =",float(solv))

    plt.axhline(0, color='black', linewidth=0.5)
    plt.axvline(0, color='black', linewidth=0.5)

    plt.plot(XD,Eq(XD),color='g',linewidth=0.9)
    plt.plot(x, y,color='green', label=nome_eq(cod))
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.title("Gráfico do método de Monte Carlo")
    plt.legend()
    plt.show()
    return



a, b, ym, n, cod = val_ex1()
Eq = ex1
y1, y2 = ym[0], ym[1]
calculadora(a, b, ym, Eq, y1, y2, cod)

a, b, ym, n, cod = val_ex2()
Eq = ex2
y1, y2 = ym[0], ym[1]
calculadora(a, b, ym, Eq, y1, y2, cod)

a, b, ym, n, cod = val_ex3()
Eq = ex3
y1, y2 = ym[0], ym[1]
calculadora(a, b, ym, Eq, y1, y2, cod)

a, b, ym, n, cod = val_ex4()
Eq = ex4
y1, y2 = ym[0], ym[1]
calculadora(a, b, ym, Eq, y1, y2, cod)

a, b, ym, n, cod = val_ex5()
Eq = ex5
y1, y2 = ym[0], ym[1]
calculadora(a, b, ym, Eq, y1, y2, cod)

a, b, ym, n, cod = val_ex6()
Eq = ex6
y1, y2 = ym[0], ym[1]
calculadora(a, b, ym, Eq, y1, y2, cod)

a, b, ym, n, cod = val_ex7()
Eq = ex7
y1, y2 = ym[0], ym[1]
calculadora(a, b, ym, Eq, y1, y2, cod)

a, b, ym, n, cod = val_ex8()
Eq = ex8
y1, y2 = ym[0], ym[1]
calculadora(a, b, ym, Eq, y1, y2, cod)

a, b, ym, n, cod = val_ex9()
Eq = ex9
y1, y2 = ym[0], ym[1]
```

Fonte: Autoria própria.

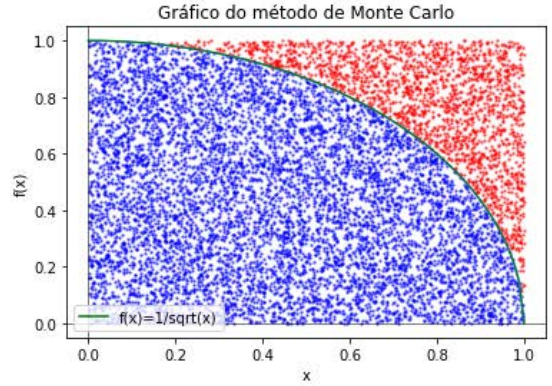
Figura 5 – Código do executado no SageMath.

```
calculadora(a, b, ym, Eq, y1, y2, cod)

a, b, ym, n, cod = val_ex10()
Eq = ex10
y1, y2 = ym[0], ym[1]
calculadora(a, b, ym, Eq, y1, y2, cod)
```

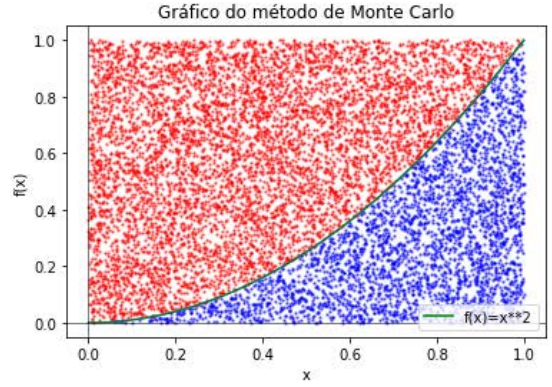
Fonte: Autoria própria.

Figura 6 – Código do executado no SageMath.

```
Exercício - 1)

Valor esperado = 3.141593
Contador de interações 10000
Valor calculado = 3.1524
```



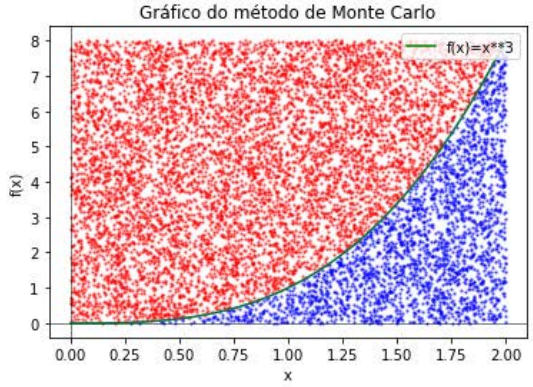

```
Exercício - 2)

Valor esperado = 0.3333
Contador de interações 10000
Valor calculado = 0.3301
```



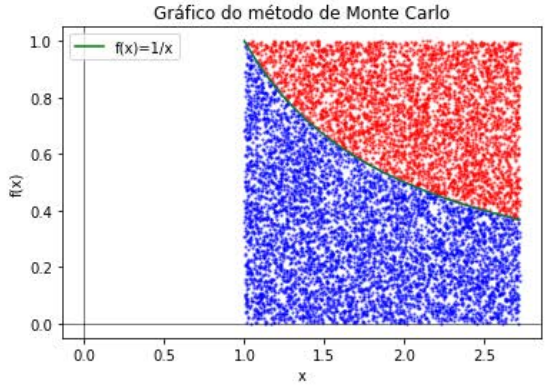

```
Exercício - 3)

Valor esperado = 4
Contador de interações 10000
Valor calculado = 4.0144
```

Fonte: Autoria própria.

Figura 7 – Código do executado no SageMath.



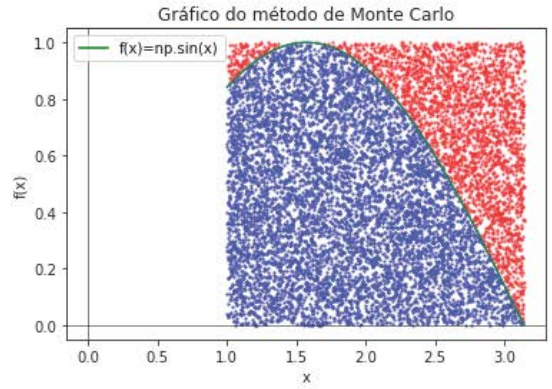

```
Exercício - 4)

Valor esperado = 1
Contador de interações 10000
Valor calculado = 0.9899021613752558
```



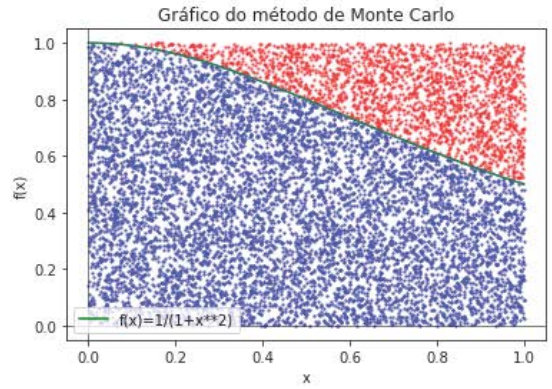

```
Exercício - 5)

Valor esperado =  1.5403023058681398
Contador de interações 10000
Valor calculado = 1.5387343216042662
```

Fonte: Autoria própria.

Figura 8 – Código do executado no SageMath.



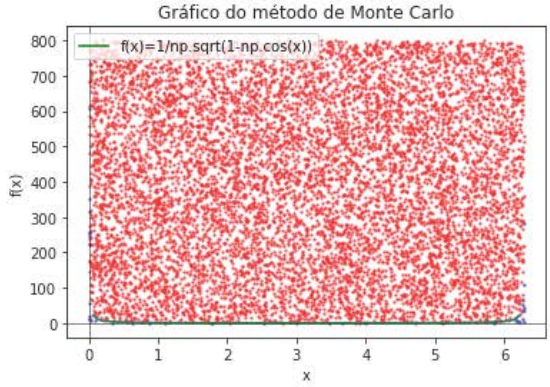

```
Exercício - 6)

Valor esperado = 0.785
Contador de interações 10000
Valor calculado = 0.7789
```



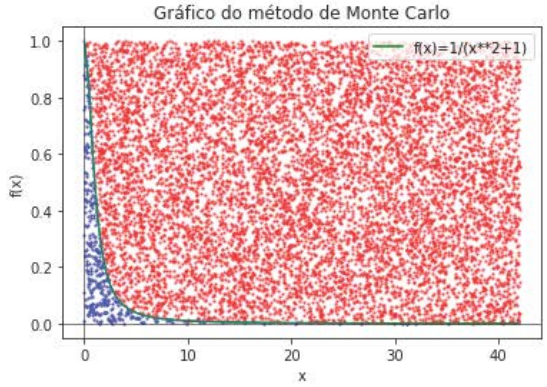

```
Exercício - 7)

Valor esperado = 50.0
Contador de interações 77

/opt/sagemath-9.3/local/lib/python3.7/site-packages/sage/repl/ipython_kern
el/__main__.py:93: RuntimeWarning: divide by zero encountered in true_divi
de

Contador de interações 10000
Valor calculado = 19.60353815840031

/opt/sagemath-9.3/local/lib/python3.7/site-packages/sage/repl/ipython_kern
el/__main__.py:93: RuntimeWarning: divide by zero encountered in true_divi
de
```

Fonte: Autoria própria.

Figura 9 – Código do executado no SageMath.



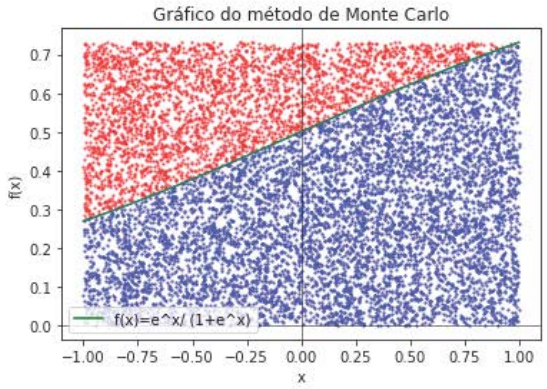

```
Exercício - 8)

Valor esperado = 1.570797
Contador de interações 10000
Valor calculado = 1.5414
```



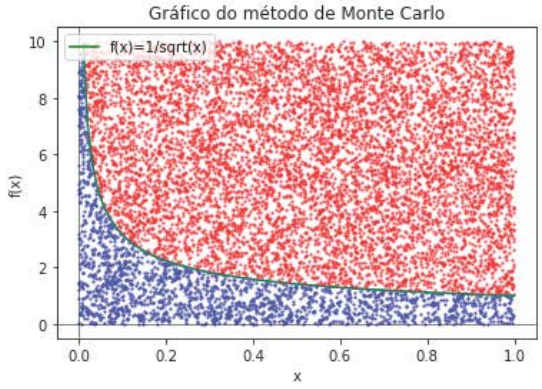

```
Exercício - 9)

Valor esperado = 1
Contador de interações 10000
Valor calculado = 1.0006735592
```

Fonte: Autoria própria.

Figura 10 – Código do executado no SageMath.




```
Exercício - 10)

Valor esperado = 2
Contador de interações 82

/opt/sagemath-9.3/local/lib/python3.7/site-packages/sage/repl/ipython_kern
el/__main__.py:136: RuntimeWarning: divide by zero encountered in true_div
ide

Contador de interações 10000
Valor calculado = 1.917

/opt/sagemath-9.3/local/lib/python3.7/site-packages/sage/repl/ipython_kern
el/__main__.py:136: RuntimeWarning: divide by zero encountered in true_div
ide
```



Fonte: Autoria própria.

1 Nota

No *exercício* 7), os valores não foram compatíveis com os valores esperados, ficando, dessa forma, próximos.

No *exercício* 8), a integral definida de $[0, \infty]$ foi subitamente substituída por uma integral de $[0, 42]$, pois o programa apresentava um erro quando o intervalo tendia ao infinito. Sendo assim, ao aproximar o valor do número para 42, verificou-se que o resultado era, bem próximo do esperado.

Referências

[1] Como funciona o Método de Monte Carlo - Uma visão intuitiva. Disponível em: <https://youtu.be/6Rvl55q8QCY>.

[2] Programando Cálculo do Número Pi com Python - MÉTODO DE MONTE CARLO. Disponível em: <https://youtu.be/RClkUTj09kI>. Acesso em: 30 jun. 2023.

[3] Monte Carlo Integration with Python montecarlosimulation integration Python. Disponível em: <https://youtu.be/qO4BCt67v80>. Acesso em: 30 jun. 2023.

[4] NumPy - Thematic Tutorials. Disponível em: <https://doc.sagemath.org/html/en/thematic_tutorials/numerical_sage/numpy.html>.

[5] Plotting - Constructions. Disponível em: <https://doc.sagemath.org/html/en/constructions/plotting.html>.

[6] Random Numbers with Python API - Utilities. Disponível em: <https://doc.sagemath.org/html/en/reference/misc/sage/misc/prandom.html>. Acesso em: 30 jun. 2023.

[7] STACK OVERFLOW. Stack Overflow - Where Developers Learn, Share, Build Careers. Disponível em: <https://stackoverflow.com/>.

[8] Solucionador matemático Symbolab - calculadora passo a passo. Disponível em: <https://pt.symbolab.com/>.

[9] SCHERFGEN, D. Integral Calculator • With Steps! Disponível em: <https://www.integral-calculator.com/>.