MAP2212 - EP1 Cálculo de π por Monte Carlo

Vinícius da Costa Collaço - 11811012

1 de abril de 2022

1 Introdução

Esse relatório visa apresentar uma solução para o primeiro exercício programa (EP1) proposto na matéria MAP2212/2022 (Laboratório de Computação e Simulação) do curso de bacharelado de matemática aplicada e computacional (BMAC) do instituito IME-USP.

O objetivo é utilizar o método estocástico de Monte Carlo para estimar o valor do número π com acurácia de 0.05%.

Utilizando a linguagem *Python* e bibliotecas adequadas[1, 4, 2], devemos gera pontos pseudo aleatórios, uniformemente distribuídos no intervalo $x_i \in [-1, 1]^2, i \in \{1, \ldots, n\}$, o valor de π será estimado pela proporção $p = \frac{1}{n}\sum_{i=1}^{n} T(x_i)$, sendo $T(x)$ a função indicadora dada por $T(x) = \mathbb{1}(\|x\|_2 \leq 1)$, que testa se o ponto cai dentro do círculo de raio 1 com centro na origem do plano cartesiano.

O número de pontos gerados deverá ser definido utilizando métodos apropriados

2 Cálculo do estimador de π

seja $p = \frac{1}{n}\sum_{i=1}^{n} T(x_i)$, tendo a função indicadora dada por $T(x) = \mathbb{1}(\|x\|_2 \leq 1)$, para $n \to \infty$ temos:

$$\hat{p} = \frac{area_{circulo}}{area_{quadrado}} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}$$

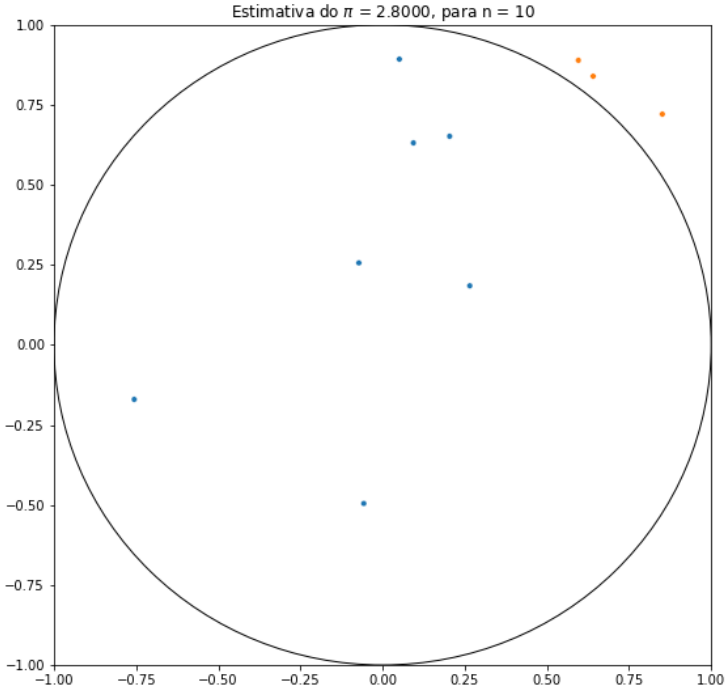
portanto o estimador $\hat{\pi}$ é dado por:

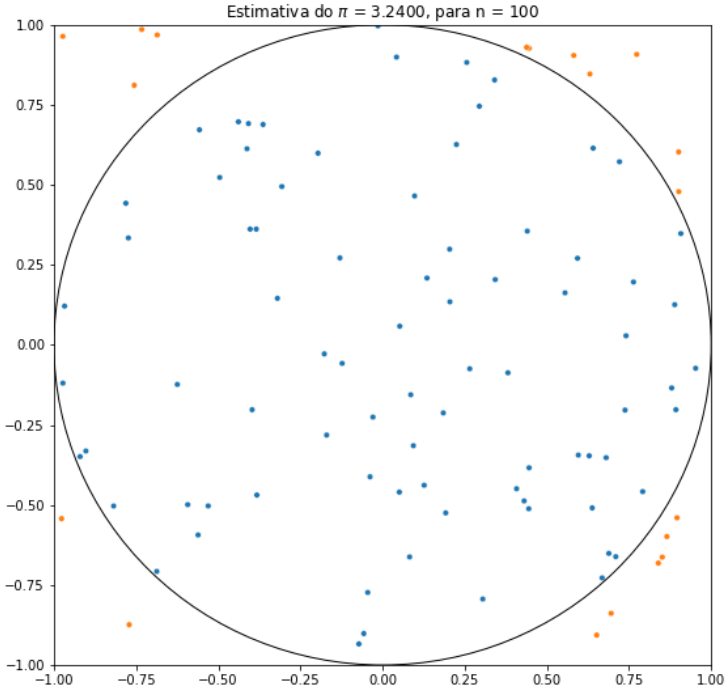
$$\hat{\pi} = \frac{4}{n}\sum_{i=1}^{n}[\mathbb{1}(\|x_i\|_2 \leq 1)]$$

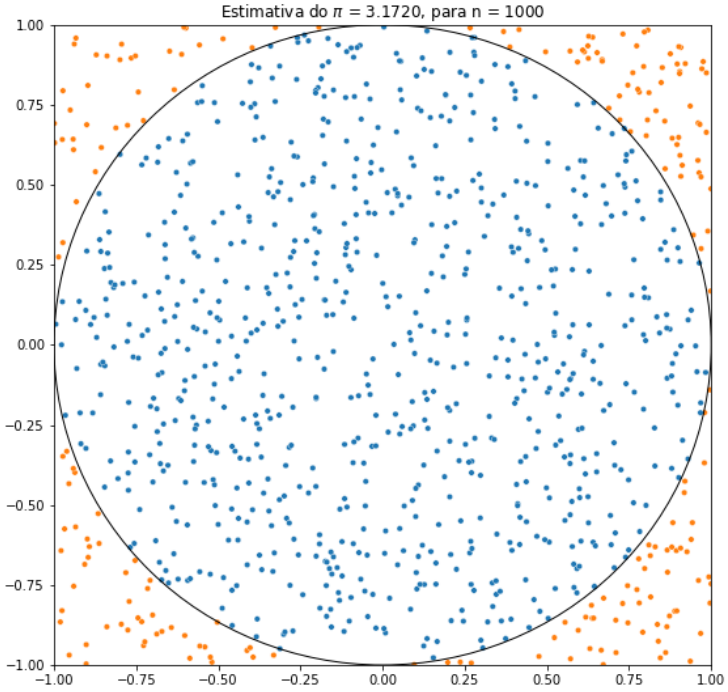
2.1 Simulações iniciais

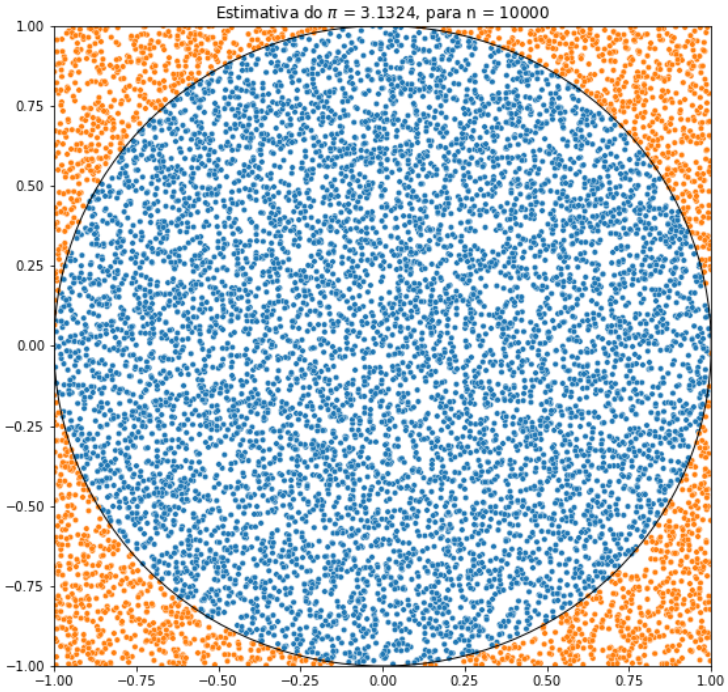
Inicialmente foram feitas algumas simulações iniciais para testar o algorítimo graficamente, para essas simulações foi utilizada a função gera_grafico () com valor de Seed =

39, e 'n' de 10, 100, 1000 e 10000, gerando as seguintes imagens respectivamente.









Podemos verificar inicialmente que o valor de π aproxima-se do valor real conforme o valor do 'n' vai aumentando, porém esses valores foram escolhidos arbitrariamente. Um valor de 'n' que atenda os requisitos do problema será calculado na próxima secção.

2.2 valor empírico do estimador $\hat{\pi}$

Para alguns cálculos futuros precisamos estimar um valor de π, como temos algumas simulações iniciais e tais simulações foram de rápida execução e trouxe uma boa estimativa, portanto será utilizado o valor de π estimado da simulação de n = 10000, ou seja:

$$\pi_{empirico} = \hat{\pi} = 3,1324$$

3 Definindo o tamanho da amostra (n)

Para a definição do valor do n, iremos utilizar a aproximação assintótica de uma distribuição Bernoulli.

Supondo que o tamanho da amostra seja relativamente grande, pelo teorema do limite central, podemos aproximar a Bernoulli por uma normal, tendo:

$$P(|\hat{p} - p| \leq \varepsilon) \geq \gamma$$

[3]

$$P(-\varepsilon \leq \hat{p} - p \leq \varepsilon) = P\left(\frac{-\sqrt{n}\varepsilon}{\sigma} \leq Z \leq \frac{\sqrt{n}\varepsilon}{\sigma}\right) \approx \gamma$$

obtendo finalmente

$$n = \frac{\sigma^2 {Z_\gamma}^2}{\varepsilon^2}$$

3.1 Intervalo de confiança

Para o problema será utilizado um intervalo de confiança $\gamma = 95\%$, escolhido arbitrariamente obtendo assim o Z_γ da $N(0,1)$, portanto $Z_\gamma = 1,96$

3.2 Variância

Para a variância podemos utilizar uma abordagem mais pessimista ou uma abordagem mais otimista.

Na abordagem pessimista podemos usar a variância máxima de uma Bernoulli, ou seja

$$\sigma^2_{pessimista} = 0,25$$

Porém sabemos que o nosso $\hat{p} = \frac{\pi}{4}$, dado que a variância de uma distribuição de Bernoulli é dada por $X \sim Ber(p)$; $Var(X) = p(1-p)$ [3], podemos então estabelecer a nossa estimativa de variância otimista dada por:

$$\sigma^2_{otimista} = \left(\frac{\hat{\pi}}{4}\right)\left(1 - \frac{\hat{\pi}}{4}\right)$$

Utilizando o valor do $\pi_{empirico} = 3,1324$, temos a variância otimista dada por:

$$\sigma^2_{otimista} = \left(\frac{3,1324}{4}\right)\left(1 - \frac{3,1324}{4}\right) = 0,1698$$

Poderíamos também estabelecer a variância empiricamente, porém a variância otimista é um bom parâmetro estabelecido, diminuindo o tamanho da amostra em relação à variância máxima, portanto será utilizado o valor calculado da variância otimista para cálculo da amostra

3.3 Erro amostral (ε)

No problema é pedido uma acurácia de 0,05%, ou seja:

$$\frac{|\hat{\pi} - \pi|}{\pi} \leq 0,0005$$

Temos, $\hat{\pi} = \dfrac{4}{n}\displaystyle\sum_{i=1}^{n} T(x_i)$, logo:

$$\frac{|\frac{4}{n}\sum_{i=1}^{n} T(x_i) - \pi|}{\pi} \leq 0,0005 \implies 4 * \frac{|\frac{1}{n}\sum_{i=1}^{n} T(x_i) - \pi|}{\pi} \leq 0,0005$$

$$\implies \frac{|\frac{1}{n}\sum_{i=1}^{n} T(x_i) - \pi|}{\pi} \leq \frac{0,0005}{4} \implies \frac{|\frac{1}{n}\sum_{i=1}^{n} T(x_i) - \pi|}{\pi} \leq 0,000125$$

então:

$$\varepsilon = 0,000125\pi$$

Porém não podemos utilizar o valor do π real, então teremos que aproximar o erro amostral com o valor do $\pi_{empirico}$, portanto temos:

$$\varepsilon \approx 0,000125 * \pi_{empirico} \approx 0,000125 * 3,1324 \approx 0,00039155$$

Portanto o erro amostral utilizado será dado por:

$$\varepsilon = 0,00039155$$

3.4 Cálculo do n

Tendo todas as variáveis calculadas podemos calcular o n adequado para o problema

$$n = \frac{\sigma^2 {Z_\gamma}^2}{\varepsilon^2} = \frac{0,1698 * (1,96)^2}{(0,00039155)^2} = 4256125.85$$

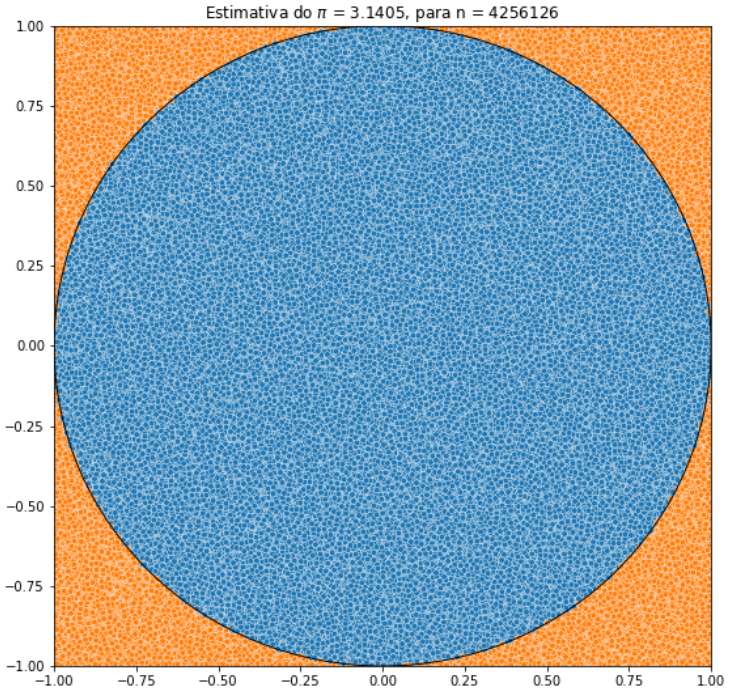
portanto:

$$n = 4256126$$

4 Resultados e discussões

Na fase inicial da criação do algorítimo tentou-se gerar ponto a ponto e a cada passo calcular o π, em um *loop*, porém com esse método o algorítimo demorava muito tempo (cerca de 30 min, no ambiente *Colab*) para executar rotinas com n na casa de 10^6, então ao invés desse método optou-se por utilizar a biblioteca *numpy* para gerar todos os pontos em um único *array* e então calcular a proporção da função indicadora, o algorítimo ficou expressivamente mais rápido, porém consumindo mais memória RAM, porém com o n calculado na ordem de $4 * 10^6$, o algorítimo não teve problema de executar no ambiente *Colab*. Começou-se a ter problemas com n na ordem de 10^8, não sendo necessário fazer qualquer mudança no algorítimo final.

Utilizando a função gera_grafico(n, Seed), com o $n = 4256126$ e $Seed = 39$, podemos observar graficamente o resultado da simulação com o π estimado dentro dos parâmetros estabelecidos.



Para um teste empírico, foi criada a função dentro_do_erro(n, acuracia, Seed), que verifica qual a porcentagem dentre n simulações estão dentro da acurácia desejada, comparando o valor simulado com o valor real de pi (dado pela biblioteca math no Phyton), a função foi executada com $n = 1000$, $acurcia = 0.0005$ e $Seed = 39$ obtendo o seguinte resultado:

Nessa simulação 95,7% dos $\pi's$ estimados ficaram dentro da acurácia estipulada, bem próximo do intervalo de confiança escolhido (95%), com os resultados desse experimento replicável, gera-se um indicativo que as contas e estipulações escolhidas durante o processo foram bem executadas, e que os resultados empíricos são condizentes com os resultados teóricos.

5 Conclusão

No processo de elaboração do exercício proposto, podemos verificar uma aplicação do método de Monte Carlo, gerando um número esperado a partir de números (ou pares de números) uniformemente aleatórios.

O método não aparenta ser a maneira mais eficiente de se calcular o valor de π, porém trás um caso prático onde foi necessário o uso de ferramentas estatísticas e de criação/otimização de algorítimos, além de uso e pesquisa de bibliotecas adequadas para uma boa execução do problema proposto

Referências

[1] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[2] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[3] Wilton de O. Bussab Pedro A. Morettin. *Estatística Básica,* 6^a *Edição*. 2010.

[4] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.