Monte Carlo: Como Calcular Integrais

Nome: Arthur Pontes Nader - Matrícula: 2019022294

Funções auxiliares

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import random

np.random.seed(131)
```

In [ ]:
```
def mostrarGrafico(f, limite_esq, limite_dir):

    x = np.linspace(limite_esq,limite_dir)

    plt.figure(figsize=(14, 8))
    plt.xlabel("x", fontsize=18)
    plt.ylabel("y", fontsize=18)
    plt.ylim(0, max(max(f(limite_esq),f(limite_dir)), f((limite_dir-limite_esq)/2)) + 0.
    plt.title("Gráfico da função analisada", fontsize = 22)
    plt.plot(x, f(x))
    plt.grid()
    plt.show()
```

In [ ]:
```
def gerarHistograma(valores, n_bins, intervalo, titulo):

  fig, ax = plt.subplots(figsize =(14, 8))
  ax.hist(valores, bins = n_bins, range = intervalo, rwidth=0.9)
  plt.title(titulo, fontsize=18)
  plt.show()
```

In [ ]:
```
def calcularEstatisticas(dados):

    print("Média: " + str(np.mean(dados)))
    print("Desvio Padrão: " + str(np.std(dados)))
    print("Erro: " + str(np.std(dados)/np.sqrt(len(dados))))
```

Método 1

In [ ]:
```
def monteCarlo1(f, limite_esq, limite_dir, y_max, N):

    resultados = []

    for _ in range(1000):
        abaixo_curva = 0
        for _ in range(N):
            valor_x = np.random.uniform(limite_esq, limite_dir)
            valor_y = np.random.uniform(0, y_max)

            if valor_y <= f(valor_x):
                abaixo_curva +=1

        resultados.append(abaixo_curva/N * ((limite_dir - limite_esq)*y_max))
```

```
    return resultados
```

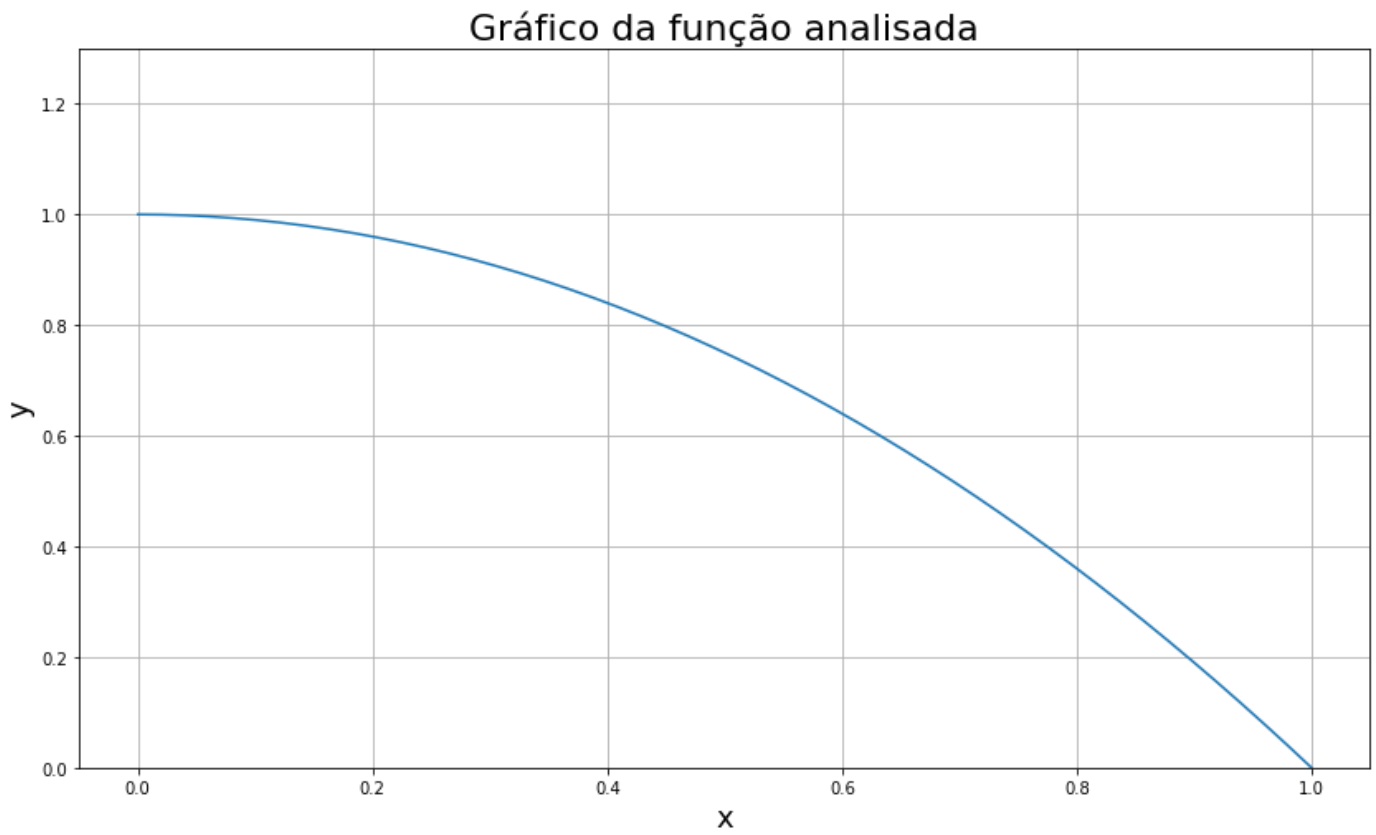
Método 2

In [ ]:
```
def monteCarlo2(f, limite_esq, limite_dir, N):

    resultados = []
    media = (limite_dir - limite_esq)/N

    for _ in range(1000):
        somatorio_y = 0
        for _ in range(N):
            valor_x = np.random.uniform(limite_esq, limite_dir)
            somatorio_y += f(valor_x)


        resultados.append(media*somatorio_y)

    return resultados
```

1 -

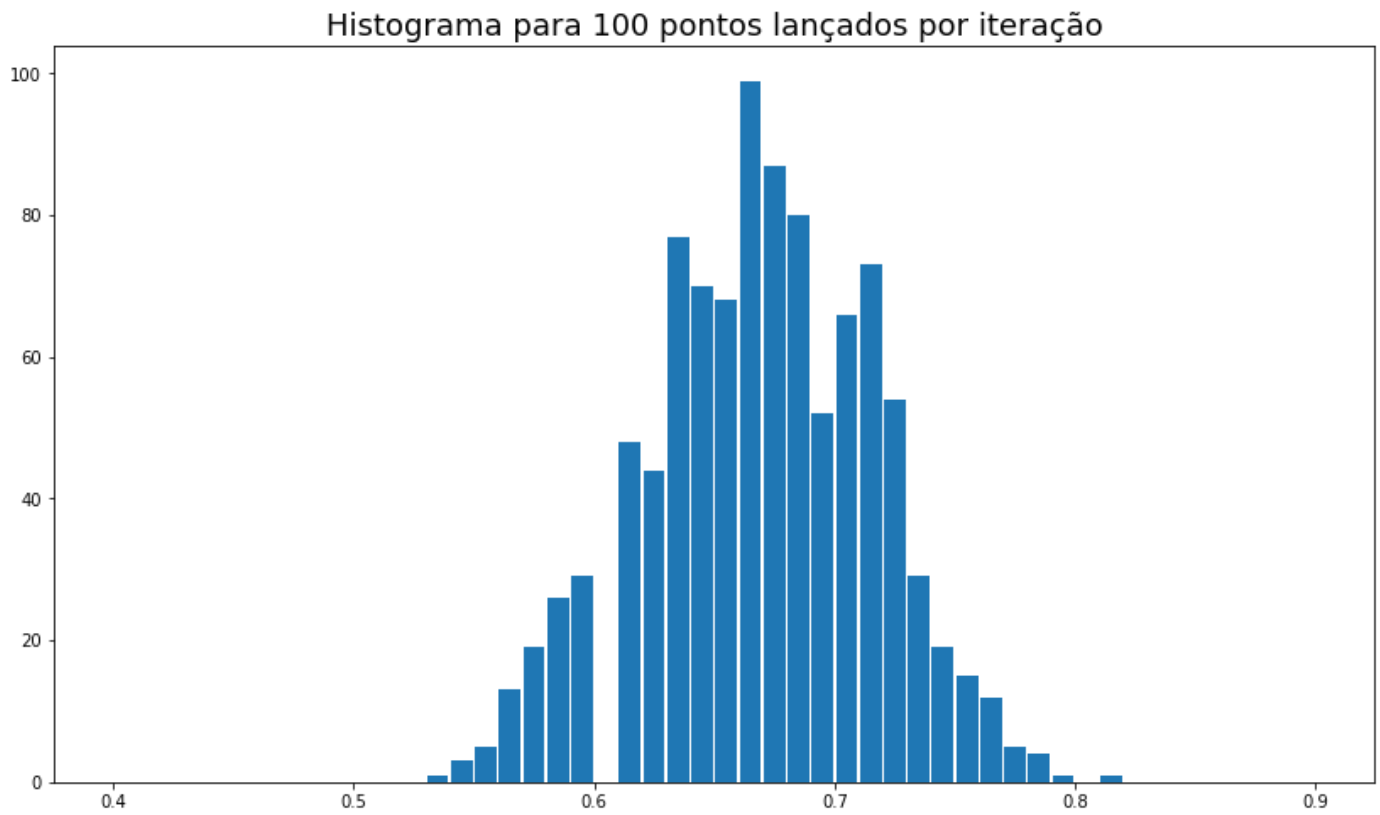
In [ ]:
```
def funcao1(x):
    return 1 - x*x
```

In [ ]:
```
mostrarGrafico(funcao1, 0, 1)
```



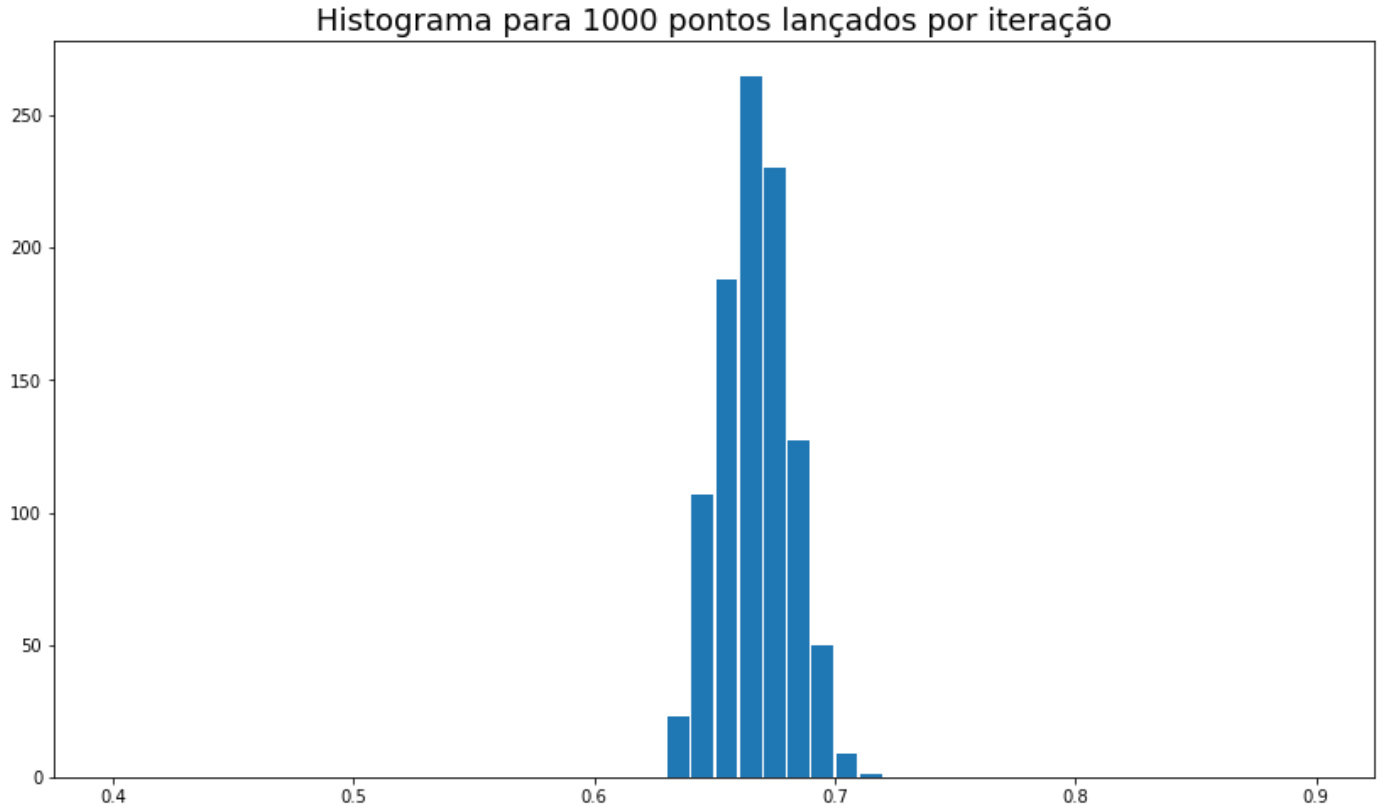
Utilizando o método 1:

In [ ]:
```
resultados_1_100 = monteCarlo1(funcao1, 0, 1, funcao1(0), 100)
gerarHistograma(resultados_1_100, 50, [0.4,0.9], "Histograma para 100 pontos lançados po
calcularEstatisticas(resultados_1_100)
```




```
Média: 0.66509
Desvio Padrão: 0.04569673839564482
Erro: 0.0014450577497110626
```

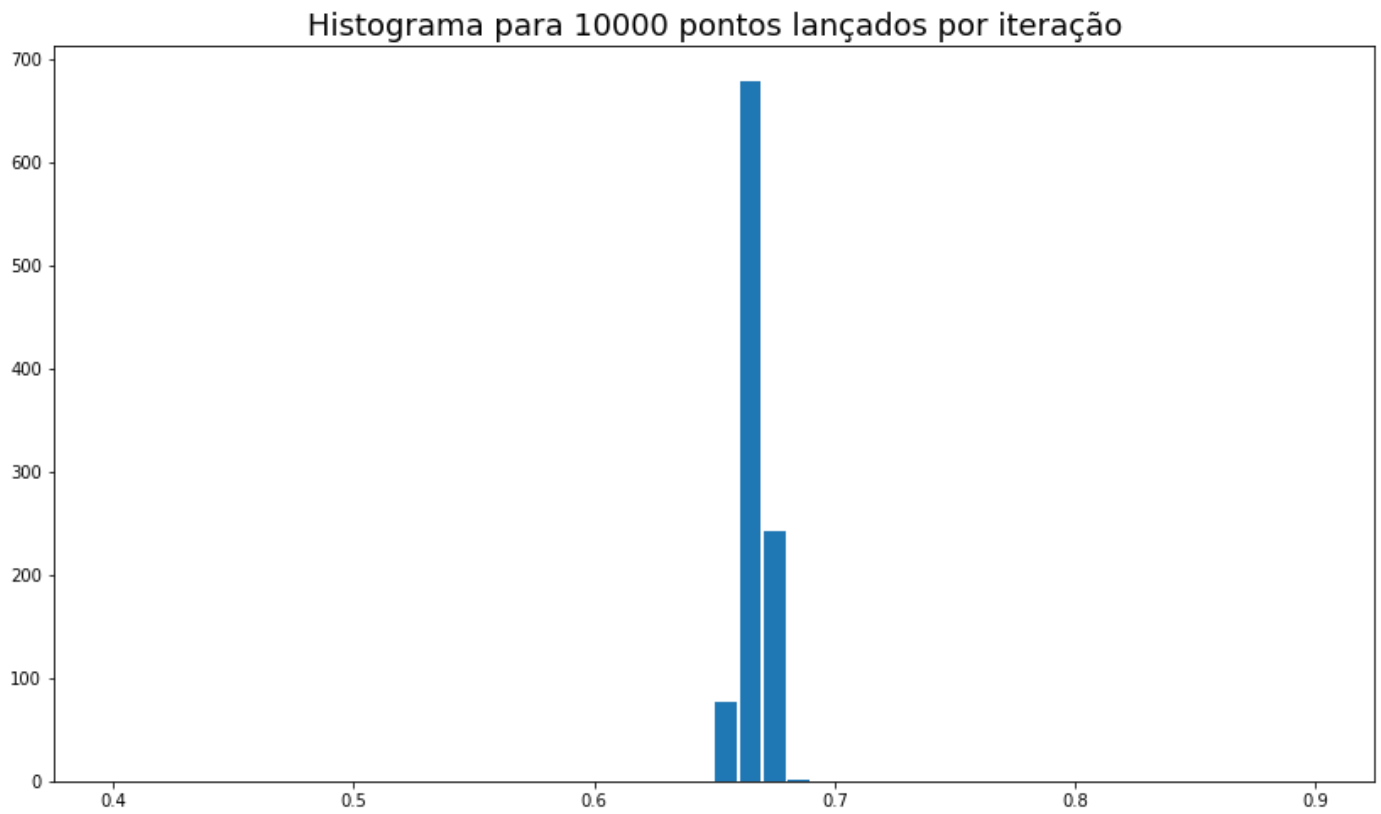
In [ ]:
```
resultados_1_1000 = monteCarlo1(funcao1, 0, 1, funcao1(0), 1000)
gerarHistograma(resultados_1_1000, 50, [0.4,0.9], "Histograma para 1000 pontos lançados
calcularEstatisticas(resultados_1_1000)
```




```
Média: 0.6664500000000001
Desvio Padrão: 0.014412338463968987
Erro: 0.0004557581595539458
```

In [ ]:
```
resultados_1_10000 = monteCarlo1(funcao1, 0, 1, funcao1(0), 10000)
gerarHistograma(resultados_1_10000, 50, [0.4,0.9], "Histograma para 10000 pontos lançado
```

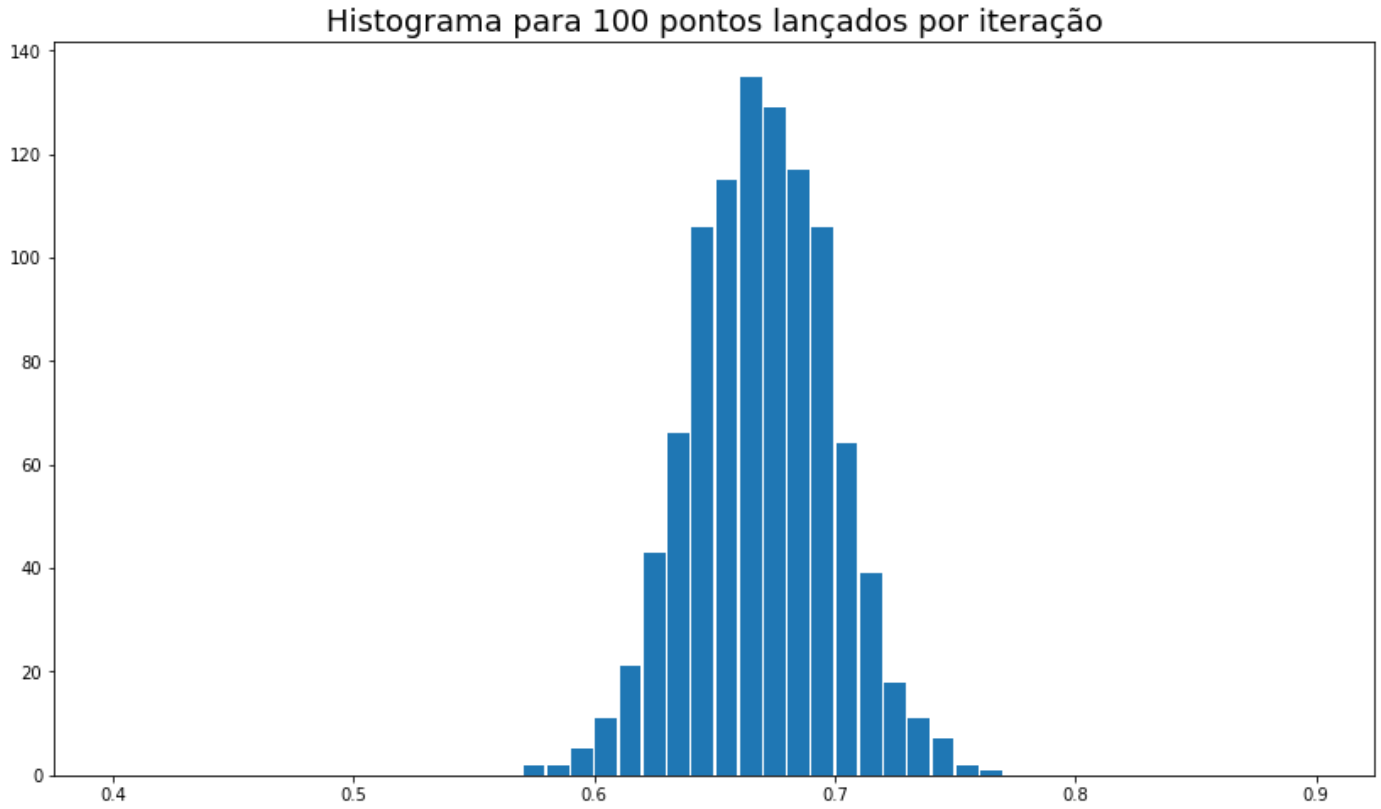
```
calcularEstatisticas(resultados_1_10000)
```




```
Média: 0.6665996
Desvio Padrão: 0.004677088821050973
Erro: 0.00014790253493432754
```

Utilizando o método 2:

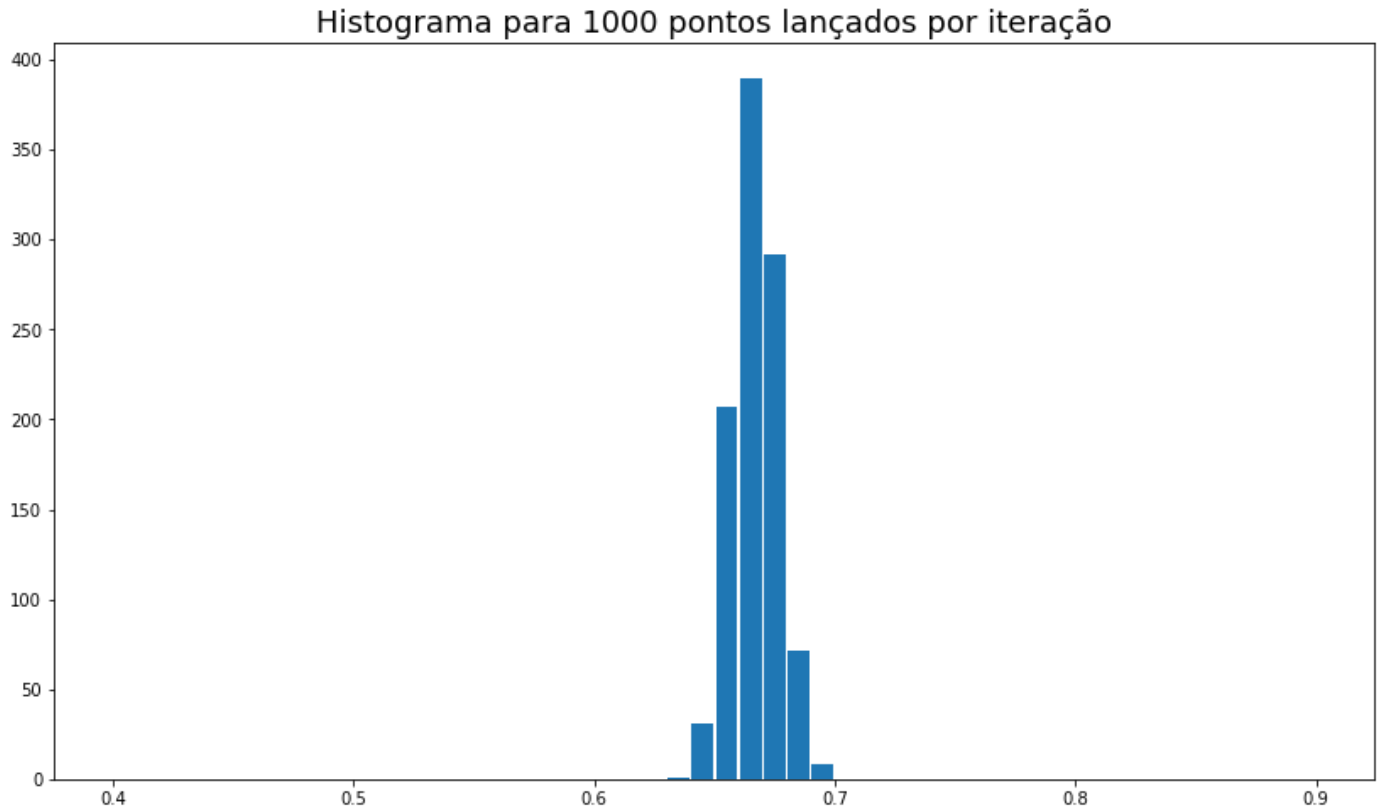
In [ ]:
```
resultados_1_100 = monteCarlo2(funcao1, 0, 1, 100)
gerarHistograma(resultados_1_100, 50, [0.4,0.9], "Histograma para 100 pontos lançados po
calcularEstatisticas(resultados_1_100)
```




```
Média: 0.6695922030701258
Desvio Padrão: 0.029073767817459455
```

```
Erro: 0.0009193932646607442
```

In [ ]:
```
resultados_1_1000 = monteCarlo2(funcao1, 0, 1, 1000)
gerarHistograma(resultados_1_1000, 50, [0.4,0.9], "Histograma para 1000 pontos lançados
calcularEstatisticas(resultados_1_1000)
```




```
Média: 0.6666755121003926
Desvio Padrão: 0.00935674141653182
Erro: 0.0002958861435347081
```

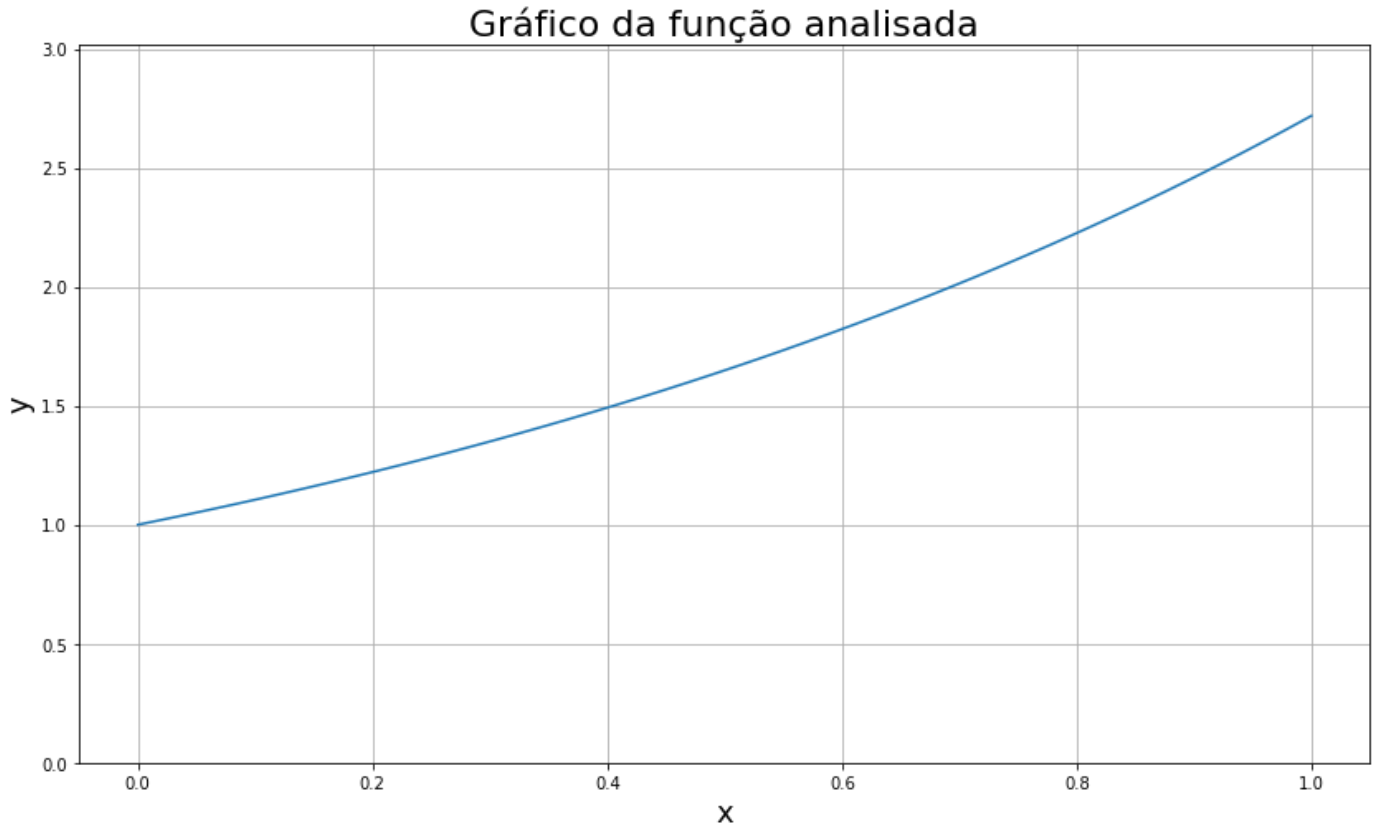
In [ ]:
```
resultados_1_10000 = monteCarlo2(funcao1, 0, 1, 10000)
gerarHistograma(resultados_1_10000, 50, [0.4,0.9], "Histograma para 10000 pontos lançado
calcularEstatisticas(resultados_1_10000)
```

```
Média: 0.6666628589695722
Desvio Padrão: 0.0029669105658730985
Erro: 9.382195002178025e-05
```

2 -

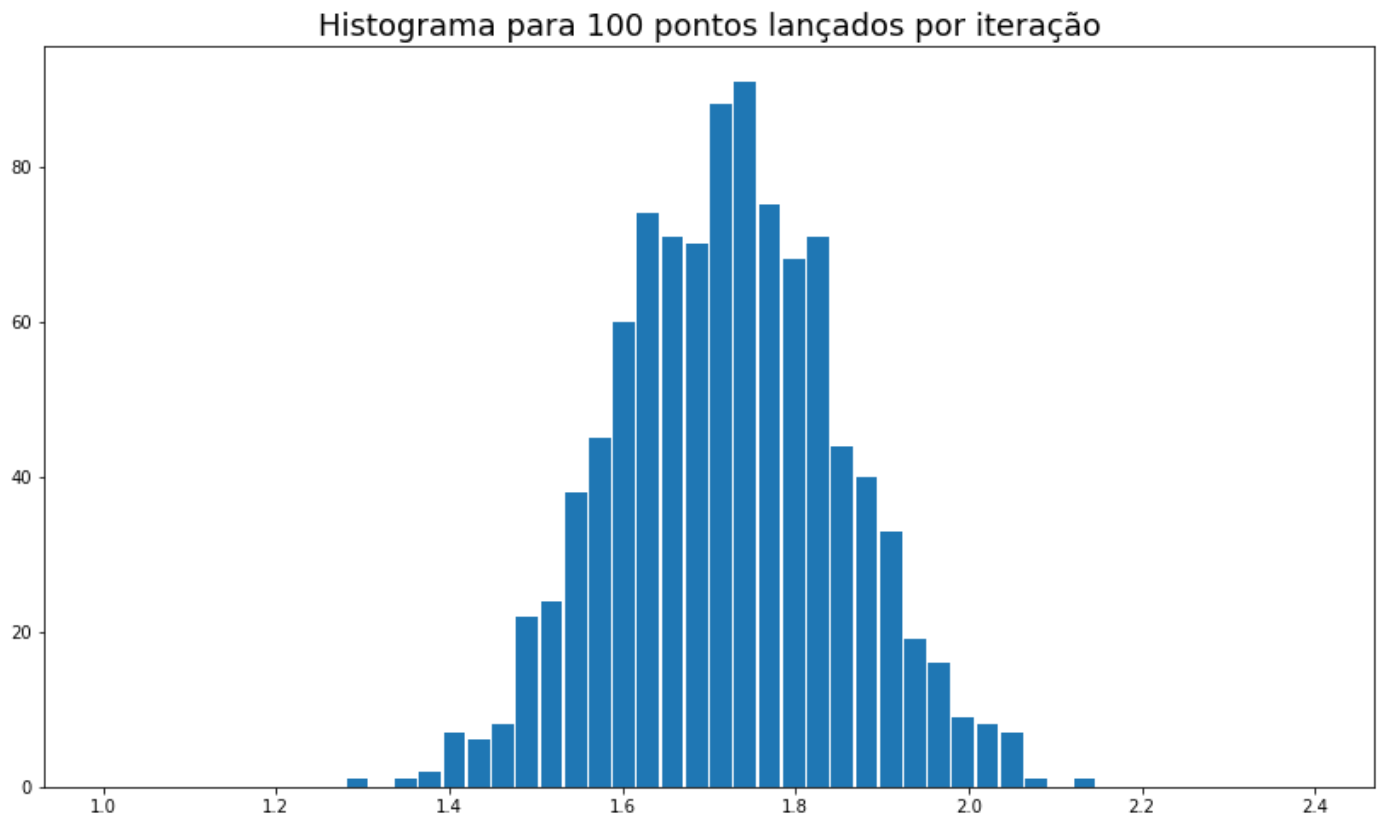
In [ ]:
```
def funcao2(x):

    return np.power(np.e, x)
```

In [ ]:
```
mostrarGrafico(funcao2, 0, 1)
```



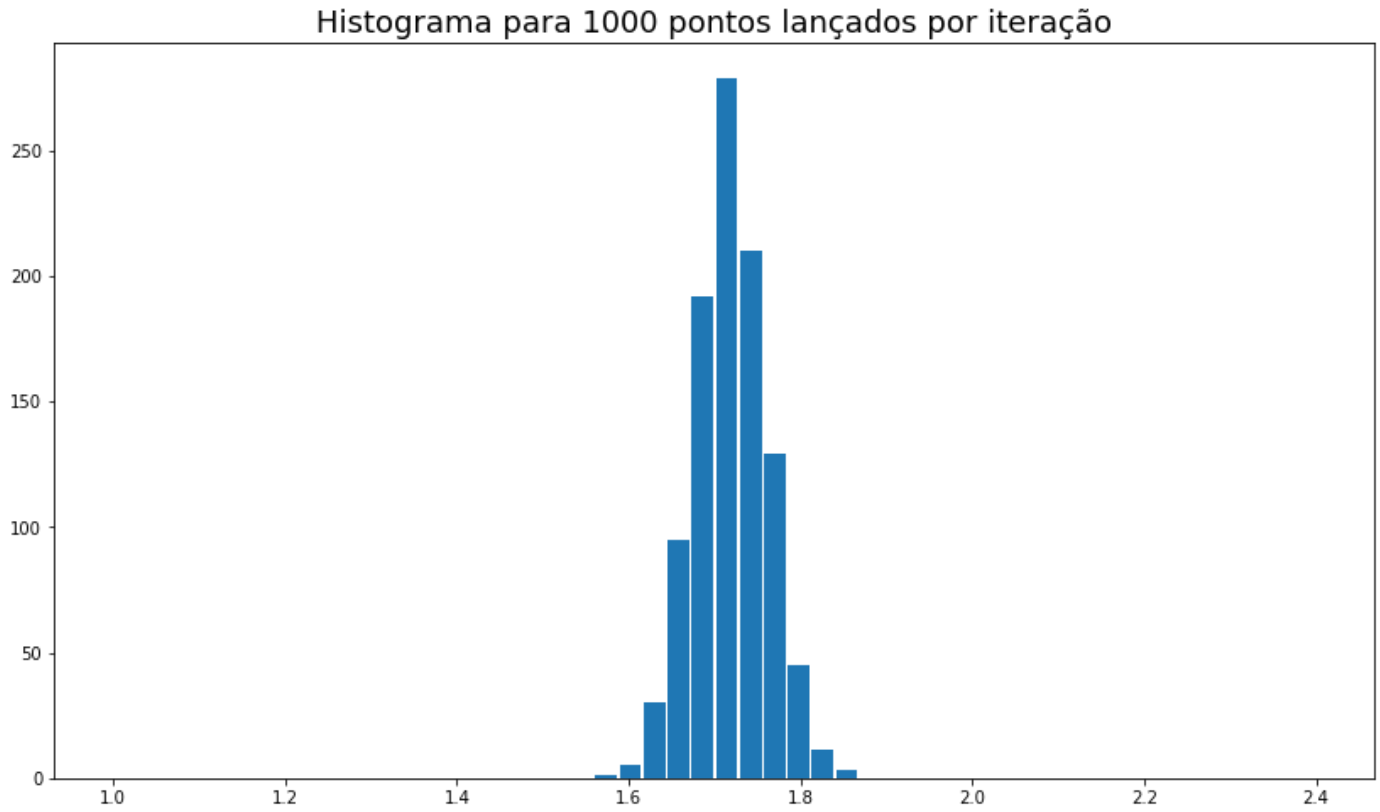
Resultados utilizando o método 1:

In [ ]:
```
resultados_2_100 = monteCarlo1(funcao2, 0, 1, funcao2(1), 100)
gerarHistograma(resultados_2_100, 50, [1, 2.4], "Histograma para 100 pontos lançados por
calcularEstatisticas(resultados_2_100)
```




```
Média: 1.7199928269574607
Desvio Padrão: 0.1271138311165837
Erro: 0.004019692284383888
```

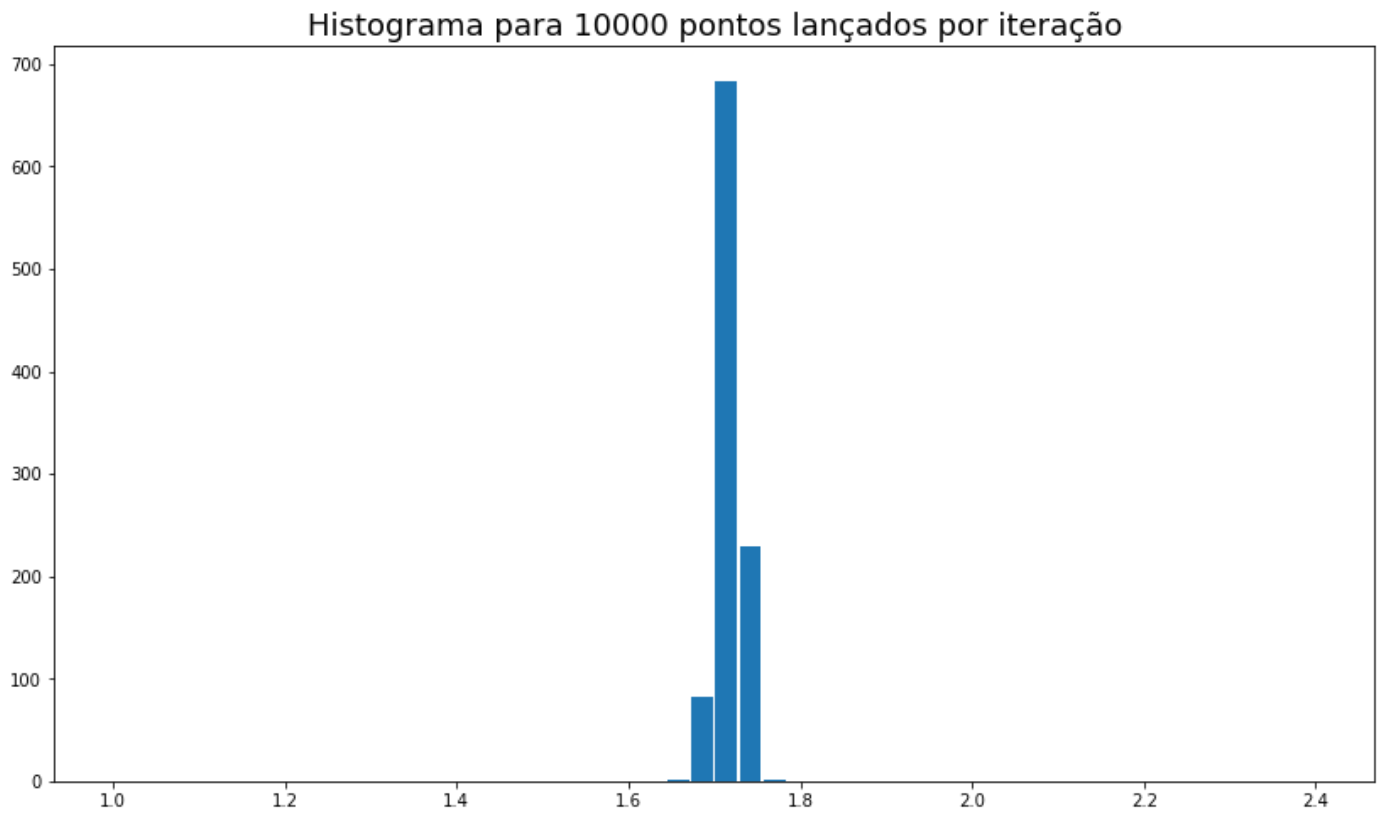
In [ ]:
```
resultados_2_1000 = monteCarlo1(funcao2, 0, 1, funcao2(1), 1000)
gerarHistograma(resultados_2_1000, 50, [1, 2.4], "Histograma para 1000 pontos lançados p
calcularEstatisticas(resultados_2_1000)
```




```
Média: 1.7188375571803658
Desvio Padrão: 0.04170887693807614
Erro: 0.001318950497719903
```

In [ ]:
```
resultados_2_10000 = monteCarlo1(funcao2, 0, 1, funcao2(1), 10000)
gerarHistograma(resultados_2_10000, 50, [1, 2.4], "Histograma para 10000 pontos lançados
```

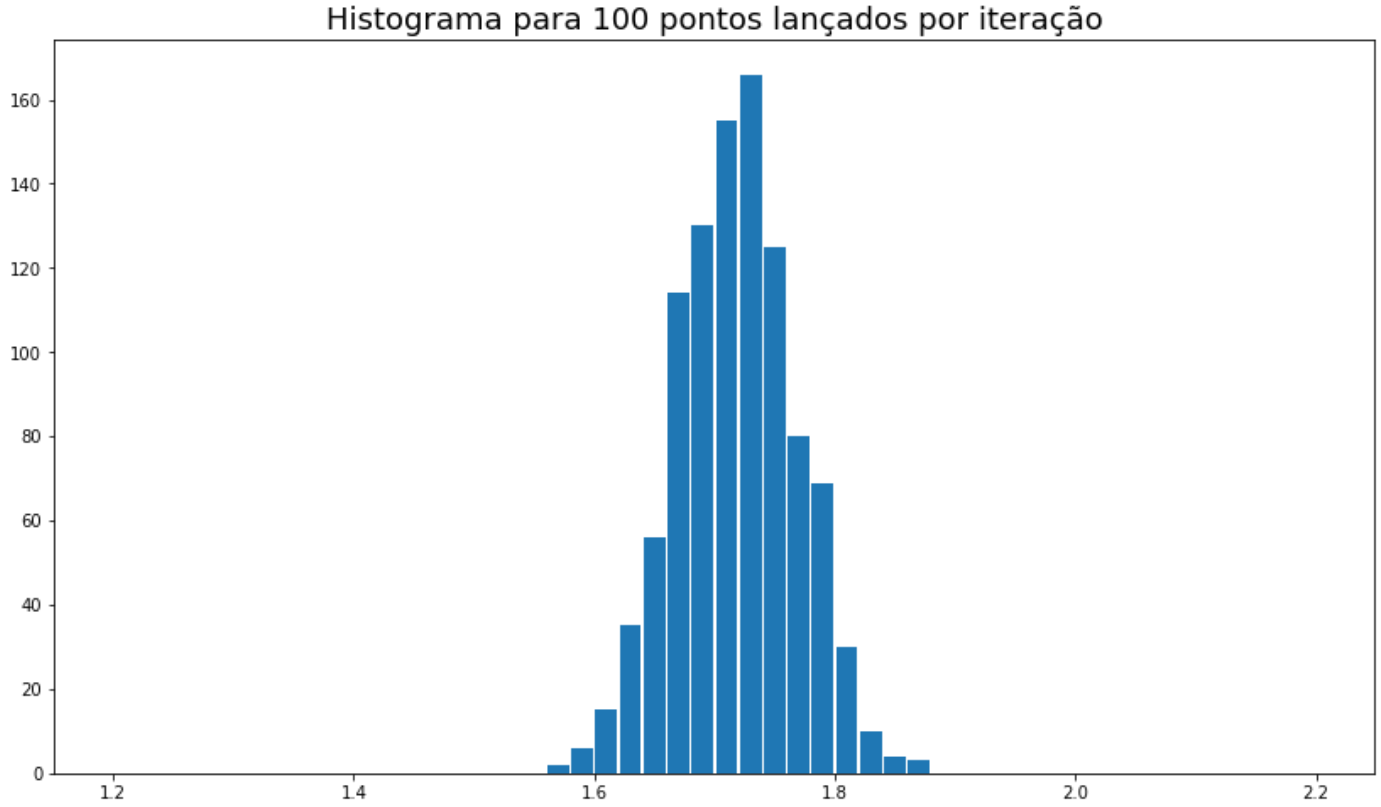
```
calcularEstatisticas(resultados_2_10000)
```




```
Média: 1.7179193215787123
Desvio Padrão: 0.01374185865815064
Erro: 0.00043455572643861194
```

Utilizando o método 2:

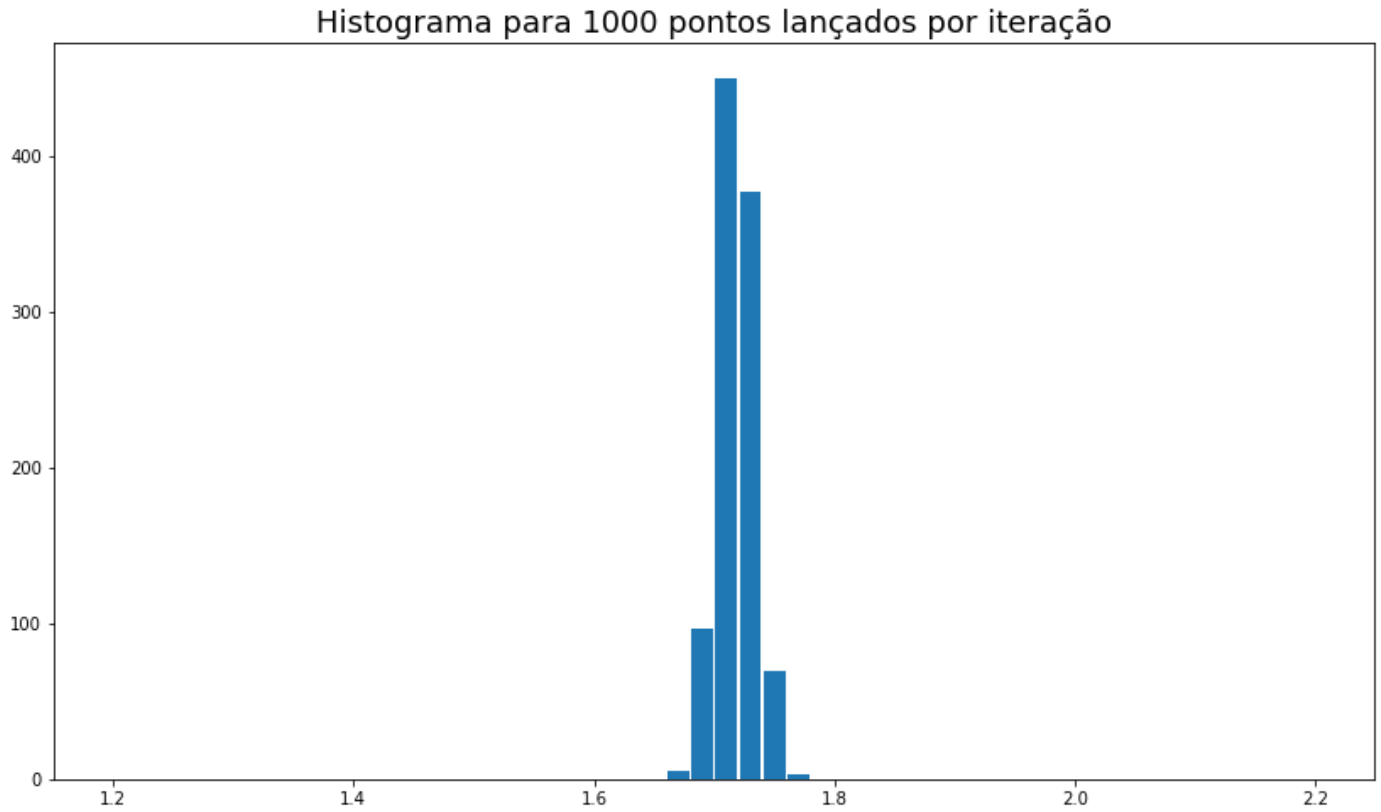
In [ ]:
```
resultados_2_100 = monteCarlo2(funcao2, 0, 1, 100)
gerarHistograma(resultados_2_100, 50, [1.2, 2.2], "Histograma para 100 pontos lançados p
calcularEstatisticas(resultados_2_100)
```




```
Média: 1.7180786393403105
Desvio Padrão: 0.04930226691121158
```

```
Erro: 0.0015590745724898309
```

In [ ]:
```
resultados_2_1000 = monteCarlo2(funcao2, 0, 1, 1000)
gerarHistograma(resultados_2_1000, 50, [1.2, 2.2], "Histograma para 1000 pontos lançados
calcularEstatisticas(resultados_2_1000)
```




```
Média: 1.718177332092408
Desvio Padrão: 0.015377777087078872
Erro: 0.00048628800945518694
```

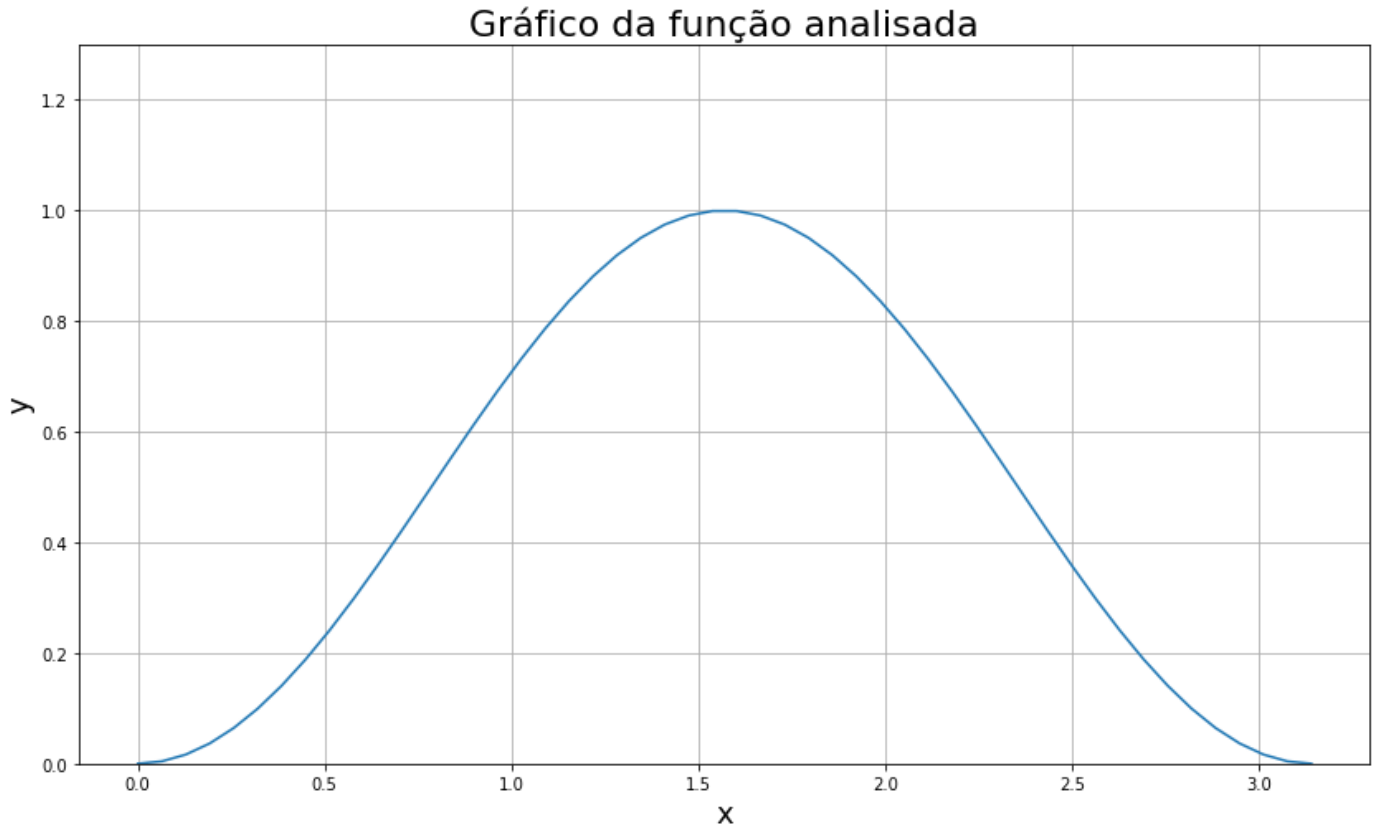
In [ ]:
```
resultados_2_10000 = monteCarlo2(funcao2, 0, 1, 10000)
gerarHistograma(resultados_2_10000, 50, [1.2, 2.2], "Histograma para 10000 pontos lançad
calcularEstatisticas(resultados_2_10000)
```

```
Média: 1.7183422517332891
Desvio Padrão: 0.004872215979222012
Erro: 0.00015407299746609174
```

3 -

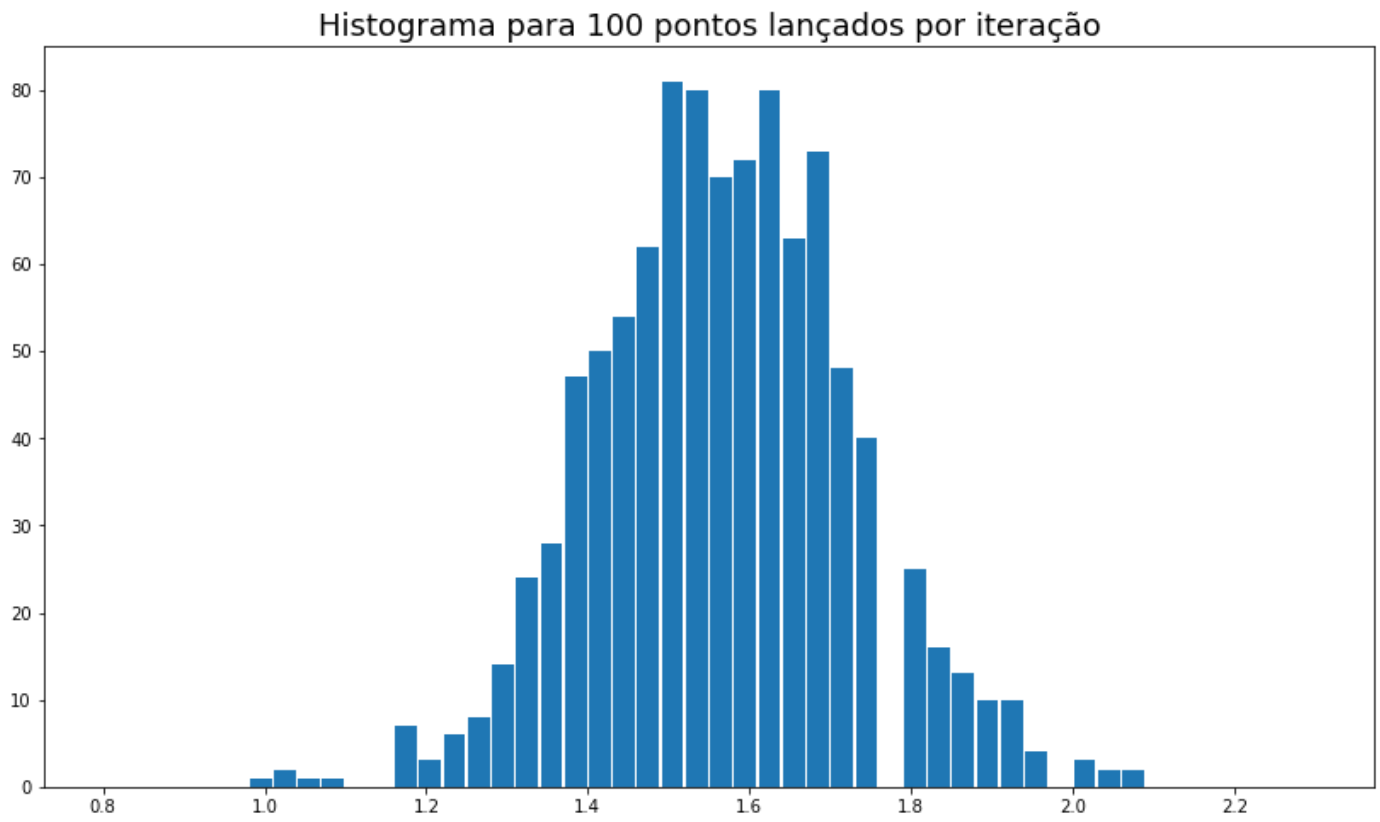
In [ ]:
```
def funcao3(x):

    return np.power(np.sin(x), 2)
```

In [ ]:
```
mostrarGrafico(funcao3, 0, np.pi)
```



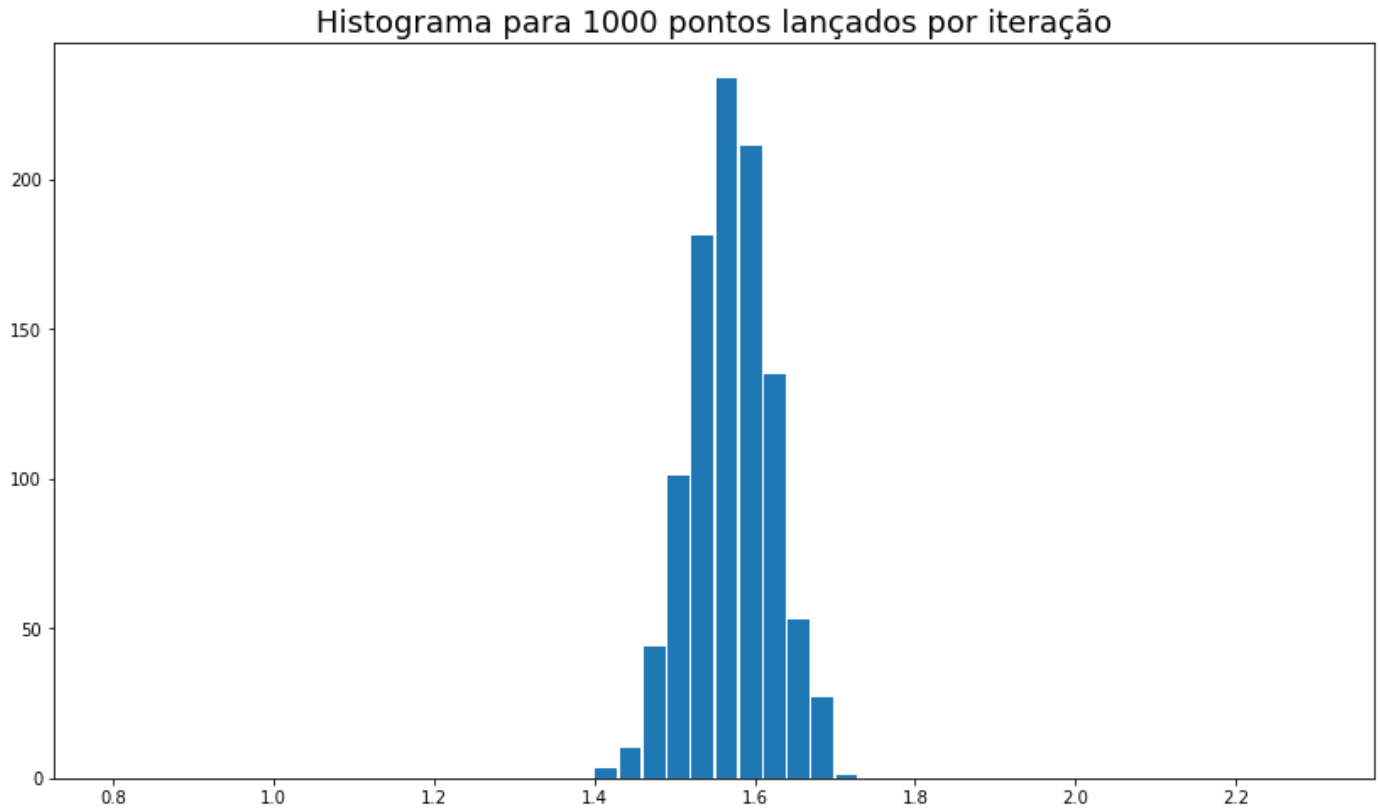
Pelo método 1:

In [ ]:
```
resultados_3_100 = monteCarlo1(funcao3, 0, np.pi, funcao3(np.pi/2), 100)
gerarHistograma(resultados_3_100, 50, [0.8, 2.3], "Histograma para 100 pontos lançados p
calcularEstatisticas(resultados_3_100)
```




```
Média: 1.568565796010848
Desvio Padrão: 0.15936875321211913
Erro: 0.00503968248011572
```

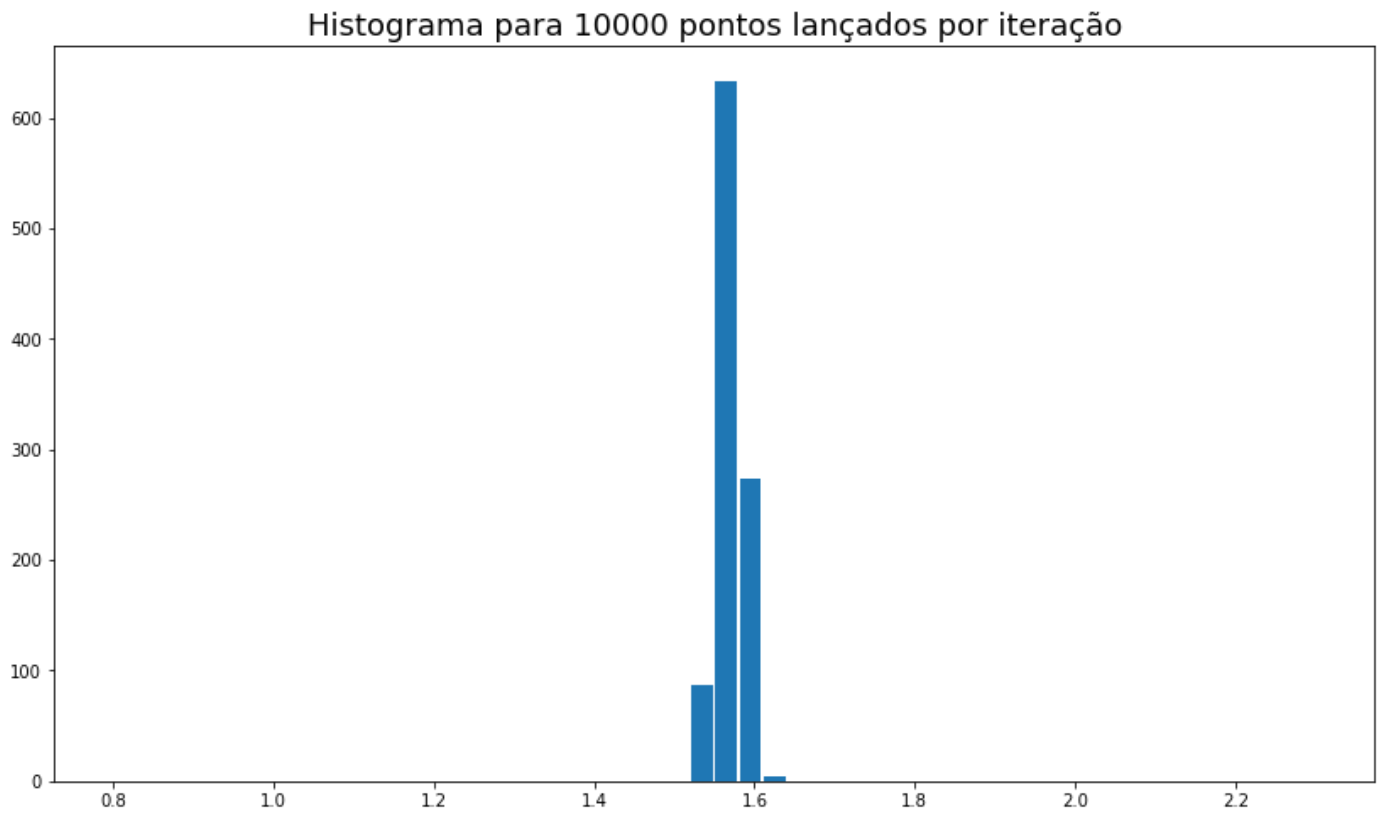
In [ ]:
```
resultados_3_1000 = monteCarlo1(funcao3, 0, np.pi, funcao3(np.pi/2), 1000)
gerarHistograma(resultados_3_1000, 50, [0.8, 2.3], "Histograma para 1000 pontos lançados
calcularEstatisticas(resultados_3_1000)
```




```
Média: 1.5707303533491712
Desvio Padrão: 0.05055002783557399
Erro: 0.0015985322374532536
```

In [ ]:
```
resultados_3_10000 = monteCarlo1(funcao3, 0, np.pi, funcao3(np.pi/2), 10000)
gerarHistograma(resultados_3_10000, 50, [0.8, 2.3], "Histograma para 10000 pontos lançad
```

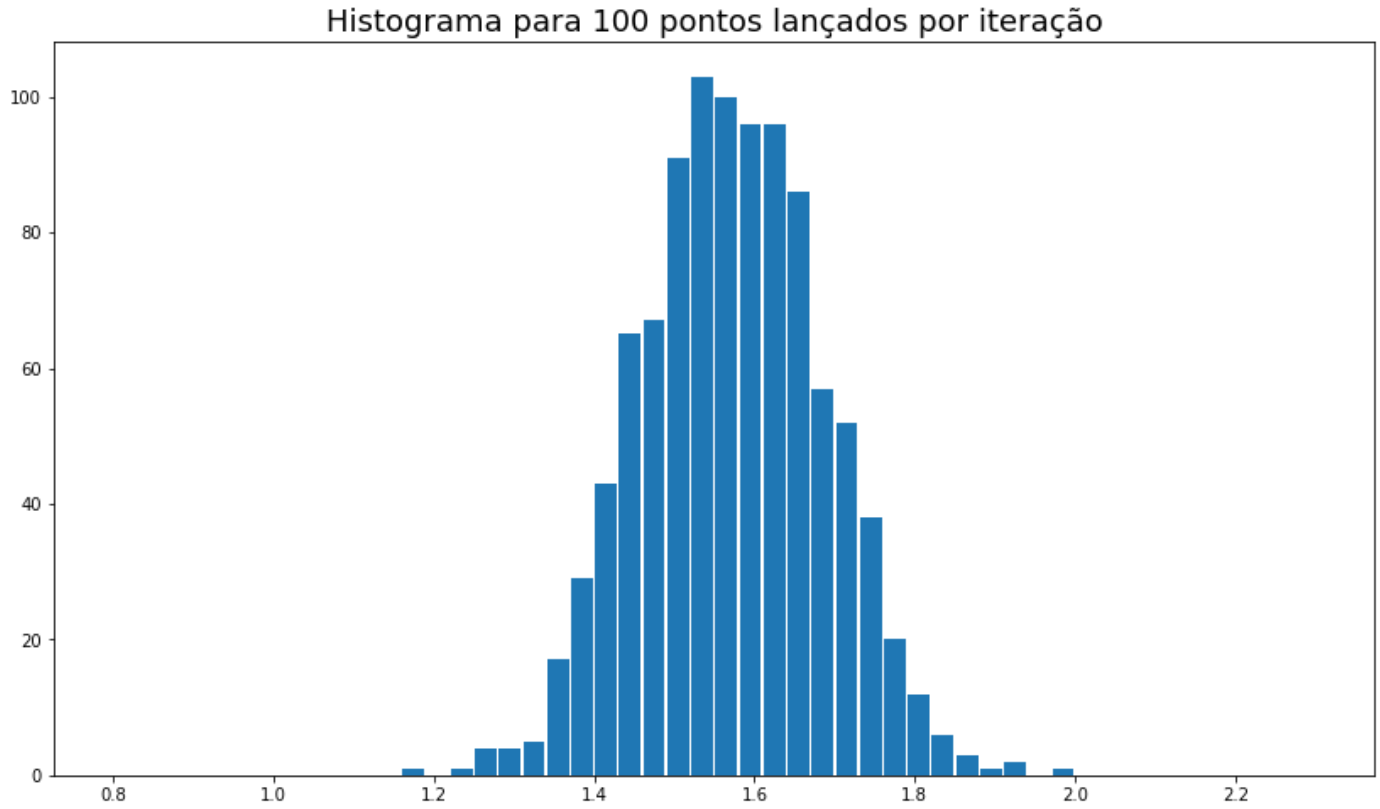
```
calcularEstatisticas(resultados_3_10000)
```




```
Média: 1.5711667205687547
Desvio Padrão: 0.015749449633896805
Erro: 0.0004980413273721893
```

Resultados usando o método 2:

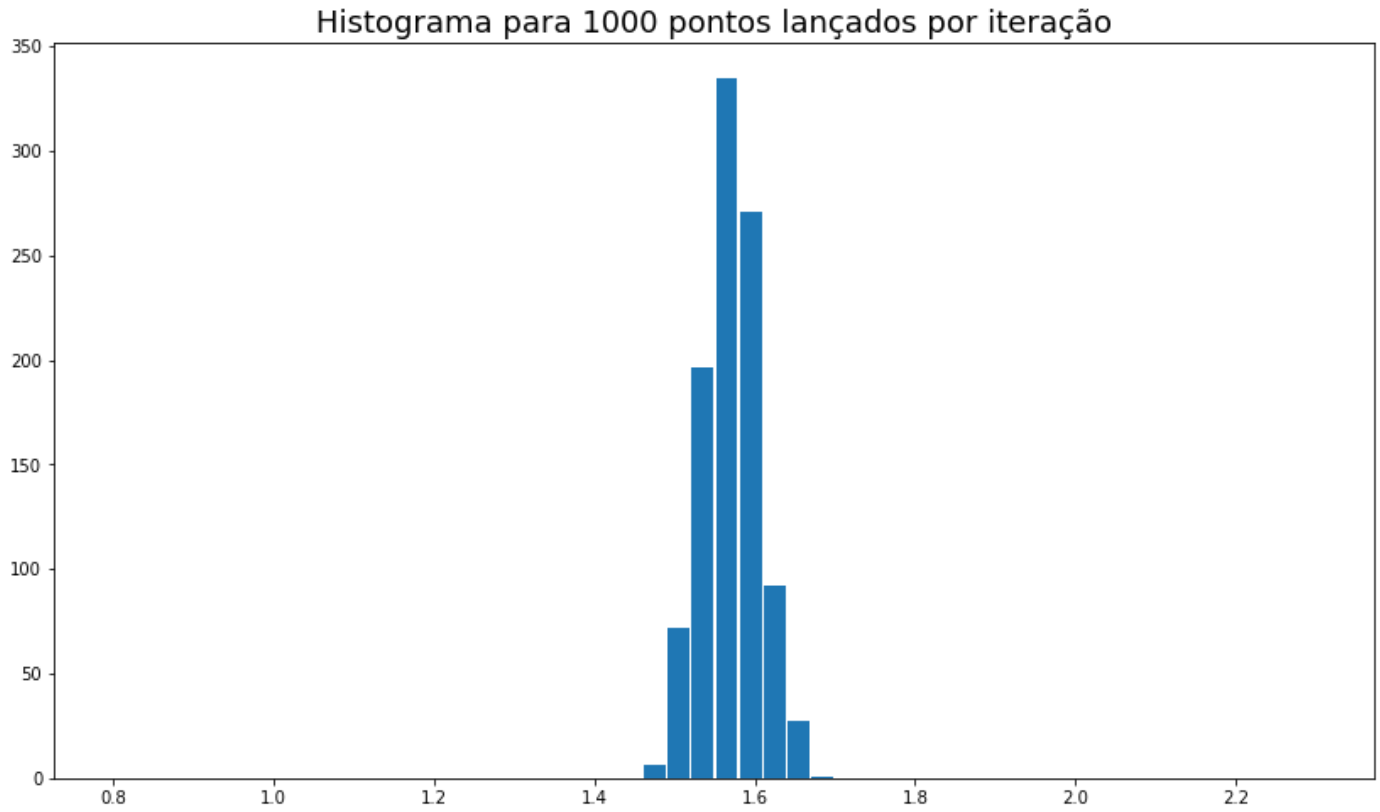
In [ ]:
```
resultados_3_100 = monteCarlo2(funcao3, 0, np.pi, 100)
gerarHistograma(resultados_3_100, 50, [0.8, 2.3], "Histograma para 100 pontos lançados p
calcularEstatisticas(resultados_3_100)
```




```
Média: 1.5723538093807818
Desvio Padrão: 0.11332488814598063
```

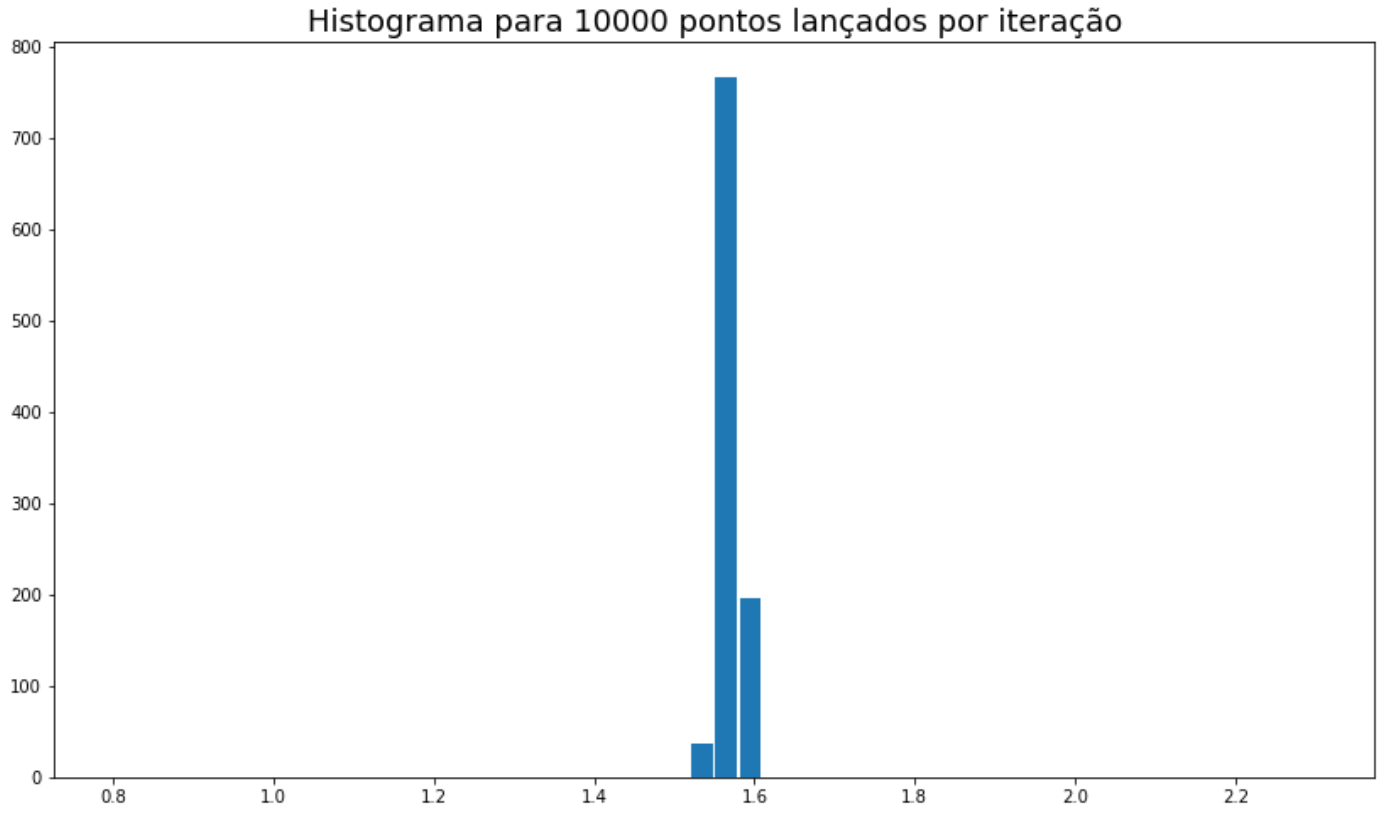
```
Erro: 0.0035836476212511 49
```

In [ ]:
```
resultados_3_1000 = monteCarlo2(funcao3, 0, np.pi, 1000)
gerarHistograma(resultados_3_1000, 50, [0.8, 2.3], "Histograma para 1000 pontos lançados
calcularEstatisticas(resultados_3_1000)
```




```
Média: 1.5707051525453395
Desvio Padrão: 0.034649520300537855
Erro: 0.001095714039819416
```

In [ ]:
```
resultados_3_10000 = monteCarlo2(funcao3, 0, np.pi, 10000)
gerarHistograma(resultados_3_10000, 50, [0.8, 2.3], "Histograma para 10000 pontos lançad
calcularEstatisticas(resultados_3_10000)
```



```
Média: 1.5709192169865764
Desvio Padrão: 0.011480742203475903
Erro: 0.0003630529459220414
```

Discussão dos resultados

Os resultados obtidos pelos métodos de Monte Carlo ficaram bem próximos do valor esperado para a integral. Esses valores esperados eram:

1:

$$\frac{2}{3} = 0.66666666...$$

2:

$$e - 1 = 1,7182818284...$$

3:

$$\frac{pi}{2} = 1.57079632679...$$

Além disso, é possível perceber que quanto maior o número de experimentos realizados para calcular os valores, menor será o desvio padrão e o erro. Isso ocorre pois a média dos resultados da realização da mesma experiência diversas vezes tem a tendência de ficar próxima do valor esperado, tal qual descrito pela Lei dos Grandes Números.

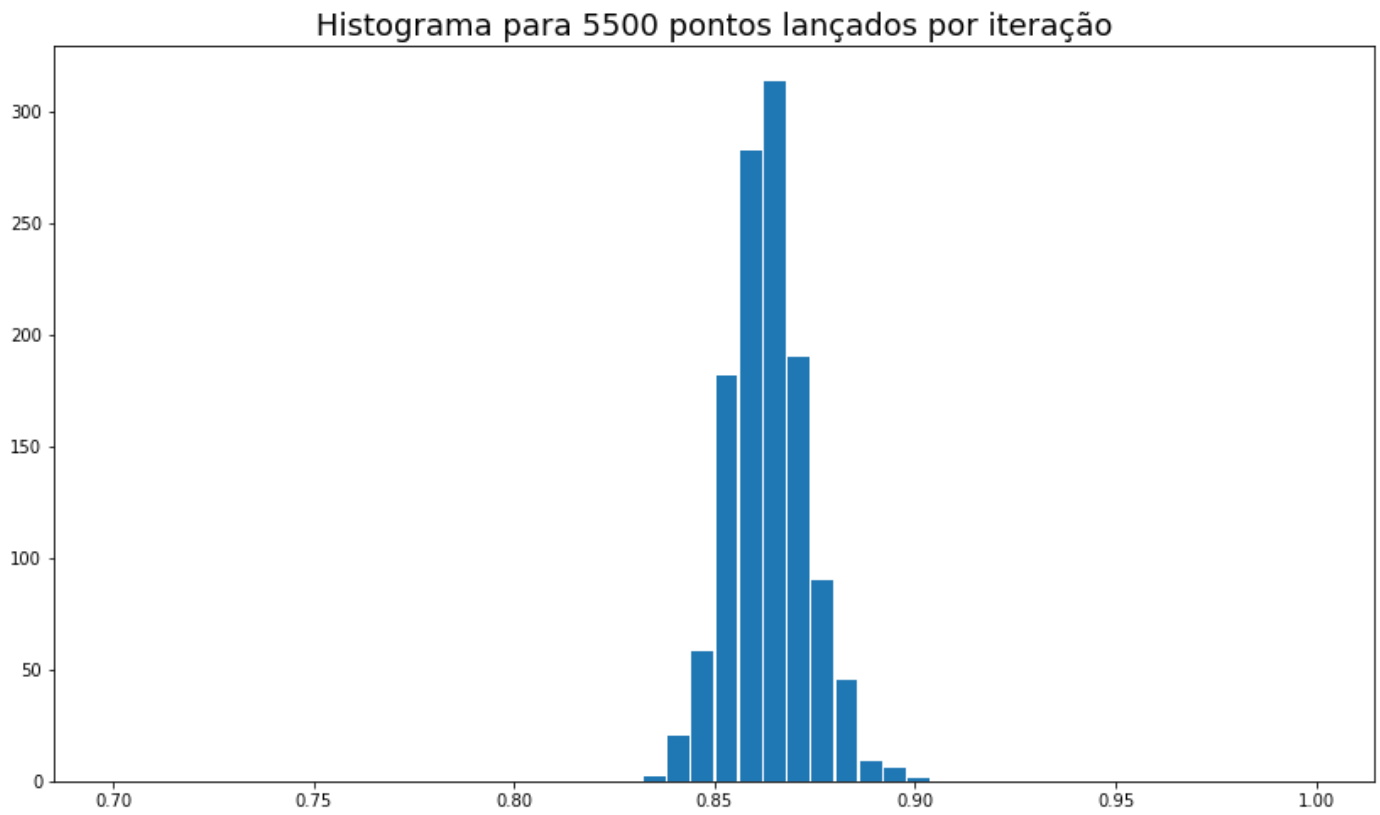
Cálculo de Integrais Múltiplas

In [ ]:
```python
def monteCarlo2Multidimensional(f, limite_esq, limite_dir, N_amostras, N_pontos):

    resultados = []
    media = (limite_dir - limite_esq)/N_pontos

    for _ in range(N_amostras):
        somatorio_y = 0
        for _ in range(N_pontos):
            valor_x = np.random.uniform(limite_esq, limite_dir, (3, 3))
            somatorio_y += f(valor_x)

        resultados.append(media*somatorio_y)

    return resultados
```

4 -

In [ ]:
```python
def funcao4(vetores):

    return 1/np.dot((vetores[0]+vetores[1]),vetores[2])
```

In [ ]:
```python
resultados_4 = monteCarlo2Multidimensional(funcao4, 0, 1, 1200, 5500)
gerarHistograma(resultados_4, 50, [0.7, 1.0], "Histograma para 5500 pontos lançados por
calcularEstatisticas(resultados_4)
```




```
Média: 0.8633414209793678
Desvio Padrão: 0.009515172646204773
Erro: 0.0002746793744336044
```

Após realização de diversos testes, os valores de 1200 amostras e 5500 pontos sorteados por iteração mostrou-se uma boa configuração por oferecer um bom compromisso entre tempo de execução e qualidade dos resultados gerados.