ENTREGA DE EXERCÍCIOS COMPUTACIONAIS - 2º BIMESTRE

MARIA EDUARDA MESQUITA MAGALHÃES

3 - Sejam Xi (i ∈ N) v.a's Bernoulli com p = 0.5.

a) Prove que $X = \sum_{i=1}^{\infty} 2^{-i} X_i$ tem distribuição uniforme em [0,1].

Para provar que a variável (X) definida como:

$$X = \sum_{i=1}^{\infty} \frac{X_i}{2^i}$$

onde X_i são variáveis aleatórias de Bernoulli com (p = 0.5), tem distribuição uniforme no intervalo [0, 1], podemos seguir o seguinte raciocínio:

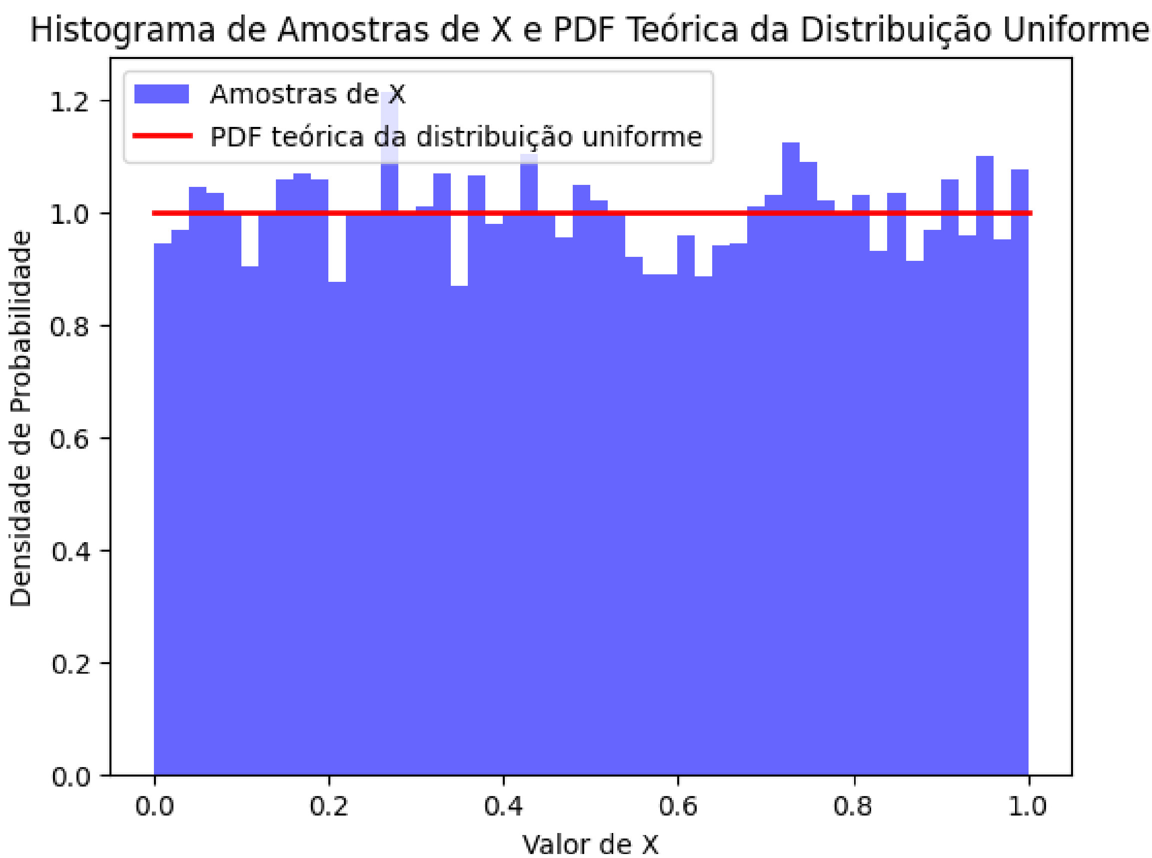
1. Cada X_i é uma variável aleatória de Bernoulli com parâmetro (p = 0.5), o que significa que X_i pode assumir os valores 0 ou 1, com igual probabilidade.
2. Portanto, cada termo na soma $X_i / 2^i$ assume um valor no conjunto {0, 1/2, 1/4, 1/8,...}. Esses valores são potências de (1/2) e, portanto, podem ser vistos como uma representação binária do número no intervalo ([0, 1]).
3. A soma desses termos resulta em uma soma infinita de potências decrescentes de (1/2), que é exatamente a representação de um número real no intervalo ([0, 1]) em base binária.
4. Dado que cada X_i é independente das outras e segue uma distribuição de Bernoulli, e que estamos somando termos que representam a parte fracionária de um número em binário, a soma (X) segue uma distribuição uniforme no intervalo [0, 1]).

Portanto, a variável (X) tem, de fato, uma distribuição uniforme no intervalo ([0, 1]).

Para provar que $ X = \sum_{i=1}^{\infty} \frac{X_i}{2^i} $ tem distribuição uniforme em ([0,1]), podemos simular a geração de X e plotar um histograma das amostras para verificar se elas seguem uma distribuição uniforme. Este código irá gerar (10,000) amostras de X e, em seguida, plotará um histograma dessas amostras. Se o histograma se parecer com uma distribuição uniforme no intervalo ([0,1]), isso fornecerá evidências de que X tem distribuição uniforme nesse intervalo.

```
# Agora, vamos mostrar esse resultado computacionalmente com python
```

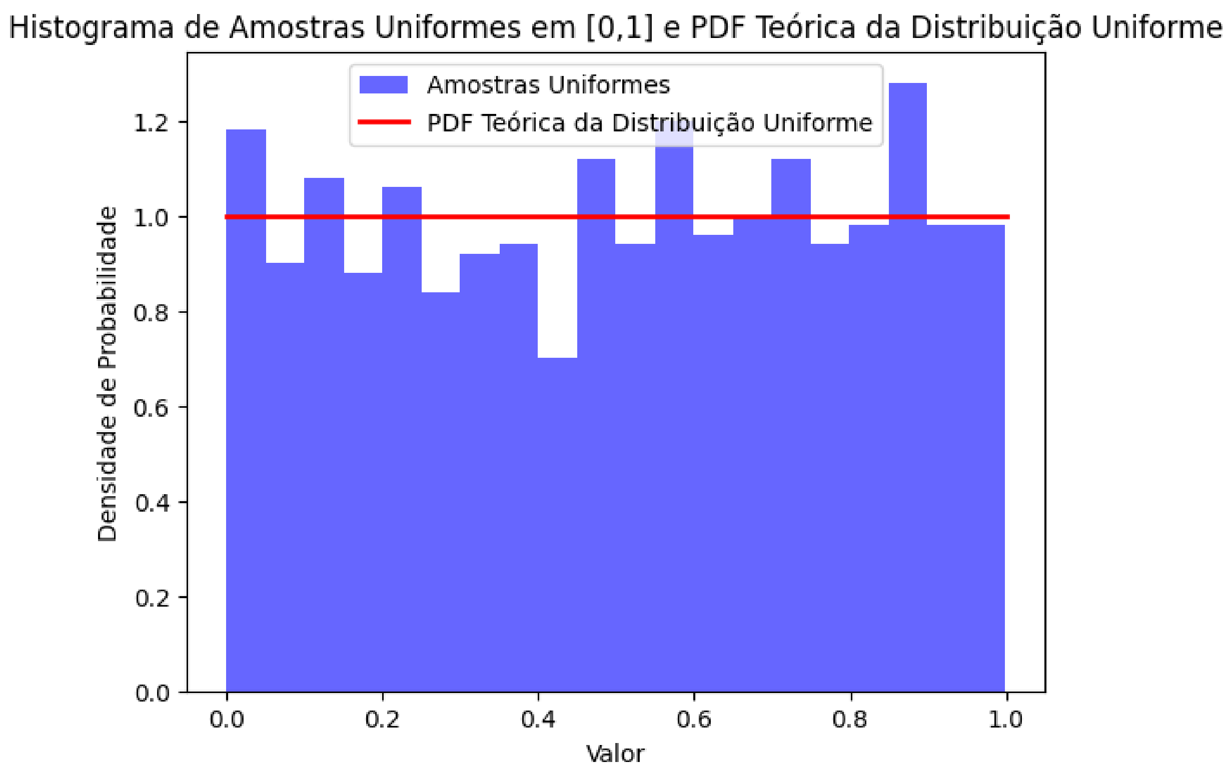
```
import numpy as np
import matplotlib.pyplot as plt

# Definiremos o número de amostras (escolha arbitrária)
num_samples = 10000

# Agora geraremos amostras de X
X_samples = np.zeros(num_samples)
for i in range(num_samples):
    X = 0
    for j in range(1, 55):  # Somar 54 termos para garantir precisão
suficiente
        X += np.random.randint(0, 2) / (2 ** j)
    X_samples[i] = X

# Plotaremos histogramas da amostra
plt.hist(X_samples, bins=50, density=True, alpha=0.6, color='b',
label='Amostras de X')

# Plotar a função de densidade de probabilidade (PDF) teórica da
distribuição uniforme
x = np.linspace(0, 1, 1000)
pdf_uniform = np.ones_like(x)  # A PDF da distribuição uniforme é 1 em
todo o intervalo [0, 1]
plt.plot(x, pdf_uniform, 'r-', lw=2, label='PDF teórica da
distribuição uniforme')

plt.title('Histograma de Amostras de X e PDF Teórica da Distribuição
Uniforme')
plt.xlabel('Valor de X')
plt.ylabel('Densidade de Probabilidade')
plt.legend()
plt.show()
```



b) Amostre n = 1000 variáveis uniformes em [0,1] usando seu gerador e verifique se seguem a distribuição correta ao plotar o histograma usando o Matplotlib ou Seaborn.

```
import numpy as np
import matplotlib.pyplot as plt

# Função para gerar amostras uniformemente distribuídas
def generate_uniform(n):
    samples = np.zeros(n)
    for i in range(n):
        sample = 0.0
        for j in range(1, 55):  # Somar 54 termos para garantir
precisão
            X_i = np.random.randint(0, 2)  # Gerar uma amostra de
Bernoulli com p=0.5
            sample += X_i / 2**j
        samples[i] = sample
    return samples

# Amostrar 1000 variáveis uniformes em [0, 1]
n = 1000
uniform_samples = generate_uniform(n)
```

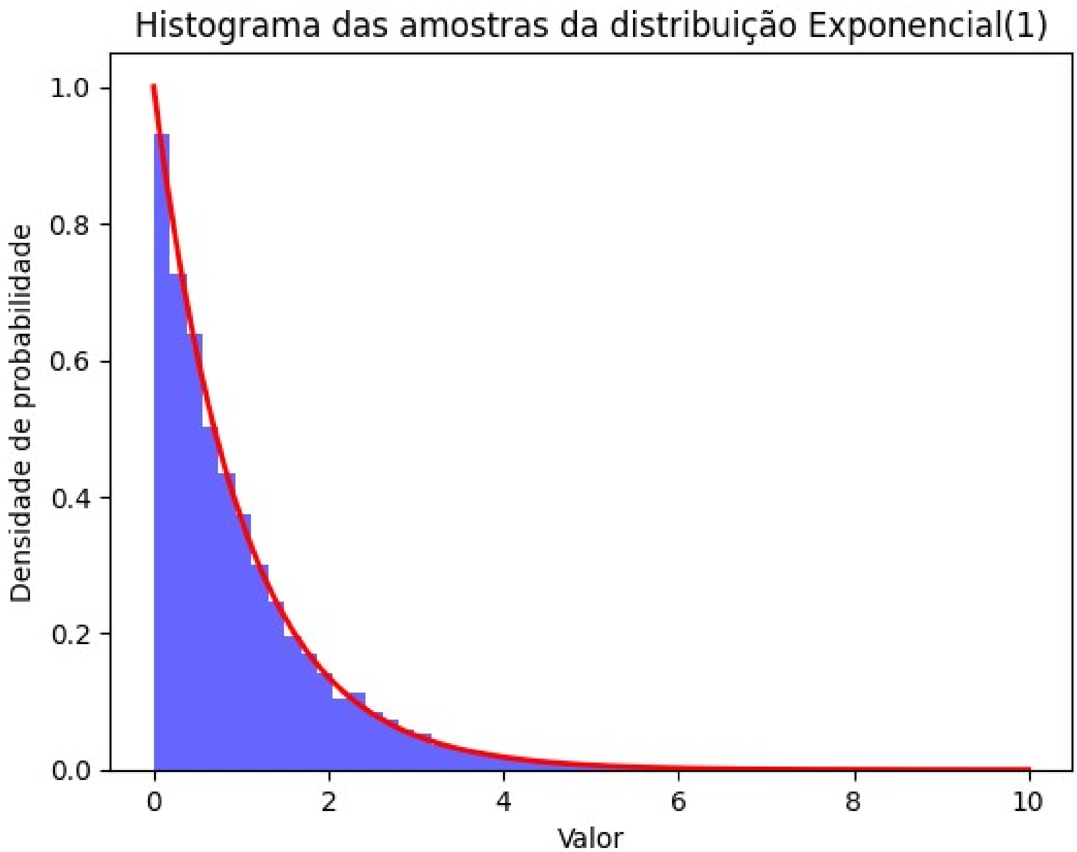
```
# Plotar histograma
plt.hist(uniform_samples, bins=20, density=True, alpha=0.6, color='b',
label='Amostras Uniformes')

# Plotar a função de densidade de probabilidade (PDF) teórica da
distribuição uniforme
x = np.linspace(0, 1, 1000)
pdf_uniform = np.ones_like(x)  # A PDF da distribuição uniforme é 1 em
todo o intervalo [0, 1]
plt.plot(x, pdf_uniform, 'r-', lw=2, label='PDF Teórica da
Distribuição Uniforme')

plt.title('Histograma de Amostras Uniformes em [0,1] e PDF Teórica da
Distribuição Uniforme')
plt.xlabel('Valor')
plt.ylabel('Densidade de Probabilidade')
plt.legend()
plt.show()
```



4 - Como discutido em aula, tendo um gerador de números uniformemente distribuídos, podemos obter amostras de várias outras distribuições.

a) Usando a transformação quantil, crie um gerador para X ~ Expo(1); plote um histograma para verificar seus resultados.

```
import numpy as np

# Definiremos o número de amostras (escolha arbitrária)
num_samples = 10000

# Geraremos amostras uniformemente distribuídas
uniform_samples = np.random.uniform(0, 1, num_samples)

# Aplicando a transformação quantil para obter amostras da
distribuição Exponencial
exponential_samples = -np.log(1 - uniform_samples)

# Plotando o histograma das amostras
plt.hist(exponential_samples, bins=50, density=True, alpha=0.6,
color='b')

# Plotando a função densidade de probabilidade teórica da distribuição
Exponencial
x = np.linspace(0, 10, 1000)
pdf_exponential = np.exp(-x)  # λ = 1
plt.plot(x, pdf_exponential, 'r-', lw=2)

plt.title('Histograma das amostras da distribuição Exponencial(1)')
plt.xlabel('Valor')
plt.ylabel('Densidade de probabilidade')
plt.show()
```



O código começa com a importação das bibliotecas necessárias: numpy para geração de números aleatórios e operações matemáticas, e matplotlib.pyplot para visualização dos dados. Em seguida, definimos o número de amostras a serem geradas e criamos 10.000 amostras de uma distribuição uniforme U (0,1), usando np.random.uniform(0, 1, num_samples).

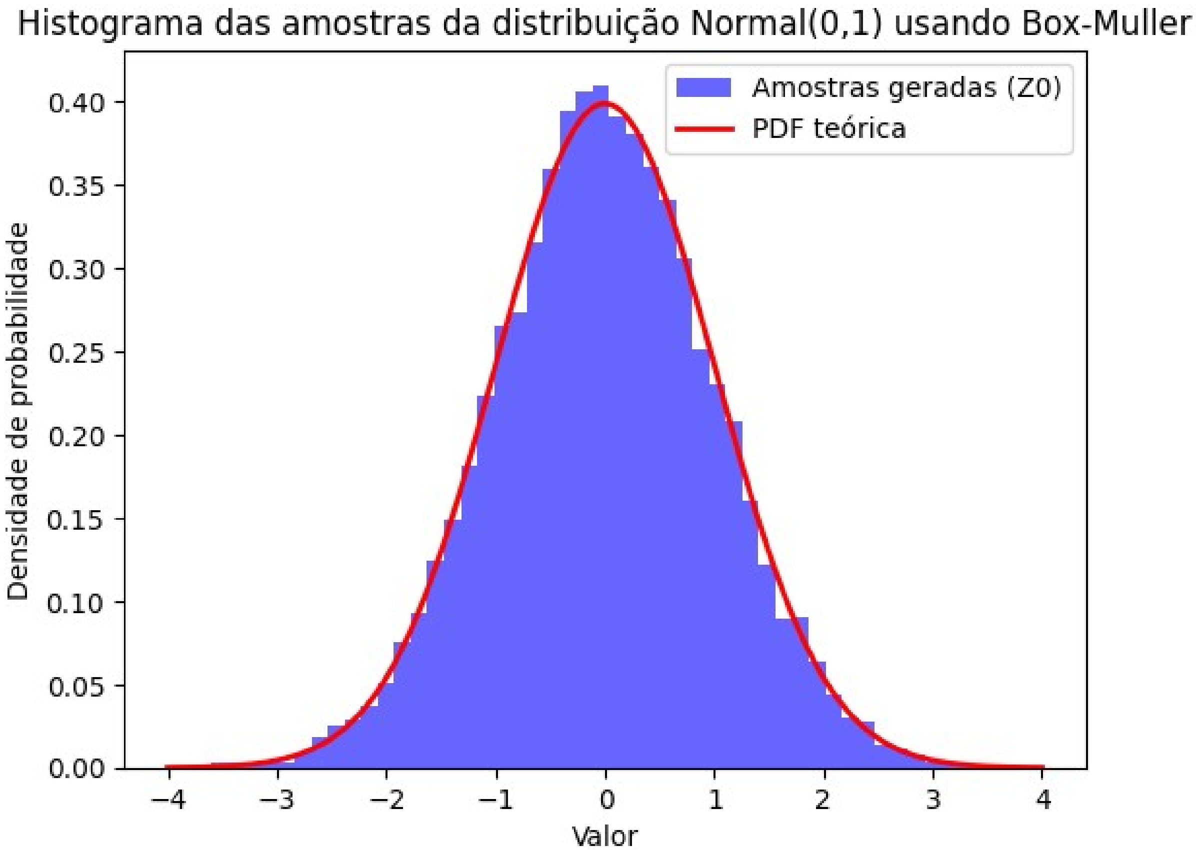
Para transformar essas amostras uniformemente distribuídas em amostras da distribuição Exponencial(1), aplicamos a transformação quantil. Esta transformação é baseada na inversa da função de distribuição acumulada (CDF) da distribuição Exponencial. A fórmula aplicada é $X = -\ln(1-U)$, onde U são as amostras da distribuição uniforme. O resultado é armazenado em exponential_samples.

Daí, o histograma das amostras geradas é plotado usando plt.hist, com o parâmetro density=True para normalizar o histograma. Adicionamos também a função densidade de probabilidade teórica da distribuição Exponencial(1) usando np.exp(-x). A curva teórica é traçada em vermelho e comparada com o histograma das amostras geradas. A boa correspondência entre a curva teórica e o histograma confirma que a transformação quantil foi aplicada corretamente.

b) Usando a transformação de Box-Muller, crie um gerador para X ∼ N(0,1); plote um histograma para verificar seus resultados

```
import numpy as np
# Número de amostras
```

```python
num_samples = 10000

# Gerar amostras uniformemente distribuídas
U1 = np.random.uniform(0, 1, num_samples)
U2 = np.random.uniform(0, 1, num_samples)

# Aplicar a transformação de Box-Muller
Z0 = np.sqrt(-2 * np.log(U1)) * np.cos(2 * np.pi * U2)
Z1 = np.sqrt(-2 * np.log(U1)) * np.sin(2 * np.pi * U2)

# Plotar o histograma das amostras Z0
plt.hist(Z0, bins=50, density=True, alpha=0.6, color='b',
label='Amostras geradas (Z0)')

# Plotar a função densidade de probabilidade teórica da distribuição
Normal(0,1)
x = np.linspace(-4, 4, 1000)
pdf_normal = (1/np.sqrt(2 * np.pi)) * np.exp(-0.5 * x**2)
plt.plot(x, pdf_normal, 'r-', lw=2, label='PDF teórica')

plt.title('Histograma das amostras da distribuição Normal(0,1) usando
Box-Muller')
plt.xlabel('Valor')
plt.ylabel('Densidade de probabilidade')
plt.legend()
plt.show()
```



Este código Python usa o método de Box-Muller para gerar amostras de uma distribuição normal padrão (média 0, desvio padrão 1) a partir de amostras uniformemente distribuídas. Aqui está uma explicação passo a passo do código:

- Definimos o número de amostras desejadas (num_samples) como 10000.
- Geramos duas sequências de amostras uniformemente distribuídas no intervalo [0, 1] usando np.random.uniform(0, 1, num_samples). Essas amostras são armazenadas em U1 e U2.
- Aplicamos o método de transformação de Box-Muller para converter as amostras uniformemente distribuídas em amostras de uma distribuição normal padrão. Este método usa as seguintes fórmulas:

$$Z_0 = \sqrt{-2\ln(U_1)} \cdot \cos(2\pi U_2)$$

$$Z_1 = \sqrt{-2\ln(U_2)} \cdot \sin(2\pi U_2)$$

onde U_1 e U_2 são as amostras uniformemente distribuídas e Z_0 e Z_1 são as amostras resultantes da distribuição normal padrão. Essas amostras são armazenadas em Z_0e Z_1 .

- Plotamos o histograma das amostras Z_0 usando plt.hist(), com 50 bins, densidade normalizada e transparência de 0.6. Isso representa a distribuição das amostras geradas.
- Plotamos a função densidade de probabilidade (PDF) teórica da distribuição normal padrão (0,1) usando a equação da PDF.
- Adicionamos um título ao gráfico, rótulos dos eixos x e y, e uma legenda para identificar as amostras geradas e a PDF teórica.
- Exibimos o gráfico usando plt.show().

O resultado é um histograma das amostras geradas Z_0 sobreposto à função densidade de probabilidade teórica da distribuição normal padrão. Isso permite visualizar como as amostras se distribuem em relação à distribuição teórica.