**1ra Lista Exercícios**

**Nome:** Adrian Alejandro Chavez Alanes
**Matrícula:** 948

**Github:** https://github.com/aadlrei/TP547-Principios-de-Simulacao-de-Sistemas-de-Comunicacao.git

**1.** Considere um Gerador Linear Congruente (GLC) misto com os seguintes parâmetros:

$$X_{n+1} = (aX_n + c)\ mod\ m$$

$$onde: a = 5; m = 16\ e\ x_0 = 7$$

a) Calcule os cinco primeiros números gerador pelo GLC misto.

$Con\ c = 1$:
$x_0 = 7$
$x_1 = (5 * 7 + 1)\ mod 16 = 36\ mod 16 = 4$
$x_2 = (5 * 4 + 1)\ mod 16 = 21\ mod 16 = 5$
$x_3 = (5 * 5 + 1)\ mod 16 = 26\ mod 16 = 10$
$x_4 = (5 * 10 + 1)\ mod 16 = 51\ mod 16 = 3$
$x_5 = (5 * 3 + 1)\ mod 16 = 16\ mod 16 = 0$

b) Determine o período desse gerador.

Por definição:

O período máximo de um GLC é no máximo igual ao módulo m.

$$período = m = 16$$

Por cálculo:

$x_6 = (5 * 0 + 1)\ mod 16 = 1\ mod 16 = 1$
$x_7 = (5 * 1 + 1)\ mod 16 = 6\ mod 16 = 6$
$x_8 = (5 * 6 + 1)\ mod 16 = 31\ mod 16 = 15$
$x_9 = (5 * 15 + 1)\ mod 16 = 76\ mod 16 = 12$
$x_{10} = (5 * 12 + 1)\ mod 16 = 61\ mod 16 = 13$
$x_{11} = (5 * 13 + 1)\ mod 16 = 66\ mod 16 = 2$
$x_{12} = (5 * 2 + 1)\ mod 16 = 11\ mod 16 = 11$
$x_{13} = (5 * 11 + 1)\ mod 16 = 56\ mod 16 = 8$
$x_{14} = (5 * 8 + 1)\ mod 16 = 41\ mod 16 = 9$
$x_{15} = (5 * 9 + 1)\ mod 16 = 46\ mod 16 = 14$
$x_{16} = (5 * 14 + 1)\ mod 16 = 71\ mod 16 = 7 = x_0$

$$período = 16$$

Por código:

```
import numpy as np

x = 7  # Valor inicial x0
a = 5
c = 1
m = 16
```

```
x1 = [x]  # Cria uma lista con x0 como valor inicial

while True:
    x = (a * x + c) % m  # Gera o próximo número
    if x == x1[0]:  # Se repetir o primeiro valor, terminamos
        break
    x1.append(x)  # Adiciona o novo valor à lista

# Imprimir os valores gerados e o período
print("Valores gerados:", x1)
print("Período:", len(x1))
```

```
Valores gerados: [7, 4, 5, 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14]
Período: 16
```

c) Explique se este GLC misto é adequado para aplicações criptográficas. Justifique sua resposta.

R.- Não é adequado porque tem um período muito curto em que um padrão já pode ser encontrado, tornando-o inadequado para aplicações de segurança. Além disso, não é totalmente aleatório, pois depende das variáveis $a$, $c$, $m$ e do valor inicial $x_0$.

**2.** Em uma central telefônica, o número médio de chamadas recebidas por minuto é igual a 3. Suponha que o número de chamadas recebidas por minuto siga uma distribuição Poisson.

a) Qual é a probabilidade de que exatamente 5 chamadas sejam recebidas em um minuto específico?

Por cálculo:

$$Dados: \lambda = 3, x = 5$$

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

$$P(X = x) = \frac{3^5 e^{-3}}{5!} = \frac{12{,}098}{120} = 0{,}1008$$

b) Qual é a probabilidade de que no máximo 2 chamadas sejam recebidas em um minuto específico?

Por cálculo:

$$Dados: \lambda = 3, x = 2$$

$$P(X \leq 2) = e^{-\lambda} \sum_{i=0}^{x} \frac{\lambda^i}{i!}$$

$$P(X = 0) = \frac{3^0}{0!} = 1 = 1$$

$$P(X = 1) = \frac{3^1}{1!} = 3 = 3$$
$$P(X = 2) = \frac{3^2}{2!} = \frac{9}{2}$$
$$P(X \leq 2) = e^{-3}\left(1 + 3 + \frac{9}{2}\right) = 0{,}4232$$

Por código (a) y (b):

```python
import numpy as np

# Definir parámetros
lambda1 = 3  # Número médio de requisições (média da Poisson)
N = 1000000  # Número de amostras (tamanho da simulação)
# Gerar amostras da distribuição de Poisson usando a transformada inversa
av = np.array([])
x = np.random.uniform(0, 1, N)

for ix in x:
    i = 0
    pr = np.exp(-lambda1)
    F = pr
    while ix >= F:
        pr = lambda1 / (i + 1) * pr
        F = F + pr
        i = i + 1
    av = np.append(av, i)

# Calcular probabilidades
prob_2 = np.sum(av == 5) / N  # P(X = 2)
prob_leq_2 = np.sum(av <= 2) / N  # P(X ≤ 2)

# Mostrar resultados
print(f"P(X = 2) ≈ {prob_2:.4f}")
print(f"P(X ≤ 2) ≈ {prob_leq_2:.4f}")
```

```
P(X = 2) ≈ 0.1006
P(X ≤ 2) ≈ 0.4235
```

**3.** Uma prova objetiva possui 10 questões, e cada questão apresenta 4 alternativas, das quais apenas uma é correta. Um aluno despreparado responde aleatoriamente todas as questões, assinalando uma alternativa por questão.

Considere que X seja a variável aleatória que representa o número de questões acertadas pelo aluno.

$Distribuição\ Binomial, dados: n = 10, \qquad q = \frac{1}{4} = 0{,}25, \qquad B(10; 0{,}25)$

a) Qual é a probabilidade de o aluno acertar exatamente 3 questões.

$$f(x) = \binom{n}{x} q^x (1-q)^{n-x}$$
$$f(3) = \binom{10}{3} 0{,}25^3 (1-0{,}25)^{10-3}$$
$$\binom{10}{3} = \frac{10!}{3!\,(10-3)!} = 120$$
$$0{,}25^3 (1-0{,}25)^7 = 0{,}0020856$$
$$f(3) = 120 * 0{,}0020856 = 0{,}25028$$

b) Qual é a probabilidade de ele acertar no máximo 2 questões.

$$F(2) = P\{X \leq 2\} = \sum_{k=0}^{x} \binom{n}{k} q^k (1-q)^{n-k}$$
$$f(0) = \binom{10}{0} 0{,}25^0 (1-0{,}25)^{10-0} = 0{,}05631$$
$$f(1) = \binom{10}{1} 0{,}25^1 (1-0{,}25)^{10-1} = 0{,}18771$$
$$f(2) = \binom{10}{2} 0{,}25^2 (1-0{,}25)^{10-2} = 0{,}28156$$
$$F(2) = P\{X \leq 2\} = 0{,}05631 + 0{,}18771 + 0{,}28156 = 0{,}5255$$

c) Determine a média e o desvio padrão da variável aleatória X.

$Média$:

$$\mu = nq = 10 * 0{,}25 = 2{,}5$$

$Variância$:

$$\sigma^2 = nq(1-q) = 2{,}5(1-0{,}25) = 1{,}875$$
$$\sqrt{\sigma^2} = \sqrt{1{,}875} = 1{,}369$$

Por código:

```
import numpy as np
import math

# Parâmetros do problema
n = 10  # Número de perguntas
q = 0.25  # Probabilidade de acerto

# Cálculo da probabilidade de acertar exatamente 3 perguntas
# Combinação para escolher 3 acertos de 10 (nCr)
comb_3 = math.factorial(n) / (math.factorial(3) *
math.factorial(n - 3))  # Combinação manual
p_3 = comb_3 * (q ** 3) * ((1 - q) ** (n - 3))  # Fórmula da
binomial para P(X=3)
```

```
print(f"P(X=3) = {p_3:.5f}")  # Exibe a probabilidade de
acertar exatamente 3

# Cálculo da probabilidade de acertar no máximo 2 perguntas
(P(X ≤ 2))
p_max_2 = 0
for k in range(3):  # k vai de 0 até 2 (inclusive)
    # Calculando a combinação e a probabilidade de acerto para
cada valor de k
    comb_k = math.factorial(n) / (math.factorial(k) *
math.factorial(n - k))
    p_k = comb_k * (q ** k) * ((1 - q) ** (n - k))  #
Probabilidade para um valor k específico
    p_max_2 += p_k  # Somando as probabilidades para P(X≤2)
print(f"P(X≤2) = {p_max_2:.5f}")  # Exibe a probabilidade de
acertar no máximo 2

# Cálculo da média (μ) e da desvio padrão (σ) para a
distribuição binomial
media = n * q  # Média de uma distribuição binomial
varianza = n * q * (1 - q)  # Variância da distribuição
binomial
desviacao_estandar = np.sqrt(varianza)  # Desvio padrão

# Exibindo os resultados
print(f"Media (μ) = {media:.2f}")
print(f"Desvio padrão (σ) = {desviacao_estandar:.5f}")
```

```
P(X=3) = 0.25028
P(X≤2) = 0.52559
Media (μ) = 2.50
Desvio padrão (σ) = 1.36931
```

**4.** Se ocorrerem falhas de energia elétrica de acordo com uma distribuição de Poisson com uma média de 6 falhas a cada duas semanas, calcule a probabilidade de que haverá ao menos 3 falhas durante uma semana específica. Traçar o histograma da variável analisada.

Por cálculo:

$Dados: \lambda = \frac{6}{2} = 3, x = 3$

$$P(X \geq 3) = 1 - P(X \leq 2)$$

$$P(X \geq 3) = e^{-\lambda} \sum_{i=0}^{x} \frac{\lambda^i}{i!}$$

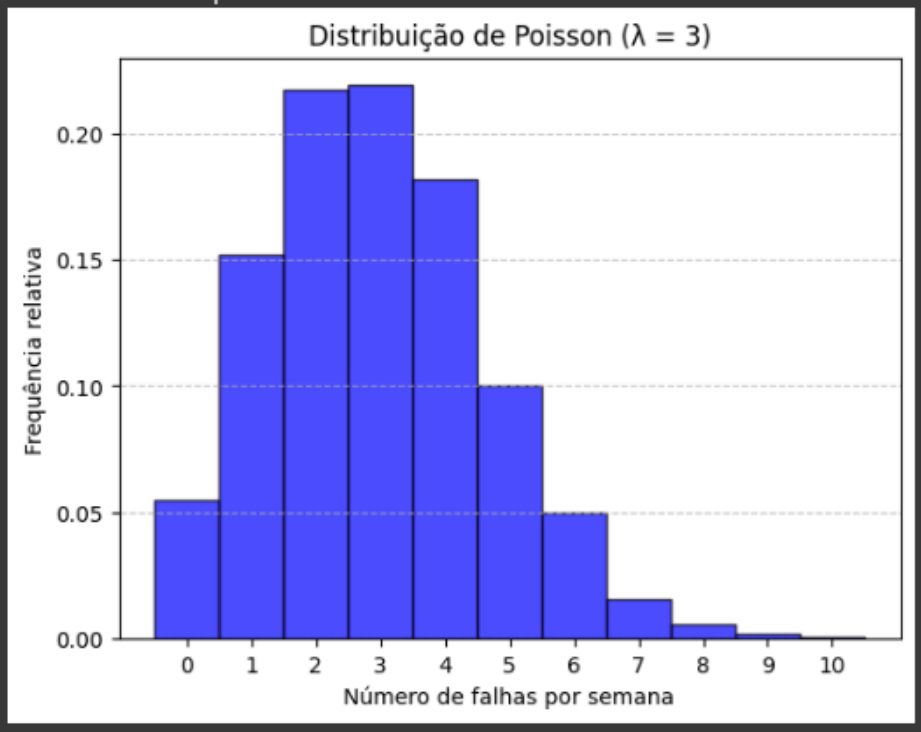
$$P(X = 0) = \frac{3^0}{0!} = 1 = 1$$
$$P(X = 1) = \frac{3^1}{1!} = 3 = 3$$
$$P(X = 2) = \frac{3^2}{2!} = \frac{9}{2}$$
$$P(X \leq 2) = e^{-3}\left(1 + 3 + \frac{9}{2}\right) = 0{,}4232$$
$$P(X \geq 3) = 1 - 0{,}4232 = 0{,}5768$$

Por código:

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

# Definir parâmetros
lambda1 = 3  # Média de falhas por semana
N = 1000  # Número de amostras

# Lista para armazenar os valores gerados
av = np.array([])

# Gerar N valores aleatórios uniformemente distribuídos entre 0 e
1
x = np.random.uniform(0, 1, N)

# Aplicar o método da Transformada Inversa para gerar valores
Poisson
for ix in x:
    i = 0  # Contador de falhas
    pr = np.exp(-lambda1)  # Primeiro termo da distribuição de
Poisson
    F = pr  # Inicializa a soma acumulativa da CDF

    # Encontrar o menor i tal que F >= U (inversão da CDF)
    while ix >= F:
        pr = lambda1 / (i + 1) * pr  # Atualiza a probabilidade
do próximo valor de i
        F = F + pr  # Atualiza a soma acumulada da CDF
        i = i + 1  # Incrementa o número de falhas

    av = np.append(av, i)  # Armazena o número de falhas gerado

# Calcular P(X >= 3) = 1 - P(X <= 2)
prob_X_geq_3 = np.sum(av >= 3) / N

# Imprimir resultado da probabilidade
print(f"Probabilidade de pelo menos 3 falhas em uma semana:
{prob_X_geq_3:.4f}")
```

```
# Graficar histograma
plt.hist(av, bins=np.arange(0, max(av.astype(int)) + 2) - 0.5,
density=True, alpha=0.7, color='b', edgecolor='black')
plt.xlabel("Número de falhas por semana")
plt.ylabel("Frequência relativa")
plt.title("Distribuição de Poisson (λ = 3)")
plt.xticks(range(0, int(max(av)) + 1))
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

```
Probabilidade de pelo menos 3 falhas em uma semana: 0.5760
```




**5.** O tempo (em minutos) entre chegadas sucessivas de clientes a um caixa eletrônico pode ser descrito por uma variável aleatória com distribuição exponencial, cuja média é de 2 minutos.

a) Qual é o parâmetro $(\lambda)$ dessa distribuição exponencial?
$Dados: \beta = 2$

$$\beta = \frac{1}{\lambda}$$
$$\lambda = \frac{1}{2}$$

b) Qual é a probabilidade de que o tempo de espera até a chegada do próximo cliente seja inferior a 1 minuto?

$$P(X \leq x) = 1 - e^{-\lambda x}$$

$$P(X \leq 1) = 1 - e^{-\frac{1}{2}} = 0{,}3934$$

c) Qual é a probabilidade de que o tempo de espera até a chegada do próximo cliente seja superior a 4 minutos?

$$P(X \geq 4) = 1 - P(X \leq 3)$$
$$P(X \leq 3) = 1 - e^{-\frac{3}{2}} = 0{,}7768$$
$$P(X \geq 4) = 1 - 0{,}7768 = 0{,}2231$$

Por código:

```python
import numpy as np

# Parâmetro da distribuição exponencial
media = 2  # Média da distribuição
lmbda = 1 / media  # Parâmetro lambda

# Cálculo das probabilidades
x1 = 1
p_x1 = 1 - np.exp(-lmbda * x1)
print(f"Probabilidade de esperar menos de {x1} minuto: P(X ≤ {x1}) = {p_x1:.4f}")

x3 = 3
p_x3 = 1 - np.exp(-lmbda * x3)  # P(X ≤ 3)
p_x2 = 1 - p_x3  # P(X ≥ 4)

print(f"Probabilidade de esperar mais de 4 minutos: P(X ≥ 4) = {p_x2:.4f}")
```

```
Probabilidade de esperar menos de 1 minuto: P(X ≤ 1) = 0.3935
Probabilidade de esperar mais de 4 minutos: P(X ≥ 4) = 0.2231
```

**6.** A distribuição discreta geométrica conta o número de tentativas até o primeiro sucesso. A pmf é dada por $f(x) = p(1-p)^{x-1}$, onde $p$ representa a probabilidade de sucesso e $x$ o número de tentativas. Fazer um algoritmo para a geração das variáveis aleatórias geométricas. Com o algoritmo proposto calcular:

Um jogador participa de um jogo no qual ele lança um dado justo (equilibrado, com 6 faces numeradas de 1 a 6). Ele ganha o jogo assim que o número "5" aparecer pela primeira vez. Considere que os lançamentos são independentes.

Seja X a variável aleatória que representa o número do lançamento no qual o jogador obtém pela primeira vez o número "5".

Dados:

$$p = \frac{1}{6}, \qquad P(X = x) = p(1-p)^{x-1}$$

a) Qual é a probabilidade de que o jogador ganhe o jogo exatamente no terceiro lançamento?

$$P(X = 3) = \frac{1}{6}\left(1 - \frac{1}{6}\right)^{3-1} = 0{,}1157$$

b) Qual é a probabilidade de que ele precise lançar o dado pelo menos 4 vezes para ganhar o jogo?

$$P(X \geq 4) = 1 - P(X \leq 3)$$

$$P(X \leq 3) = p\sum_{i=0}^{x}(1-p)^{i-1}$$

$$P(X = 1) = \left(1 - \frac{1}{6}\right)^{1-1} = 1$$

$$P(X = 2) = \left(1 - \frac{1}{6}\right)^{2-1} = 0{,}8333$$

$$P(X = 3) = \left(1 - \frac{1}{6}\right)^{3-1} = 0{,}6944$$

$$P(X \leq 3) = \frac{1}{6}(1 + 0{,}8333 + 0{,}6944) = 0{,}4213$$

$$P(X \geq 4) = 1 - 0{,}4213 = 0{,}5787$$

c) Calcule a média e o desvio padrão de X.

Média:

$$média = E(X) = \frac{1}{p}$$

$$E(X) = \frac{1}{\frac{1}{6}} = 6$$

Desvio padrão:

$$\sigma = \sqrt{\frac{1-p}{p^2}}$$

$$\sigma = \sqrt{\frac{1 - \frac{1}{6}}{\left(\frac{1}{6}\right)^2}} = \sqrt{30} = 5{,}4772$$

```
import random  # Para gerar números aleatórios
import math    # Para cálculos matemáticos (raiz quadrada)

# Configurações do problema do dado
p = 1/6        # Probabilidade de sucesso (sair o número 5)
num_simulacoes = 100000  # Número de repetições para as
simulações
```

```
# Gerando uma variável geométrica (sem usar funções)
intentos = 1  # Começa no primeiro lançamento
# Enquanto não obtivermos sucesso (número aleatório > p)
while random.random() > p:
    intentos += 1  # Conta mais um lançamento
valor_geometrico = intentos  # Guarda o resultado

# Inciso a) Probabilidade P(X=3) - Ganhar no terceiro
lançamento exatamente

contador_a = 0  # Contador para casos onde X=3

for _ in range(num_simulacoes):
    intentos = 1  # Reinicia contagem
    # Simula lançamentos até obter sucesso
    while random.random() > p:
        intentos += 1
    # Se conseguiu no 3° lançamento, conta
    if intentos == 3:
        contador_a += 1

# Calcula probabilidade como (casos favoráveis)/(total de
tentativas)
prob_a_sim = contador_a / num_simulacoes

print(f"a) Probabilidade de ganhar no 3° lançamento:
{prob_a_sim:.4f}")

# Inciso b) Probabilidade P(X≥4) - Necessitar pelo menos 4
lançamentos para ganhar
contador_b = 0  # Contador para casos onde X≥4

for _ in range(num_simulacoes):
    intentos = 1
    while random.random() > p:
        intentos += 1
    if intentos >= 4:  # Conta se precisou de 4+ lançamentos
        contador_b += 1

prob_b_sim = contador_b / num_simulacoes

print(f"b) Probabilidade de precisar de ≥4 lançamentos:
{prob_b_sim:.4f}")

# Inciso c) Cálculo da média e desvio padrão #
valores = []  # Armazenará todos os resultados das simulações

for _ in range(num_simulacoes):
```

```
    intentos = 1
    while random.random() > p:
        intentos += 1
    valores.append(intentos)  # Guarda cada resultado

# Cálculo da média (soma todos valores e divide pelo total)
media_sim = sum(valores)/num_simulacoes

# Cálculo da variância (média dos quadrados das diferenças)
varianza_sim = sum((x - media_sim)**2 for x in
valores)/num_simulacoes

# Desvio padrão é a raiz quadrada da variância
desviacion_sim = math.sqrt(varianza_sim)

print(f"c) Média estimada: {media_sim:.2f} lançamentos")
print(f"   Desvio padrão estimado: {desviacion_sim:.2f}
lançamentos")
```

```
a) Probabilidade de ganhar no 3º lançamento: 0.1160
b) Probabilidade de precisar de ≥4 lançamentos: 0.5761
c) Média estimada: 5.99 lançamentos
   Desvio padrão estimado: 5.45 lançamentos
```

**7.** Utilizando o método da inversa gerar amostrar para a distribuição:

$$f(x) = 3x^2, \qquad 0 \leq x \leq 1$$

Plotar a pdf analítica e o histograma normalizado.

Cálculos:

$CDF$:

$$F(x) = \int_0^x 3t^2 dt$$
$$F(x) = [t^3]_0^x = x^3$$
$$F(x) = x^3, \qquad 0 \leq x \leq 1$$

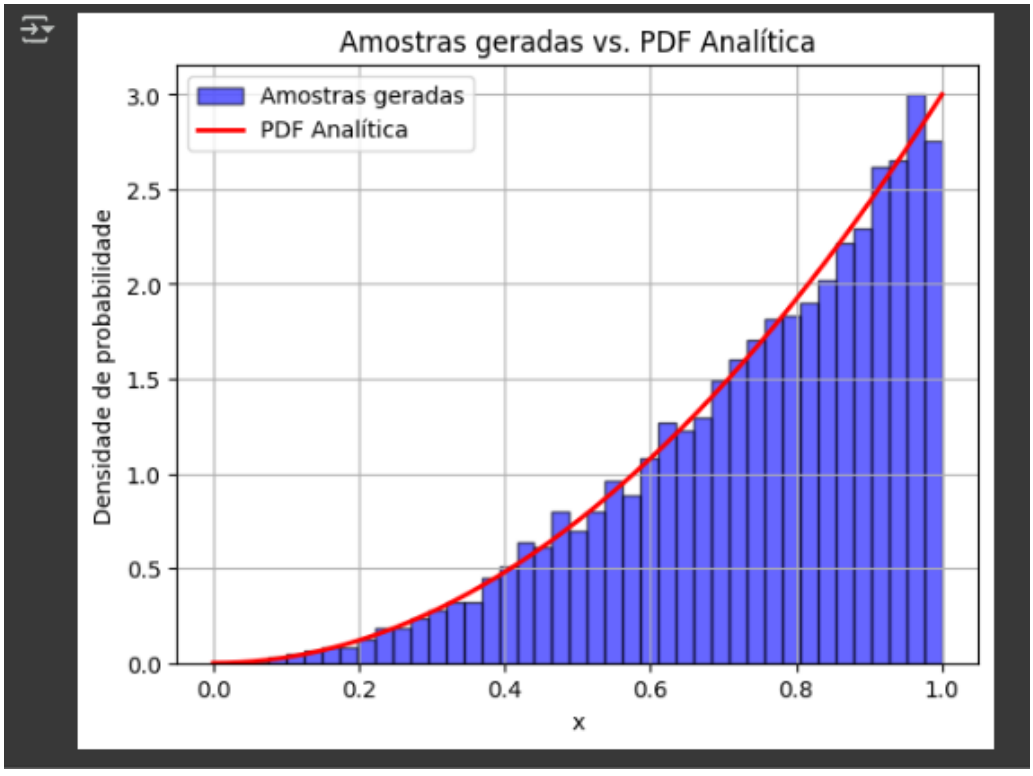
$Aplicação\ da\ inversa$:

$$U = F(x) = x^3$$
$$x = F^{-1}(U) = U^{\frac{1}{3}}$$

Por código:

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

```python
# Número de amostras
N = 10000

# Passo 1: Gerar valores uniformes U entre 0 e 1
U = np.random.uniform(0, 1, N)

# Passo 2: Aplicar a transformada inversa
X = U**(1/3)  # F^-1(U) = U^(1/3)

# Criar pontos para a PDF analítica
x_vals = np.linspace(0, 1, 1000)
pdf_vals = 3 * x_vals**2  # f(x) = 3x^2

# Plotar o histograma normalizado
plt.hist(X, bins=40, density=True, alpha=0.6, color='b',
edgecolor='black', label='Amostras geradas')

# Plotar a PDF analítica
plt.plot(x_vals, pdf_vals, 'r-', lw=2, label='PDF Analítica')

# Configurações do gráfico
plt.xlabel("x")
plt.ylabel("Densidade de probabilidade")
plt.title("Amostras geradas vs. PDF Analítica")
plt.legend()
plt.grid()
plt.show()
```



**8.** Considere uma variável aleatória contínua X cuja função densidade de probabilidade (pdf) é dada por:

$$f(x) = 3x^2, \qquad 0 \leq x \leq 1$$
$$e\ f(x) = 0, \qquad caso\ contrario$$

Suponha que você queira gerar valores dessa variável usando o **método da aceitação-rejeição.**

a) Verifique que $f(x)$ é uma densidade válida.

Uma função deve satisfazer:

1. Não negatividade: $f(x) \geq 0$

$$3x^2 \geq 0, \qquad 0 \leq x \leq 1$$

2. Normalização: A integral de $f(x)$ no seu domínio deve ser 1

$$\int_0^1 3x^2 dx = [x^3]_0^1 = 1 - 0 = 1$$

"$A\ função$ é $uma\ densidade\ válida$"

b) Encontre uma constante c adequada para a aplicação do método da aceitação-rejeição, considerando a distribuição candidata escolhida.

Se deve satisfazer:

$$f(x) \leq cg(x)$$

Uma escolha comum para $g(x)$ é a distribuição uniforme $g(x) = 1,\ [0,1]$:

$$3x^2 \leq c * 1, \qquad para\ todo\ x \in [0,1]$$

O valor máximo é $x = 1$:

$$c = 3(1)^2 = 3$$

c) Explique o procedimento passo a passo para gerar uma observação de X usando esse método.

1. Escolha da distribuição $g(x)$:

$g(x)$ é escolhido e a variável $c$ é calculada (isso foi feito anteriormente).

2. Geração de um possível $X$:

Geramos um número aleatório $U_1 \sim U(0,1)$ e definimos $X = U_1$, pois segue a distribuição uniforme $g(x)$

3. Geração de um critério de aceitação:

Geramos outro número aleatório $U_2 \sim U(0,1)$, que será usado para decidir se aceitamos ou não $X$.

4. Critério de aceitação

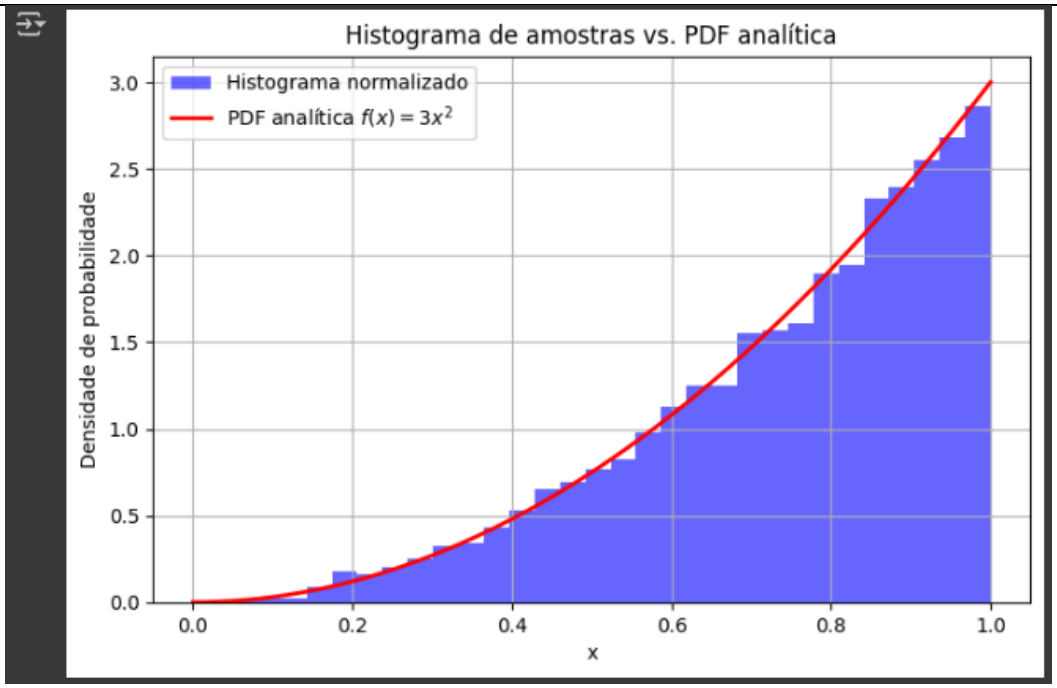
A razão:

$$\frac{f(X)}{cg(X)} = \frac{3X^2}{3} = X^2$$

Aceitamos $X$ como mostra válida se $U_2 \leq X^2$, caso contrário descartamos $X$ e voltamos ao passo 2.

Plotar a pdf analítica e o histograma normalizado

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

# Número de amostras
n_amostras = 10000
amostras = []

# Gerar amostras usando o método da aceitação-rejeição
while len(amostras) < n_amostras:
    U1 = np.random.uniform(0, 1)  # Gerar U1
    U2 = np.random.uniform(0, 1)  # Gerar U2

    if U2 <= U1**2:  # Critério de aceitação
        amostras.append(U1)

amostras = np.array(amostras)

# Gerar valores para a PDF teórica
x_vals = np.linspace(0, 1, 1000)
pdf_vals = 3 * x_vals**2

# Plotar histograma e PDF teórica
plt.figure(figsize=(8, 5))
plt.hist(amostras, bins=30, density=True, alpha=0.6,
color='b', label="Histograma normalizado")
plt.plot(x_vals, pdf_vals, 'r-', lw=2, label="PDF analítica
$f(x) = 3x^2$")
plt.xlabel("x")
plt.ylabel("Densidade de probabilidade")
plt.title("Histograma de amostras vs. PDF analítica")
plt.legend()
plt.grid()
plt.show()
```


Histograma de amostras vs. PDF analítica
Histograma normalizado
PDF analítica $f(x) = 3x^2$
Densidade de probabilidade
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0