MAP2212 - EP2 Métodos de integração de Monte Carlo

Vinícius da Costa Collaço - 11811012

Abril de 2022

1 Introdução

Esse relatório visa apresentar uma solução para o segundo exercício programa (EP2) 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 implementar quatro variantes do método estocástico de integração de Monte Carlo, para integrar a função $f(x) = \exp(-ax)\cos(bx)$ no intervalo $[0, 1]$, onde $a = 0, RG, b = 0, CPF$. RG e CPF são os números de identificação do autor. Para proteção dos dados do autor, serão utilizados somente 6 dígitos significativos do RG e CPF, essa alteração não traz mudança significativa na função ou nos métodos utilizados.

Portanto, a integral cujo o valor será estimado será:

$$\gamma = \int_0^1 e^{-0.460279x}\cos(0,382023x)dx \tag{1}$$

A estimativa da integral deverá ser calculada com um erro mínimo de:

$$\frac{|\hat{\gamma} - \gamma|}{\gamma} \leq 0.0005 \tag{2}$$

Utilizando a linguagem *Python* e bibliotecas adequadas[2, 7, 3, 6], a integral em 1 será estimada utilizando os seguintes métodos de Monte Carlo:

1. *Crude*
2. *Hit or Miss*
3. *Importance Sampling*
4. *Control Variates*

2 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:

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

[5]

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

obtendo finalmente

$$n = \frac{\sigma^2 {Z_\gamma}^2}{\varepsilon^2} \tag{3}$$

esse resultado obtido em 3 será utilizado nos quatro métodos

2.1 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 $Z_\gamma = 1,96$ e será utilizado nos quatro métodos.

2.2 Erro amostral (ε)

O erro amostral é dado por $|\hat{\gamma} - \gamma|$ com o erro exigido pelo problema em 2, temos que o erro amostral será dado por:

$$\varepsilon = 0.0005 \cdot \gamma$$

porém como não sabemos o valor real de gamma, será utilizado o estimador $\hat{\gamma}$ para cálculo do erro amostral, portanto o erro amostral será dado por:

$$\varepsilon = 0.0005 \cdot \hat{\gamma} \tag{4}$$

o valor do estimador será obtido em cada método através de uma amostra piloto. Para meios de comparação será utilizado o mesmo tamanho de amostra para o piloto, que será descrito nas próximas seções.

2.3 Variância

Para obtenção da variância, em cada método será utilizada a variância amostral ($\hat{\sigma}^2$) obtida a partir da amostra piloto

2.4 Amostra Piloto

Para o tamanho da amostra piloto foi escolhido arbitrariamente $n = 100$

3 Método *Crude*

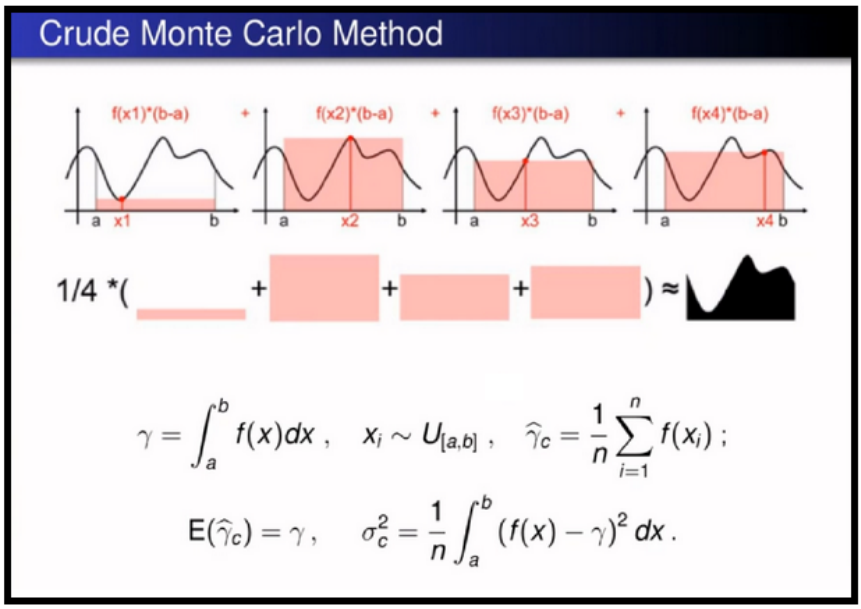

Figura 1: Método Crude

Um resumo do método *Crude* pode ser observado na figura 1

3.1 Estimador $\hat{\gamma}_c$ inicial

Para o cálculo do $\hat{\gamma}_c$ será utilizado um método empírico, uma amostra piloto na função crude(Seed,n), com $n = 100$ e $Seed = 38$, para ser possível a replicação. Resultando em

$$\hat{\gamma}_c = 0.773535$$

3.2 Variância amostral no Método *Crude*

Para o cálculo das variâncias empíricas foi criada a função variancias (Seed, n), retornando a variância de cada método. As variâncias poderiam ser retornadas nas funções principais, mas optou-se por criar outra função para que as funções que foram exigidas no exercício retornassem somente os valores como foram descritos.

Para o método *Crude*, a variância pode ser calculada por:

$$\hat{\sigma_c}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(f(x) - \hat{\gamma}_c)^2$$

Com $Seed = 38$ e $n = 100$, a variância amostral calculada na função variancia, resultou em:

$$\hat{\sigma_c}^2 = 0,0115492$$

Com o resultado obtido em 2, temos:

3.3 Erro amostral *Crude* (ε_c)

$$\varepsilon_c = 0,0005 \cdot \hat{\gamma} = 0,0005 \cdot 0,773535 = 0,000386768$$

3.4 Cálculo do n para o método *Crude* (n_c)

O cálculo do n pode ser obtido através da equação em 3, com os valores amostrais obtidos anteriormente, temos:

$$n_c = \frac{0,0115492 \cdot 1,96^2}{0,000386768^2} = 296595,03$$

portanto:

$$n_c = 296596$$

4 Método *Hit or Miss*

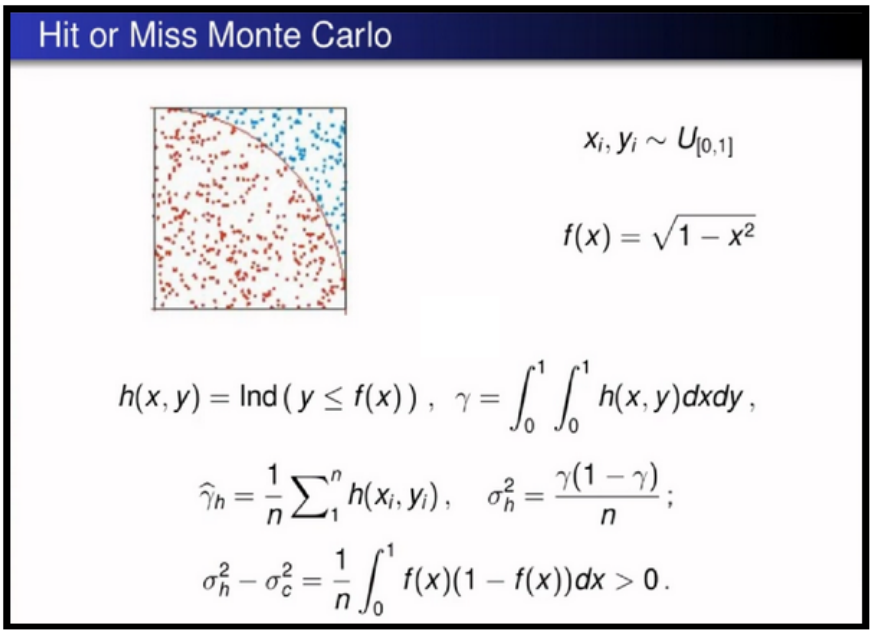

Figura 2: Método Hit or Miss

Um resumo do método *Hit or Miss* pode ser observado na figura 2

4.1 Estimador $\hat{\gamma}_{hom}$ inicial

Para o cálculo do $\hat{\gamma}_{hom}$ será utilizado um método empírico, uma amostra piloto na função hit_or_miss(Seed,n), com $n = 100$ e $Seed = 38$, para ser possível a replicação, resultando em:

$$\hat{\gamma}_{hom} = 0.76$$

4.2 Variância amostral no Método *Hit or Miss*

Para o método *Hit or Miss*, a variância pode ser calculada por:

$$\hat{\sigma}^2_{hom} = \hat{\gamma}_{hom}(1 - \hat{\gamma}_{hom})$$

Com $Seed = 38$ e $n = 100$, a variância amostral calculada na função variancia, resultou em:

$$\hat{\sigma}^2_{hom} = 0,1824$$

4.3 Erro amostral *Hit or Miss* (ε_{hom})

$$\varepsilon_{hom} = 0,0005 \cdot \hat{\gamma} = 0,0005 \cdot 0,76 = 0,00038$$

4.4 Cálculo do n para o método *Hit or Miss* (n_{hom})

O cálculo do n pode ser obtido através da equação em 3, com os valores amostrais obtidos anteriormente, temos:

$$n_{hom} = \frac{0,1824 \cdot 1,96^2}{0,00038^2} = 4852547,36$$

portanto:

$$n_{hom} = 4852548$$

5 Método *Importance Sampling*

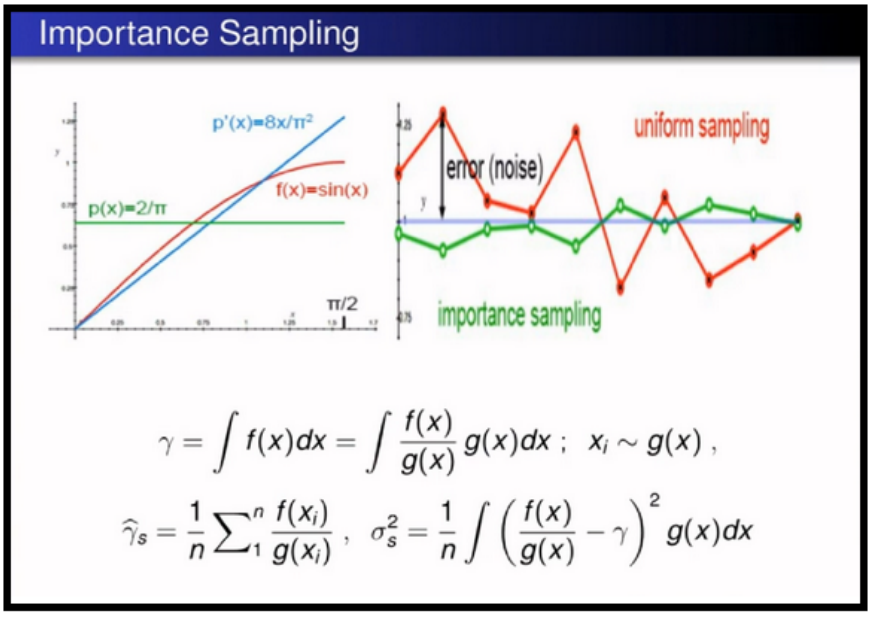

Figura 3: Método Importance Sampling

Um resumo do método *Importance Sampling* pode ser observado na figura 3.

Para esse método utilizou-se a função de probabilidade exponencial truncada da biblioteca scipy[6], a função scipy.stats.truncexpon, com parâmetros $b = 0.460279$ e $scale = 1/0.460279$, para resultar em uma $g(x)$, no intervalo $[0, 1]$, da seguinte forma:

$$g(x) = \frac{0,460279 \cdot \exp(-0,460279x)}{1 - \exp(-0,460279)}$$

Dessa forma,

$$\frac{f(x)}{g(x)} = \frac{\exp(-0,460279x) \cdot \cos(0,382023x)}{\frac{0,460279 \cdot \exp(-0,460279x)}{1-\exp(-0,460279)}}$$

Resultando em:

$$\frac{f(x)}{g(x)} = \frac{\cos(0,382023x) \cdot (1 - \exp(-0,460279))}{0,460279} \tag{5}$$

Essa a escolha da $g(x)$, simplificou algebricamente o resultado em 5, além de ser de fácil implementação da sua distribuição. Podemos observar a forte correlação no intervalo $[0, 1]$ através da imagem 4, gerada pela plataforma Desmos[1].

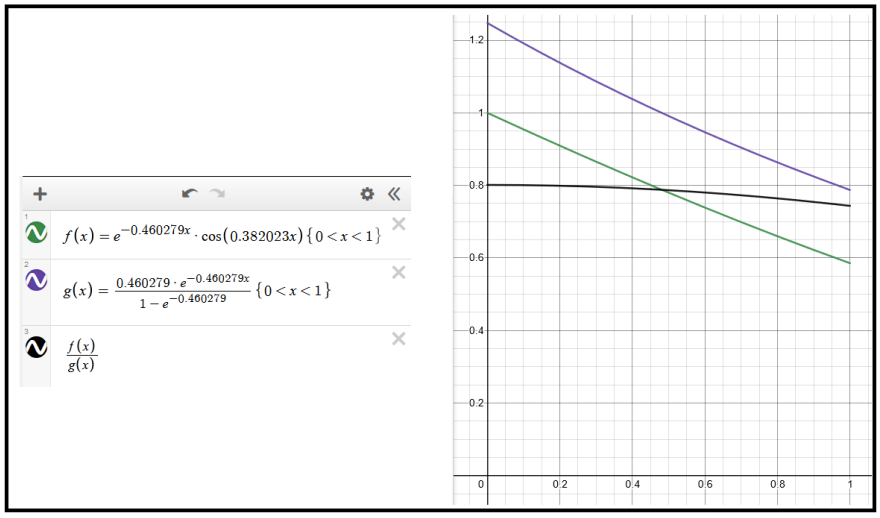

Figura 4: Curvas do Importance Sampling gerada pela plataforma Desmos

Foi plotado também a equação 5, podendo ser observada um valor próximo da integral preterida.

5.1 Estimador $\hat{\gamma}_{is}$ inicial

Para o cálculo do $\hat{\gamma}_{is}$ será utilizado um método empírico, uma amostra piloto na função importance_sampling(Seed,n), com $n = 100$ e $Seed = 38$, para ser possível a replicação, resultando em:

$$\hat{\gamma}_{is} = 0.784086$$

5.2 Variância amostral no Método *Importance Sampling*

Para o método *Importance Sampling*, a variância foi calculada empiricamente com auxílio da função numpy.var() da biblioteca numpy [2], com grau de liberdade = 1:

Com $Seed = 38$ e $n = 100$, a variância amostral calculada na função variancia(), resultou em:

$$\hat{\sigma}^2_{is} = 0,000287614$$

5.3 Erro amostral *Importance Sampling* (ε_{is})

$$\varepsilon_{is} = 0,0005 \cdot \hat{\gamma} = 0,0005 \cdot 0,784086 = 0,000392043$$

5.4 Cálculo do n para o método *Importance Sampling* (n_{is})

O cálculo do n pode ser obtido através da equação em 3, com os valores amostrais obtidos anteriormente, temos:

$$n_{is} = \frac{0,000287614 \cdot 1,96^2}{0,000392043^2} = 7186,26$$

portanto:

$$n_{is} = 7187$$

6 Método *Control Variates*

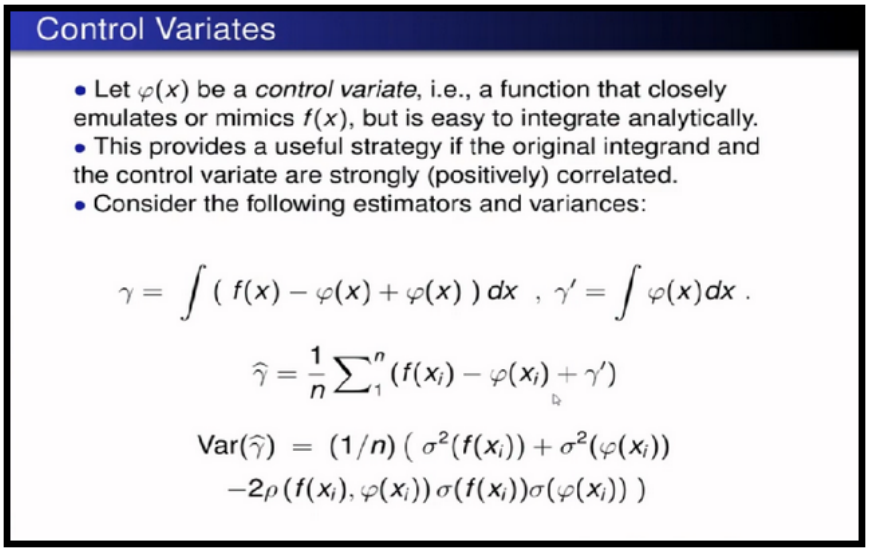

Figura 5: Método Control Variates

Um resumo do método *Control Variates* pode ser observado na figura 5.

Para a escolha da função $\varphi(x)$ de controle, foi utilizado a ferramenta gráfica Desmos[1], para achar uma

função com alta correlação e fácil integração no intervalo $[0, 1]$.

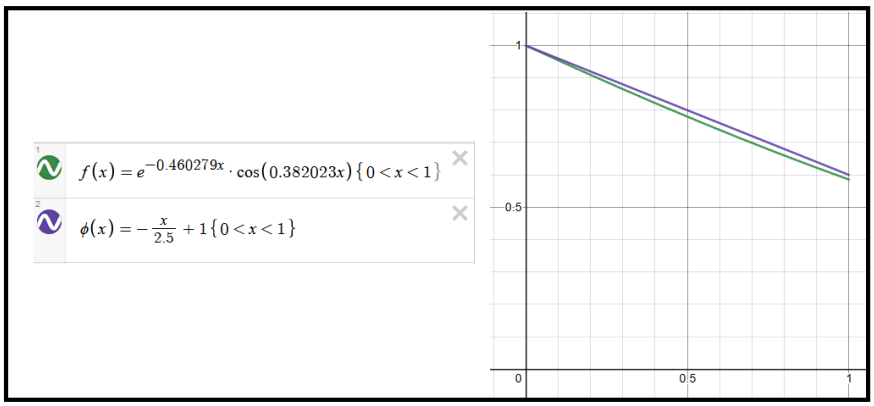

Figura 6: Curvas da função $f(x)$ e $\varphi(x)$ pela plataforma Desmos

Optou-se por:

$$\varphi(x) = -\frac{x}{2,5} + 1$$

Na figura 6 pode-se observar uma alta correlação entre as duas figuras.

6.1 Integral de $\varphi(x)$

$$\int_0^1 \varphi(x)dx = \int_0^1 -\frac{x}{2,5}dx + \int_0^1 1dx = -0,2 + 1 = 0,8$$

6.2 Estimador $\hat{\gamma}_{cv}$ inicial

Para o cálculo do $\hat{\gamma}_{cv}$ será utilizado um método empírico, uma amostra piloto na função control_variate(Seed,n), com $n = 100$ e $Seed = 38$, para ser possível a replicação, resultando em:

$$\hat{\gamma}_{cv} = 0,783084$$

6.3 Variância amostral no Método *Control Variate*

No método *Control Variate* a variância pode ser calculada por:

$$var(\hat{\gamma}_{cv}) = (1/n)var(f(x)) + var(\varphi(x) - 2 \cdot cov(f(x), \varphi(x))$$

Como a variância de $f(x)$ e $\varphi(x)$ é desconhecida, foi calculada empiricamente a covariância amostral e as variâncias amostrais com auxílio da função numpy.var(), e numpy.cov() da biblioteca numpy [2], com grau de liberdade = 1:

Com $Seed = 38$ e $n = 100$, a variância amostral calculada na função variancia(), resultou em:

$$\hat{\sigma}^2_{cv} = 0,0000352536$$

6.4 Erro amostral *Control Variate* (ε_{cv})

$$\varepsilon_{cv} = 0,0005 \cdot \hat{\gamma} = 0,0005 \cdot 0,783084 = 0,000391542$$

6.5 Cálculo do n para o método *Control Variate* (n_{cv})

O cálculo do n pode ser obtido através da equação em 3, com os valores amostrais obtidos anteriormente, temos:

$$n_{cv} = \frac{0,0000352536 \cdot 1,96^2}{0,0003915423^2} = 883,40$$

portanto:

$$n_{cv} = 884$$

7 Resultados e discussões

Para comparação dos métodos será utilizado o seguinte método de cálculo de eficiência: [4]

$$Eff(\hat{\gamma}) = [MSE(\hat{\gamma}) \times C(\hat{\gamma})]^{-1}$$

Como os estimadores são não-viesados a eficiência pode ser dada por:

$$Eff(\hat{\gamma}) = [\hat{\sigma}^2(\hat{\gamma}) \times C(\hat{\gamma})]^{-1}$$

onde $\hat{\sigma}^2(\hat{\gamma})$ é a variância do estimador e $C(\hat{\gamma})$ o tempo estimado para calcular $\hat{\gamma}$

Portanto quanto maior a eficiência melhor o método, contabilizando o tempo de execução, em consequência o algorítimo utilizado

	$\hat{\sigma}^2(\hat{\gamma})$	n	$C(\hat{\gamma})$(s)	Eficiência ($\hat{\gamma}$)
Crude	0,01154	296596	0,02039	4246,2
Hit or Miss	0,1824	4852548	0,7539	7,3
Importance Sampling	0,0002875	7187	0,0007908	4398001,2
Control Variate	0,00003554	884	0,004347	6525271,3

Tabela 1: Tabela Comparativa

8 Conclusão

A tarefa foi eficiente para mostrar o método de Monte Carlo nativo, e algumas variantes com redução de variância. Na tabela 1, podemos observar que o método mais eficiente foi o método *Control Variate*, seguido do *Importance Sampling*, em ambos foi possível encontrar facilmente funções apropriadas e tal fato se mostrou efetivo nos resultados. O método *Crude* foi o método de mais fácil implementação, porém precisou de uma amostra grande para atingir o intervalo de confiança desejado. O método *Hit or Miss* foi o com cálculo de variância mais fácil de se calcular, porém por ter um n grande e necessitar de dois arrays de pontos foi o método menos eficiente.

Referências

[1] Inc. Desmos. Plataforma desmos. `https://www.desmos.com/`, 2022. [Online; acessado 19-Abril-2022].

[2] 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.

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

[4] C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer Series in Statistics. Springer, 2008.

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

[6] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

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