Monte Carlo: Números pseudo-aleatórios, estimativas de π e cálculo de integrais definidas.

Introdução a Física Estatística e Computacional - IFEC

1 Números pseudo-aleatórios

Em qualquer processo onde haja a necessidade de se escolher de forma não viesada entre diferentes alternativas, ou em situações onde pretendemos amostrar distribuições de probabilidades, ou ainda no uso de protocolos de criptografia, gerar números aleatórios é uma tarefa essencial. Mas como podemos gerar tais números? Existem basicamente duas opções. A primeira, seria utilizarmos algum fenômeno físico sabidamente aleatório para gerá-los. Até muito pouco tempo atrás essa era uma forma extremamente ineficiente de se fazer isso. De fato, os processos de medida são, em geral, complicados e demandam equipamentos grandes, fornecendo números numa taxa nem sempre compatível com a demandada em grande parte das aplicações. Atualmente, existem dispositivos no mercado que exploram fenômenos quânticos que são capazes de gerar números realmente aleatórios que podem ser implementados em dispositivos pequenos, chips, integrados a computadores ou até mesmo a aparelhos de celular. Veja, por exemplo, as soluções disponíveis em https://www.idquantique.com/random-number-generation/overview/.

A segunda forma de se obter números "aleatórios" seria simularmos um processo dinâmico que gere uma sequência de valores que apresenta as propriedades que esperamos de uma sequência de números aleatórios. De fato, esse processo gera o que chamamos formalmente de números pseudo-aleatórios e há diversos algoritmos capazes de gerarem tais números. A ideia básica por trás de tais algoritmos é usar uma semente inicial, um ou mais números que são passados ao algoritmo, e, partindo desta semente, uma sequência de outros números é gerada de forma determinística, seguindo uma sequência de operações matemáticas fornecidas pelo algoritmo. Assim, fornecendo a mesma semente, a mesma sequência de números será gerada, evidenciando o fato que não há nada de realmente aleatório neste processo. No entanto, devido à dificuldade de se utilizar valores realmente aleatórios, esse processo ainda é extensivamente utilizado.

O gerador de números aleatórios mais simples e também o mais utilizado até hoje se baseia no método Linear Congruencial. Esse algoritmo pode ser sintetizado na seguinte expressão matemática:

$$x_{n+1} = (ax_n + c) \mod m.$$

Nesta expressão, a, c e m são constantes escolhidas de forma criteriosa. x_0 seria a semente deste gerador e os valores $x_1, x_2, \cdots, x_N$ formam uma sequência de valores que podem ter propriedades muito semelhantes às que esperamos para números aleatórios dependendo dos valores escolhidos para as constantes a, c e m.

Outro ponto relevante que devemos levar em consideração é que da forma expressa acima, o algoritmo gera valores inteiros no intervalo $[0, m)$. Pare para pensar um pouco o porquê disso! Um fato é que se tivermos números no intervalo $[0, 1)$, podemos reescalá-los para diferentes intervalos de forma simples e direta. Por exemplo, se precisarmos escolher entre 10 diferentes possibilidades, basta tomarmos a parte inteira de $(10 * r + 1)$, onde r é um número aleatório no intervalo $[0, 1)$; se precisarmos de um valor real no intervalo $[-7.32, 6.80)$, basta calcularmos $(14.12 * r - 7.32)$. No gerador Linear Congruencial, para obtermos um número no intervalo $[0, 1)$, basta calcularmos x_n/m.

Mas, afinal de contas, o que esperamos de um número aleatório? Pare para pensar alguns minutos antes de continuar.

Para te ajudar a chegar a respostas à questão acima, sugiro testar algumas propriedades de alguns geradores de números aleatórios na tentativa de entender melhor o que é algo realmente aleatório. Vamos testar 4 tipos de geradores diferentes:

- Linear Congruencial

- Mapa Logístico (vocês se lembram que para alguns valores do parâmetro de controle r o comportamento era imprevisível?)
- PCG94
- MT19932

Os geradores linear congruencial e mapa logístico serão implementados por vocês, enquanto o PCG94 é o gerador default do numpy e o MT19932 é um gerador que também está disponível no numpy. Os testes que proponho realizar para cada gerador são os seguintes:

- Histograma dos valores gerados para diferentes números de bins (sugiro o comando `plt.hist` do matplotlib)
- Valor esperado e desvio padrão para diferentes tamanhos de sequências (vocês podem usar os comandos `np.mean` e `np.std` do numpy)
- Gráfico dos valores de x_k por x_{k+1} (aqui vocês devem atribuir o valor x_n à coordenada y e o valor seguinte, x_{n+1}, à coordenada x de pontos num gráfico. Desta forma, os valores com índices pares correspondem à coordenada y enquanto os valores com índices ímpares correspondem à coordenada x. Utilize `plt.scatter` para plotar os pontos num gráfico)
- Cálculo de autocorrelações (mais informações à baixo)

Dentre os geradores a se testar, considerem os seguintes parâmetros para o gerador linear congruencial:

- $a = 16807, \quad c = 0, \quad m = 2^31 - 1, \quad x_0 = 3141549$ (Park-Muller)
- $a = 5, \quad c = 0, \quad m = 2^7, \quad x_0 = 1$

Para o mapa logístico, utilize um valor de r onde você espera um comportamento caótico.

Para usar os geradores PCG94 e MT19932 recomento utilizar a seguinte sintaxe:

```
rng = np.random.default_rng(seed=42)
r=rng.random(N)
```

para o PCG94. Para o MT19932 a sintaxe recomendada é

```
sg = np.random.SeedSequence(1234)
bg = np.random.MT19937(sg)
rg = np.random.Generator(bg)
r=rg.random(N).
```

Para calcular as autocorrelações entre os valores gerados, o mais simples é usar a biblioteca: `from statsmodels.graphics import tsaplots` e o comando `tsaplots.plot_acf(r, lags=200, fft=False, zero=True, title='Autocorrelação')` onde `r` é uma lista com os valores dos números aleatórios gerados. Neste comando, a flag `lags` é o tempo máximo que será utilizado para determinar as autocorrelações.

Recomendo gerar sequências grandes, da ordem de milhares ou até mesmo milhões de valores para fazer as análises. Normalmente os geradores de números aleatórios são rápidos, muitas vezes podendo até ser paralelizados.

A título de informação, o NIST (National Institute of Standards and Technology) disponibiliza uma suíte para testes de geradores de números pseudo-aleatórios (https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf)

Recomendo fortemente que façam essa parte da atividade sobre números aleatórios, no entanto, a entrega dessa atividade não será cobrada.

2 O que é Monte Carlo?

Sistemas de relativamente poucas variáveis tais como o pêndulo duplo e o modelo logístico discutidos anteriormente frustram nossa capacidade de previsão mesmo quando resolvemos esses sistemas numericamente. Assim, para estudar sistemas cada vez mais complexos, é necessário recorrer ao uso de métodos mais poderosos.

Em sistemas que contenham um número muito grande de entes ou onde o comportamento não seja completamente determinístico, o uso de métodos estatísticos se torna imprescindível. Talvez, o mais conhecido deles é o **método de Monte Carlo** que faz uso da lei dos grandes números.

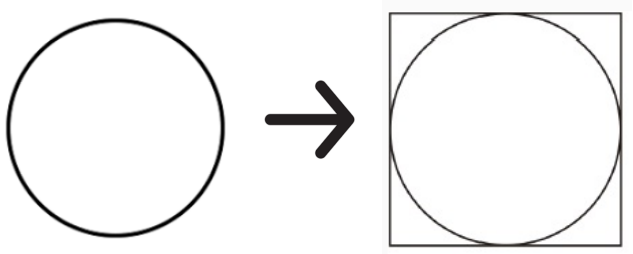
A origem do método de Monte Carlo remonta, em sua versão mais simples, à determinação do número π por Buffon no século XVIII. Avanços mais significativos e a formalização do método surgiram na década de 40 em conexão com o projeto Manhattan (projeto de desenvolvimento da bomba atômica pelos EUA). Quem cunhou o nome Monte Carlo para o método que utilizava números aleatórios para amostrar distribuições de probabilidades foi Nicholas Metropolis para fornecer um codinome ao projeto secreto que vinha sendo desenvolvido por Ulam e von Neumann. O nome é uma referência ao famoso cassino de Monte Carlo no principado de Mônaco, onde a aleatoriedade desempenha um papel relevante nos ganhos dos apostadores. Hoje em dia o termo Monte Carlo é empregado para designar uma imensa gama de métodos estatísticos baseados no uso de números aleatórios.

Algoritmos de Monte Carlo costumam resolver três tipos de problema: integração, otimização e amostragens de distribuições de probabilidades. Todavia, eles partem de um mesmo princípio: **usar muitas configurações aleatórias** chamadas *samples* com o auxílio de números aleatórios. Ao longo do curso exploraremos diferentes tipos de métodos de Monte Carlo. Discutiremos primeiro formas simples de se calcular integrais usando este método.

3 Estimando π (amostragem direta)

Não é novidade para ninguém que $\pi = 3,14159265...$ que por sua vez é a razão entre o perímetro e o diâmetro de um círculo. Resultados cada vez mais precisos podem ser obtidos por medidas cada vez mais precisas. No entanto, não podemos nos esquecer que qualquer processo de medida está sujeito à incertezas, que se propagam para os valores calculados. Mas, se não soubéssemos que $\pi = C/D$, haveria outras formas de determiná-lo?

Olhar para a área do círculo pode ser mais adequado do que olhar para o seu comprimento se quisermos estimar π. Sabemos que $A = \pi R^2$ onde R é o raio do círculo. Se consideramos um círculo unitário, sua área é exatamente π. A primeira coisa que podemos fazer é inscrevê-lo em um quadrado.



Como o círculo tem raio unitário, o lado do quadrado será 2 e a área do quadrado é 4 (estamos considerando unidades arbitrárias). O próximo passo é "lançar" dentro do quadrado aleatoriamente um número muito grande de "dardos", i.e., escolher um numero grande N_{total} de pares (x, y) dentro do quadrado e contabilizar quais deles caíram dentro do círculo. Pela lei dos grandes números podemos afirmar o seguinte sobre a proporção das áreas:

$$\frac{A_{circulo}}{A_{quadrado}} = \frac{\pi}{4} \approx \frac{N_{dentro}}{N_{total}} \longrightarrow \pi \approx 4 \times \frac{N_{dentro}}{N_{total}}.$$

É claro que se o número N_{total} for baixo a nossa estimativa de π é grosseira. Considere, por exemplo, $N_{total} = 2$, se um ponto for sorteado dentro e outro fora do círculo teríamos que $\pi \approx 2$, e se ambos forem sorteados dentro teríamos que $\pi \approx 4$. Mas observe o que acontece para N_{total} a partir de 100 na Figura 1 Obtivemos $\pi \approx 3.36$ para $N_{total} = 100$, $\pi \approx 3.152$ para $N_{total} = 1000$ e $\pi = 3.148$ para $N_{total} = 10000$. Um fato que não podemos esquecer é que este é um método probabilístico. Se realizarmos novamente o experimento com $N_{total} = 100$, dificilmente obteríamos o mesmo valor, $\pi \approx 3.36$. Assim, temos que ter em mente que as estimativas obtidas estão sujeitas a erros estatísticos que podem ser estimados ao repetirmos

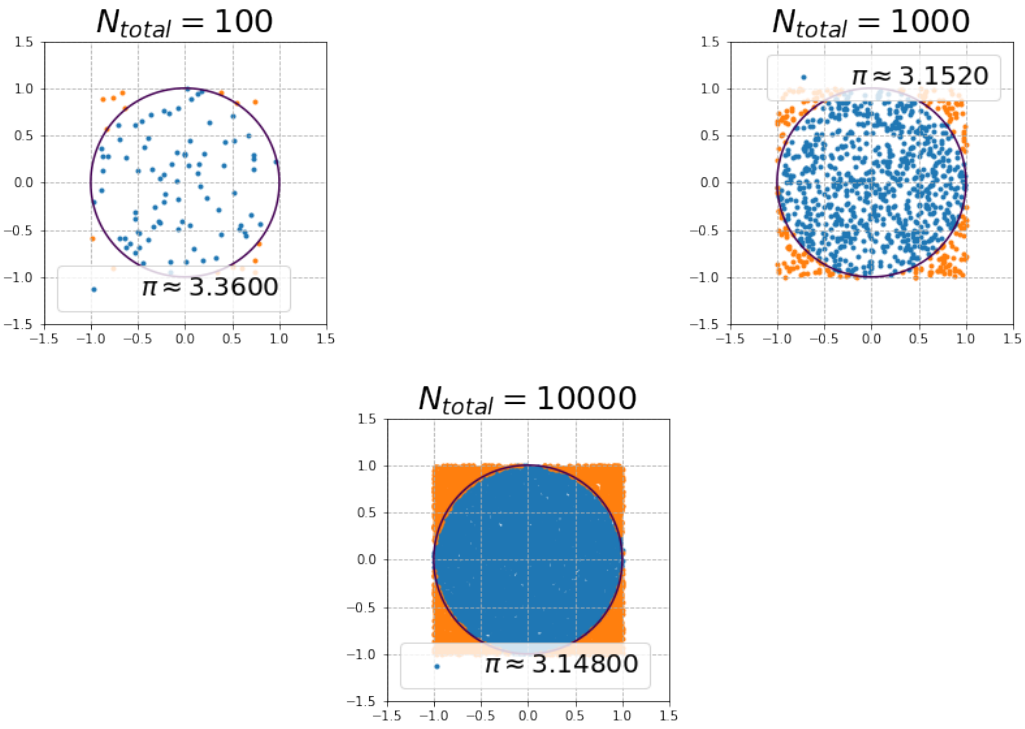


Figure 1: Estimativas de π para $N = 10^2, 10^3, 10^4$

um mesmo experimento muitas vezes. Além disso, quanto maior for N_{total} mais precisa deve ser a estimativa de π e estes fatos podem ser testados da seguinte forma:

1. Obtenha $N_{amos} = 10000$ estimativas diferentes de π para um valor fixo de $N_{total} = 100$ dardos jogados no círculo;
2. Faça um histograma desses valores;
3. Faça a mesma coisa para $N_{total} = 1000$ e $N_{total} = 10000$.

Os resultados são mostrados na Figura 2:

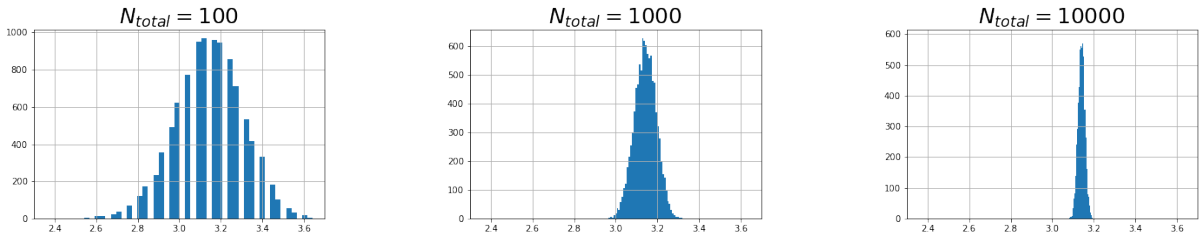


Figure 2: Histogramas de estimativas de π para $N = 10^2, 10^3, 10^4$

Observe que quanto maior o número de dardos jogados, mais centrada é a distribuição de estimativas em torno do valor real de π. Além disso, a largura da distribuição também diminui, indicando uma redução no erro estatístico associado ao valor calculado. Quanto maior for o número de experimentos realizados, menor será a incerteza estatística no valor obtido. O cálculo de π mostrado aqui é um dos exemplos mais didáticos para se introduzir o método de **Monte Carlo** em sua versão mais simples, a amostragem direta (Direct Sampling), e pode ser expandido para calcular integrais em geral em quantas dimensões for necessário.

3.1 Calculando integrais por amostragem direta - Método 1

O mesmo procedimento que foi feito para calcular π também pode ser feito para calcular numericamente integrais definidas, $\int_{a_1}^{a_2} f(x)$. A diferença é que ao invés de escolher pontos (x, y) aleatoriamente em um quadrado de área 4 e contar os (x, y) t.q $x^2 + y^2 < 1$, você os atirará em um retângulo de lados $a_2 - a_1$ e $\max(f(x))$, $a_1 < x < a_2$ e contará os pontos (x, y) t.q $y < f(x)$. Ao usar esse método, um cuidado especial deve ser tomado com funções que apresentam valores negativos no domínio considerado. Neste caso, pode-se escolher pontos também no eixo y negativo, tomando o cuidado de subtrair os pontos que ficaram entre a curva da função e o eixo x. Outro ponto muito relevante é a eficiência do método, que depende fortemente da forma da função e da escolha feita para o retângulo onde os pontos serão sorteados. De fato, se a função apresenta picos ou vales agudos, pode ocorrer de uma fração muito grande dos pontos caírem sempre dentro ou sempre fora da região de interesse, tornando o método pouco eficiente e pouco preciso.

4 Estimando π - O método de Buffon

O método utilizado por Buffon no século XVIII é significantemente diferente do apresentado acima e pode nos levar a uma compreensão mais profunda do método de Monte Carlo. A ideia é jogar sobre uma superfície que contenha linhas paralelas igualmente espaçadas por uma distância t, as linhas vermelhas da figura 3, agulhas de comprimento $l < t$. Sendo o processo aleatório, podemos chegar às seguintes distribuições de probabilidades para os valores de x, que é a distância do centro da agulha à linha mais próxima, e ϕ, que é o ângulo que a agulha faz com a direção das linhas paralelas:

$$P(x) = \begin{cases} \frac{2}{t}, & \text{para} \quad 0 \leq x \leq t/2, \\ 0, & \text{caso contrário;} \end{cases} \tag{1}$$

$$P(\phi) = \begin{cases} \frac{2}{\pi}, & \text{para} \quad 0 \leq \phi \leq \pi/2, \\ 0, & \text{caso contrário.} \end{cases} \tag{2}$$

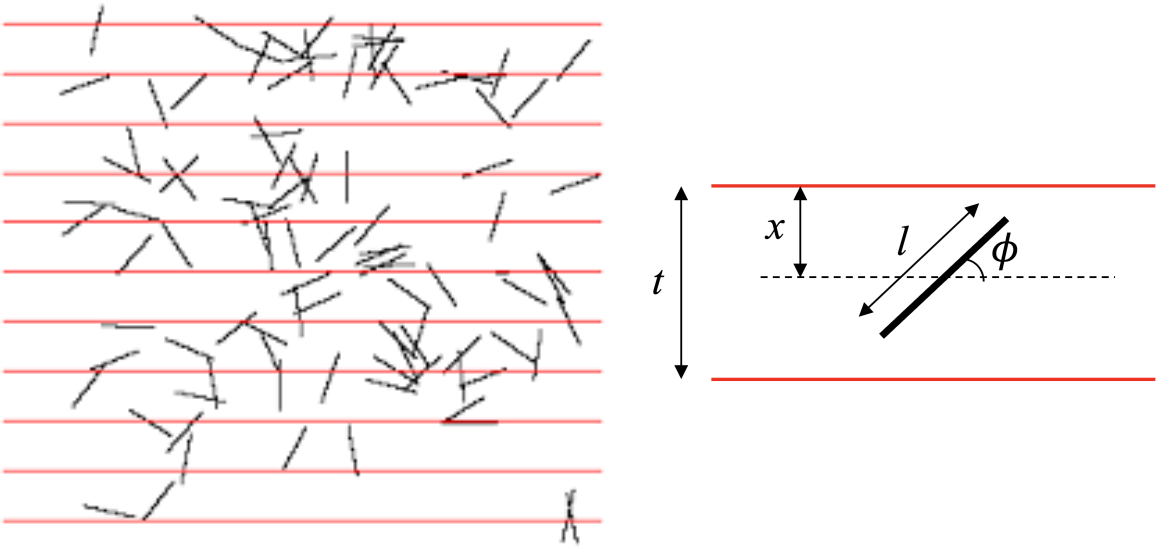


Figure 3: Ilustração do método de Buffon para estimar π.

Como as distribuições dos valores de x e phi são independentes, a probabilidade conjunta é $P(x, phi) =$

$P(x)P(\phi)$, ou seja,

$$P(x,\phi) = \begin{cases} \frac{4}{\pi t}, & \text{para} \quad 0 \leq x \leq t/2 \quad \text{e} \quad 0 \leq \phi \leq \pi/2 \\ 0, & \text{caso contrário.} \end{cases} \tag{3}$$

Perceba que a agulha cruza a linha caso $x \leq \frac{l}{2}\sin\phi$. Assim, a probabilidade de uma agulha cruzar a linha pode ser facilmente calculada como mostrado abaixo:

$$P = \int_{\phi=0}^{\frac{\pi}{2}} \int_{x=0}^{\frac{l}{2}\sin\phi} \frac{4}{\pi t} dx d\phi = \frac{2l}{\pi t}. \tag{4}$$

Com este resultado, podemos estimar o valor de π lançando ou espalhando de forma aleatória um grande número de agulhas, N, e contando quantas cruzam as linhas h. A razão entre estes dois valores fornece uma estimativa não enviesada da probabilidade que calculamos acima que é melhor quanto maior for o número de agulhas lançadas.

5 Calculando integrais por amostragem direta - Método 2

Tomando como inspiração os métodos apresentado anteriormente e o método do trapézio para cálculo numérico de integrais definidas, podemos definir um método de Monte Carlo alternativo para o cálculo de integrais. A ideia é partir da própria definição do valor médio de uma função:

$$\langle f(x) \rangle = \frac{1}{a_2 - a_1} \int_{a_1}^{a_2} f(x) dx. \tag{5}$$

Desta forma, bastaria encontrar o valor médio da função, $\langle f(x) \rangle$, fazendo uma amostragem aleatória dos seus valores. Assim, considerando um conjunto de N valores de x escolhidos de forma aleatória e uniformemente distribuídos no intervalo (a_1, a_2), $\{x_i\}$, podemos estimar o valor médio da função e, consequentemente o valor da integral fazendo:

$$\int_{a_1}^{a_2} f(x) dx = (a_2 - a_1)\langle f(x) \rangle = \frac{a_2 - a_1}{N} \sum_{i=1}^{N} f(x_i). \tag{6}$$

Desta forma, evitamos alguns dos problemas levantados anteriormente. Ainda poderíamos melhorar o método fazendo uma escolha dos valores de x que leve em consideração, de alguma forma, os valores da função, amostrando de forma mais efetiva as regiões que contribuem mais para a integral. Esse método, no entanto, está fora do nosso escopo inicial e será discutido mais a frente.

6 Cálculo de integrais múltiplas

Pelo exposto na seção anterior, pode-se notar que não há elementos suficientes para acreditarmos que o método 2, mesmo sendo mais eficiente que o método 1, seja muito superior à regra do trapézio ou da regra de 3/8 de Simpson, por exemplo. Então, por que deveríamos nos preocupar em aprender métodos de Monte Carlo na resolução de Integrais? A resposta é relativamente simples: Enquanto para integrais unidimensionais, i.e., aquelas que envolvem apenas uma variável de integração, a vantagem do método de Monte Carlo não seja evidente, para integrais multidimensionais a situação é diferente. A amostragem aleatória em integrais multidimensionais é capaz de fornecer resultados precisos a um custo computacional muito menor que o necessário ao usar métodos como a regra do trapézio ou 3/8 de Simpson. Neste caso, são escolhidos N pontos aleatórios em d dimensões e o valor médio da função pode então ser calculado e o valor da integral estimado. Matematicamente, considerando uma integral em 3 dimensões,

$$\int_{a_1}^{a_2} \int_{b_1}^{b_2} \int_{c_1}^{c_2} f(x_1, x_2, x_3) dx_1 dx_2 dx_3 = (a_2 - a_1)(b_2 - b_1)(c_2 - c_1)\langle f(x_1, x_2, x_3) \rangle \tag{7}$$

$$\langle f(x_1, x_2, x_3)\rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_1^i, x_2^i, x_3^i). \tag{8}$$

A generalização para problemas envolvendo d dimensões é imediata.

6.1 Tarefa:

Resolva, usando o método de Monte Carlo por amostragem direta, as seguintes integrais:

1. $$\int_{-3}^{3} e^{-x^2}\mathrm{d}x$$

2. $$\int_{-10}^{10} e^{-x^2}\mathrm{d}x$$

3. $$\int_{1}^{2}\int_{0}^{1}\int_{-1}^{1} \frac{x_1 x_2^2}{x_3}\mathrm{d}x_1\mathrm{d}x_2\mathrm{d}x_3$$

4. $$\int_0^1 \cdots \int_0^1 \frac{1}{(\vec{r}_1+\vec{r}_2)\cdot\vec{r}_3} d\vec{r}_1 d\vec{r}_2 d\vec{r}_3 = \int_0^1 \cdots \int_0^1 \frac{1}{((x_1+x_2)x_3+(y_1+y_2)y_3+(z_1+z_2)z_3)} dx_1 dy_1 dz_1 dx_2 dy_2 dz_2 dx_3 dy_3 dz_3$$

- Para o cálculo das integrais 1 e 2, além do valor obtido para a integral, apresente os histogramas para $N_{total} = 100$, 1000 e 10000 pontos lançados considerando $N_{amos} = 1000$ amostras diferentes, ou seja, obtenha 1000 estimativas diferentes para o valor da integral considerando, por exemplo, que $N_{total} = 100$ pontos são sorteados em cada uma destas amostras.
- Calcule o valor da integral 4, porém, não é necessário usar o número de pontos e amostras descritos acima.
- Determine, em todos os casos, o erro estatístico associado ao valor que você obteve.
- Apresente seus resultados de forma clara, indicando o valor calculado para a integral e o respectivo erro estatístico. Comente os resultados.

Dica: Para as integrais 3 e 4 você deve fazer alguns testes, que não precisam ser apresentados, para determinar um número razoável de pontos a serem sorteados (N_{total}) e de amostras a serem usadas (N_{amos}).

7 Referências:

- B.A. Stickler e E. Schachinger, Basic Concepts in Computational Physics, Springer (2016)
- D.P. Landau e K. Binder, A guide to Monte Carlo Simulations in Statistical Physics, Cambridge University Press (2014)
- W. Krauth, Statistical Mechanics Algorithms and Computations, Oxford (2006)
- Approximating Pi (Monte Carlo integration) — animation https://www.youtube.com/watch?v=ELetCV_wX_c
- Monte Carlo Integration In Python https://www.youtube.com/watch?v=WAf0rqwAvgg&t=324s
- Monte Carlo Method https://en.wikipedia.org/wiki/Monte_Carlo_method
- Buffon's needle experiment https://www.youtube.com/watch?v=sJVivjuMfWA