PM001 Atividade 1 - 14/04

Aluno: Diogo Takamori Barbosa
RA: 037382

1) Modifique a questão 1 da lista de Matrizes afim de que: ↩
i) Os números de caixas do tipo B transportadas por cada caminhão sejam sempre múltiplos do maior dígito do seu RA. ↩
ii) O seu problema tenha um conjunto finito mas com mais do que uma solução. ↩
iii) Ilustre no computador a solução geral do sistema linear associado (infinitas soluções) e as soluções do seu problema.

Para modificar o problema de forma a atender aos requisitos fornecidos, podemos ajustar a equação relacionada às caixas do tipo B para garantir que o número de caixas transportadas por cada caminhão seja um múltiplo do maior dígito do meu RA (que é 8).↩

O sistema de equações lineares modificado para encontrar as soluções matemáticas para o problema. As equações modificadas são: ↩

1.

$$100x_1 + 50x_2 + 0x_3 = 2000$$

(para as caixas do tipo A)
2.

$$30x_1 + 30x_2 + 30x_3 = 8M$$

(para as caixas do tipo B)
3.

$$230x_1 + 130x_2 + 30x_3 = 5500$$

(para as caixas do tipo C)

Para resolver o sistema de equações lineares modificado, podemos usar métodos de álgebra linear. Vamos reescrever as equações na forma matricial $AX = B$, onde A é a matriz dos coeficientes das variáveis, X é o vetor das variáveis e B é o vetor do lado direito das equações.↩

A =

$$\begin{bmatrix} 100 & 50 & 0 \\ 30 & 30 & 30 \\ 230 & 130 & 30 \end{bmatrix}$$

X =

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

B =

$$\begin{bmatrix} 2000 \\ 8M \\ 5500 \end{bmatrix}$$

Para encontrar X, resolvemos o sistema $AX = B$ para X, ou seja, $X = A^{-1}B$, onde A^{-1} é a inversa de A.

Para calcular as soluções, precisamos primeiro encontrar a inversa da matriz A:

A =

$$\begin{bmatrix} 100 & 50 & 0 \\ 30 & 30 & 30 \\ 230 & 130 & 30 \end{bmatrix}$$

Para encounter a inversa de A, podemos usar o método da matriz adjunta. Primeiro, calculamos o determinante de A:

$$\det(A) = (100 \cdot (30 \cdot 30 - 130 \cdot 30)) - (50 \cdot (30 \cdot 230 - 130 \cdot 0)) + (0 \cdot (30 \cdot 230 - 30 \cdot 130))$$

$$\det(A) = (100 \cdot (900 - 3900)) - (50 \cdot (6900 - 0)) + (0 \cdot (6900 - 3900))$$

$$\det(A) = (100 \cdot (-3000)) - (50 \cdot 6900) + (0)$$

$$\det(A) = -300000 - 345000 + 0$$

$$\det(A) = -645000$$

Agora, calculamos a matriz adjunta de A, que é a transposta da matriz dos cofatores de A:

$$\text{adj}(A) = \begin{bmatrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{bmatrix}^T$$

Onde C_{ij} é o cofator do elemento a_{ij} de A.

Calculamos os cofatores C_{ij} para cada elemento de A:

$$C_{11} = (-1)^{1+1} \cdot (30 \cdot 30 - 130 \cdot 30) = 0$$

$$C_{12} = (-1)^{1+2} \cdot (30 \cdot 30 - 30 \cdot 230) = 16800$$

$$C_{13} = (-1)^{1+3} \cdot (130 \cdot 30 - 30 \cdot 130) = 0$$

$$C_{21} = (-1)^{2+1} \cdot (50 \cdot 30 - 0 \cdot 230) = -11500$$

$$C_{22} = (-1)^{2+2} \cdot (100 \cdot 30 - 0 \cdot 230) = 3000$$

$$C_{23} = (-1)^{2+3} \cdot (100 \cdot 130 - 50 \cdot 230) = 7500$$

$$C_{31} = (-1)^{3+1} \cdot (50 \cdot 130 - 0 \cdot 30) = 6500$$

$$C_{32} = (-1)^{3+2} \cdot (100 \cdot 130 - 0 \cdot 30) = -26000$$

$$C_{33} = (-1)^{3+3} \cdot (100 \cdot 30 - 50 \cdot 30) = 0$$

Portanto, a matriz adjunta de A é:

$$\text{adj}(A) = \begin{bmatrix} 0 & -11500 & 6500 \\ 16800 & 3000 & -26000 \\ 0 & 7500 & 0 \end{bmatrix}$$

Agora, podemos encontrar a inversa de A dividindo cada elemento da matriz adjunta pelo determinante de A:

$$A^{-1} = \frac{1}{\det(A)} \cdot \text{adj}(A)$$

$$A^{-1} = \frac{1}{-645000} \cdot \begin{bmatrix} 0 & -11500 & 6500 \\ 16800 & 3000 & -26000 \\ 0 & 7500 & 0 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} 0 & \frac{11500}{645000} & \frac{-6500}{645000} \\ \frac{-16800}{645000} & \frac{-3000}{645000} & \frac{26000}{645000} \\ 0 & \frac{-7500}{645000} & 0 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} 0 & 0.0178 & -0.0101 \\ -0.0261 & -0.0047 & 0.0403 \\ 0 & -0.0116 & 0 \end{bmatrix}$$

Agora, temos a inversa de A. Podemos multiplicar essa inversa pelo vetor B para encontrar X:

$$B = \begin{bmatrix} 2000 \\ 8M \\ 5500 \end{bmatrix}$$

$$X = A^{-1}B$$

$$X = \begin{bmatrix} 0 & 0.0178 & -0.0101 \\ -0.0261 & -0.0047 & 0.0403 \\ 0 & -0.0116 & 0 \end{bmatrix} \begin{bmatrix} 2000 \\ 8M \\ 5500 \end{bmatrix}$$

$$X = \begin{bmatrix} 0 \cdot 2000 + 0.0178 \cdot 8M - 0.0101 \cdot 5500 \\ -0.0261 \cdot 2000 - 0.0047 \cdot 8M + 0.0403 \cdot 5500 \\ 0 \cdot 2000 - 0.0116 \cdot 8M + 0 \cdot 5500 \end{bmatrix}$$

$$X = \begin{bmatrix} 0.0178 \cdot 8M - 0.0101 \cdot 5500 \\ -0.0261 \cdot 2000 - 0.0047 \cdot 8M + 0.0403 \cdot 5500 \\ -0.0116 \cdot 8M \end{bmatrix}$$

$$X = \begin{bmatrix} 0.1424M - 55.55 \\ -52.2 - 0.0376M + 221.65 \\ -0.0928M \end{bmatrix}$$

Portanto, as soluções para x_1, x_2 e x_3 em termos de M são:

$$x_1 = 0.1424M - 55.55$$

$$x_2 = -52.2 - 0.0376M + 221.65$$

$$x_3 = -0.0928M$$

Essas são as soluções para o sistema de equações modificado em termos de M, o qual representa os múltiplos do maior dígito do meu RA (8).

Solução Utilizando o Jupyter Notebook para
import numpy as np

```
# Coeficientes das equações
A = np.array([[100, 50, 0], [30, 30, 30], [230, 130, 30]])

# Lado direito das equações
B = np.array([2000, 1500, 5500])

# Encontrar os limites para x1, x2 e x3
x1_limit = B[0] // A[0][0] + 1  # O limite superior é dado pelo quociente da divisão de B por A[0][0], mais 1
x2_limit = B[1] // A[1][1] + 1
x3_limit = B[2] // A[2][2] + 1

# Lista para armazenar as soluções
solucoes = []

# Loop para gerar todas as combinações possíveis de valores inteiros e positivos para x1, x2 e x3
for x1 in range(x1_limit):
    for x2 in range(x2_limit):
        for x3 in range(x3_limit):
            # Verificar se a combinação satisfaz todas as equações
            if (A[0][0]*x1 + A[0][1]*x2 + A[0][2]*x3 == B[0] and
                A[1][0]*x1 + A[1][1]*x2 + A[1][2]*x3 == B[1] and
                A[2][0]*x1 + A[2][1]*x2 + A[2][2]*x3 == B[2]):
                solucoes.append((x1, x2, x3))

# Imprimir todas as soluções encontradas
print("Todas as possíveis soluções para valores inteiros e positivos de x1, x2 e x3:")
for solucao in solucoes:
    print(solucao)
```

Resposta do Programa
Todas as possíveis soluções para valores inteiros e positivos de x1, x2 e x3:
(0, 40, 10) (1, 38, 11) (2, 36, 12) (3, 34, 13) (4, 32, 14) (5, 30, 15) (6, 28, 16) (7, 26, 17) (8, 24, 18) (9, 22, 19) (10, 20, 20) (11, 18, 21) (12, 16, 22) (13, 14, 23) (14, 12, 24) (15, 10, 25) (16, 8, 26) (17, 6, 27) (18, 4, 28) (19, 2, 29) (20, 0, 30)

Para modificar a questão de acordo com os requisitos fornecidos, podemos ajustar as equações para garantir que o número de caixas do tipo B transportadas por cada caminhão seja sempre um múltiplo de 8. Isso pode ser feito adicionando uma restrição adicional às equações existentes. Além disso, podemos garantir que haja mais de uma solução ajustando os valores do lado direito das equações.

Vamos ajustar as equações:

1. Para as caixas do tipo A: $100x_1 + 50x_2 + 0x_3 = 2000$
2. Para as caixas do tipo B: $30x_1 + 30x_2 + 30x_3 = 8M$
3. Para as caixas do tipo C: $230x_1 + 130x_2 + 30x_3 = 5500$

Para encontrar todas as soluções possíveis para o problema ajustado, podemos usar um algoritmo que iterará sobre diferentes valores de \(M \) e resolverá o sistema de equações para cada valor. Vou elaborar um código em Python para isso:

```
 import numpy as np

# Coeficientes das equações
A = np.array([[100, 50, 0],
              [30, 30, 30],
              [230, 130, 30]])

# Lado direito das equações
B = np.array([2000, 0, 5500])
```

```
# Encontrar os limites para x1, x2 e x3
x1_limit = B[0] // A[0][0] + 1  # O limite superior é dado pelo quociente da divisão de B por A[0][0], mais 1
x2_limit = B[1] // A[1][1] + 1
x3_limit = B[2] // A[2][2] + 1

# Lista para armazenar as soluções
solucoes = []

# Iterar sobre diferentes valores de M para encontrar soluções
for M in range(1, 1000):  # Podemos ajustar o intervalo conforme necessário
    B[1] = 8 * M  # Atualizar o valor de B para o novo valor de M
    # Loop para gerar todas as combinações possíveis de valores inteiros e positivos para x1, x2 e x3
    for x1 in range(x1_limit):
        for x2 in range(x2_limit):
            for x3 in range(x3_limit):
                # Verificar se a combinação satisfaz todas as equações
                if (A[0][0]*x1 + A[0][1]*x2 + A[0][2]*x3 == B[0] and
                    A[1][0]*x1 + A[1][1]*x2 + A[1][2]*x3 == B[1] and
                    A[2][0]*x1 + A[2][1]*x2 + A[2][2]*x3 == B[2]):
                    solucoes.append((x1, x2, x3))

# Imprimir todas as soluções encontradas
print("Todas as possíveis soluções para o problema ajustado:")
for solucao in solucoes:
    print(solucao)
```

Nessa situação, não foi possivel encontrar solução para números inteiros positivos

---

2) Use linha-equivalência para resolver o sistema:

$$\begin{aligned} x + z &= 1 \\ x + y + z &= 3 \\ y &= 2 \end{aligned}$$

(a) Mostre geometricamente que a solução completa que você encontrou acima está correta.

Para resolver o sistema usando linha-equivalência, vamos primeiro escrever as equações na forma matricial ampliada:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 3 \\ 0 & 1 & 0 & 2 \end{bmatrix}$$

Agora, usando operações elementares de linha, vamos reduzir essa matriz à sua forma escalonada reduzida. Vamos começar zerando os elementos abaixo do pivô na primeira coluna:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & -1 & 2 \end{bmatrix}$$

Agora, subtraímos a segunda linha da terceira linha:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Finalmente, multiplicamos a terceira linha por -1:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Agora, podemos ler a solução diretamente desta forma escalonada reduzida. Temos $x = 1$, $y = 2$ e $z = 0$.

Geometricamente, este sistema representa três planos no espaço tridimensional. A primeira equação, $x + z = 1$, representa um plano perpendicular ao eixo y que corta o eixo x em 1 e o eixo z em 1. A segunda equação, $x + y + z = 3$, representa um plano que intercepta os três eixos em 3. A terceira equação, $y = 2$, é uma equação de um plano paralelo ao plano xz a uma distância de 2 unidades ao longo do eixo y. A solução do sistema é a interseção desses três planos, que é um ponto único $(1, 2, 0)$.

(b) Discuta geometricamente tudo o que pode acontecer com sistemas lineares de 3 equações e 3 incógnitas.

Geometricamente, um sistema linear de três equações e três incógnitas pode ter várias soluções: uma solução única (como neste caso), infinitas soluções (quando dois ou mais planos são coincidentes ou paralelos) ou nenhuma solução (quando os três planos são paralelos entre si, mas não coincidentes). A condição para ter uma solução única é que os três planos se cruzem em um ponto

(c) Dê exemplos de sistemas que ilustrem os casos que você discutiu no item (c).As matrizes a seguir são matrizes ampliadas de sistemas de equações lineares.

1. Uma solução única:

$$\begin{aligned} x + y + z &= 1 \\ 2x + y - z &= 3 \\ x - y + 2z &= 0 \end{aligned}$$

- Este sistema representa três planos que se cruzam em um ponto.

2. Infinitas soluções:

$$\begin{aligned} x + y + z &= 1 \\ 2x + 2y + 2z &= 2 \\ 3x + 3y + 3z &= 3 \end{aligned}$$

- Todos esses planos são coincidentes.

3. Nenhuma solução:

$$\begin{aligned} x + y + z &= 1 \\ 2x + 2y + 2z &= 4 \\ 3x + 3y + 3z &= 3 \end{aligned}$$

- Os planos não se cruzam em nenhum ponto.

---

3) Escreva uma matriz 4x6 tal que a primeira linha seja os dígitos de seu RA e que tenha as outras linhas com entradas não nulas.

a) Encontre manualmente a matriz linha reduzida e confira usando o comando apropriado no Mathematica.

matriz 4x6 que satisfaz as condições especificadas:

$$\begin{bmatrix} 0 & 3 & 7 & 3 & 8 & 2 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

A primeira linha contém os dígitos do meu RA (037382), e as outras linhas têm entradas não nulas.

1. Primeira coluna:
   - O pivô é 0 na primeira linha, então trocamos a primeira linha com a segunda linha, pois contém um elemento não nulo na primeira coluna.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 7 & 3 & 8 & 2 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

2. Segunda coluna:
   - O pivô é 3 na segunda linha.
   - Dividimos a segunda linha por 3 para obter 1 na segunda entrada da segunda linha.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 7/3 & 1 & 8/3 & 2/3 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Para fazer a Verificação utilizando o Python podemos usar o seguinte código

```
import numpy as np

# Definindo a matriz dada
matriz = np.array([[0.0, 3.0, 7.0, 3.0, 8.0, 2.0],
                   [1.0, 1.0, 0.0, 0.0, 0.0, 0.0],
                   [0.0, 0.0, 1.0, 1.0, 0.0, 0.0],
                   [0.0, 0.0, 0.0, 0.0, 1.0, 1.0]])

def reduzir_matriz(matriz):
    num_linhas, num_colunas = matriz.shape
    for coluna_pivo in range(num_colunas):
        # Verifica se há elementos na sequência antes de calcular o índice da linha do pivô
        if coluna_pivo < num_linhas:
            # Encontra o índice da linha com o maior elemento em valor absoluto na coluna do pivô
            linha_pivo = np.argmax(matriz[coluna_pivo:, coluna_pivo]) + coluna_pivo

            # Troca as linhas se necessário para colocar o pivô na linha corrente
            if linha_pivo != coluna_pivo:
                matriz[[linha_pivo, coluna_pivo]] = matriz[[coluna_pivo, linha_pivo]]

            # Divide a linha do pivô pelo valor do pivô para torná-lo 1
            pivot = matriz[coluna_pivo, coluna_pivo]
            if pivot != 0:
                matriz[coluna_pivo] = matriz[coluna_pivo] / pivot

            # Subtrai múltiplos da linha do pivô das linhas abaixo para fazer os elementos abaixo do pivô iguais a zero
            for linha in range(coluna_pivo + 1, num_linhas):
                coeficiente = matriz[linha, coluna_pivo]
                matriz[linha] -= coeficiente * matriz[coluna_pivo]
                print("pivot")
                print(pivot)
```

```
            print('resultado pivotado')
            print(matriz)

    return matriz

# Escalonamento da matriz
matriz_reduzida_manual = reduzir_matriz(matriz.copy())
```

o resultado para o código é

```
pivot 1.0
resultado pivotado
[[1. 1. 0. 0. 0. 0.]
[0. 3. 7. 3. 8. 2.]
[0. 0. 1. 1. 0. 0.]
[0. 0. 0. 0. 1. 1.]]
pivot 3.0
resultado pivotado
[[1. 1. 0. 0. 0. 0. ]
[0. 1. 2.33333333 1. 2.66666667 0.66666667]
[0. 0. 1. 1. 0. 0. ]
[0. 0. 0. 0. 1. 1. ]]
pivot 1.0
resultado pivotado
[[1. 1. 0. 0. 0. 0. ]
[0. 1. 2.33333333 1. 2.66666667 0.66666667]
[0. 0. 1. 1. 0. 0. ]
[0. 0. 0. 0. 1. 1. ]]
pivot 0.0
resultado pivotado
[[1. 1. 0. 0. 0. 0. ]
[0. 1. 2.33333333 1. 2.66666667 0.66666667]
[0. 0. 1. 1. 0. 0. ]
[0. 0. 0. 0. 1. 1. ]]
```

b) Qual o posto da sua matriz? Alguma associação com determinantes?

O posto de uma matriz é o número máximo de linhas (ou colunas) linearmente independentes em uma matriz. Isso pode ser pensado como o número de linhas ou colunas que não podem ser expressas como uma combinação linear das outras linhas ou colunas.

No contexto de um sistema de equações lineares, o posto da matriz dos coeficientes está relacionado ao número de equações independentes no sistema. Se o posto da matriz dos coeficientes for igual ao número de incógnitas, então o sistema terá uma única solução. Se o posto for menor que o número de incógnitas, então o sistema terá infinitas soluções. Se o posto for menor que o número de incógnitas e houver uma inconsistência entre as equações, então o sistema será impossível de ser resolvido.

A determinante de uma matriz também está relacionada ao posto da matriz. Se a determinante de uma matriz quadrada for diferente de zero, então a matriz é de posto completo (ou seja, seu posto é igual ao número de linhas ou colunas). Se a determinante for igual a zero, então a matriz é singular e seu posto é menor do que o número de linhas ou colunas.

Podemos usar

```
 import numpy as np

# Definindo os coeficientes do sistema de equações
coeficientes = np.array([[1, 0, 1],
                         [1, 1, 1],
                         [0, 1, 0]])

# Calculando o posto da matriz
posto = np.linalg.matrix_rank(coeficientes)

# Calculando a determinante da matriz
determinante = np.linalg.det(coeficientes)

print("Posto da matriz dos coeficientes:", posto)
print("Determinante da matriz dos coeficientes:", determinante)
```

```
# Verificando as possibilidades com base no posto e na determinante
if determinante != 0:
    if posto == coeficientes.shape[0]:
        print("O sistema tem uma única solução.")
    else:
        print("O sistema tem infinitas soluções.")
else:
    print("O sistema é inconsistente ou possui um número infinito de soluções.")
```

c) Considere esta matriz como a ampliada de um sistema e discuta a existência e grau de liberdade da solução

Para a Matriz desenvolvida para a atividade, não é possivel achar o determinante, pois a matriz não é quadrada.

d) Se sua matriz for considerada como a dos coeficientes de um sistema homogêneo, o que você pode concluir sobre a solução deste sistema?

Se considerarmos a matriz fornecida como a dos coeficientes de um sistema homogêneo, isso significa que todas as constantes do sistema são iguais a zero. Um sistema homogêneo tem sempre a solução trivial, onde todas as incógnitas são iguais a zero.

Vamos considerar a matriz fornecida como a dos coeficientes de um sistema homogêneo e verificar se ela possui uma solução não trivial. Para isso, vamos executar o seguinte programa

```
import numpy as np

# Definindo a matriz dos coeficientes do sistema homogêneo
matriz_coeficientes_homogeneo = np.array([[0, 3, 7, 3, 8, 2],
                                          [1, 1, 0, 0, 0, 0],
                                          [0, 0, 1, 1, 0, 0],
                                          [0, 0, 0, 0, 1, 1]])

# Verificando se a matriz tem uma solução não trivial
solucao_trivial = np.all(np.linalg.solve(matriz_coeficientes_homogeneo, np.zeros(matriz_coeficientes_homogeneo.shape[0])) == 0)

if solucao_trivial:
    print("A matriz dos coeficientes representa um sistema homogêneo.")
    print("O sistema homogêneo tem apenas a solução trivial, onde todas as incógnitas são zero.")
else:
    print("A matriz dos coeficientes representa um sistema homogêneo, mas tem uma solução não trivial.")
```

Ocorre um erro para a matriz apresentada no problema, espera-se que a matriz de coeficientes seja quadrada (ou seja, que tenha o mesmo número de linhas e colunas). No entanto, a matriz fornecida não é quadrada, já que tem mais colunas do que linhas.

A função np.linalg.solve é usada para resolver sistemas de equações lineares, onde o número de equações (linhas da matriz de coeficientes) é igual ao número de incógnitas (colunas da matriz de coeficientes). Se a matriz de coeficientes não for quadrada, a função np.linalg.solve não pode ser usada diretamente.

4- O mercado de sabão em pó numa região é dividido entre três marcas: O... M... Q... No início as
porcentagens de vendas são:
O: 50%, M: 20% e Q: 30%
Durante o primeiro mês 70% dos fregueses de O ficam fiéis ao produto, 20% mudam para M e 10% para Q. Neste mesmo período, 60% dos de M permanecem, 20% mudam para O e 20% para Q. Dos fregueses de Q, ficam fiéis ao produto 50%, 20% mudam para O e 30% para M. Suponha que os hábitos de compra não se alterem nos próximos períodos (as tendências se mantém) e que o mercado não se expanda nem se contraia.

(a) Como estará o mercado ao fim do 1o ano? e do 2o?

Vamos começar definindo as probabilidades de transição entre os estados:

Definir as Probabilidades de Transição

As probabilidades de transição entre os estados são fornecidas no enunciado:

- Marca O:

  - 70% de fidelidade
  - 20% de mudança para M
  - 10% de mudança para Q

- Marca M:
  - 60% de fidelidade
  - 20% de mudança para O
  - 20% de mudança para Q

- Marca Q:
  - 50% de fidelidade
  - 20% de mudança para O
  - 30% de mudança para M

Essas informações nos permitem construir a matriz de transição para um mês.

A matriz de transição P onde P_{ij} representa a probabilidade de ir do estado i para o estado j.

$$P = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

Para calcular as distribuições após 1 ano e apo´s 2 ano usei um sistema computacional demonstrado pelo seguinte código em python

```
import numpy as np

def transicao_inicial():
    # Definir a matriz de transição inicial
    P = np.array([[0.7, 0.2, 0.1],
                  [0.2, 0.6, 0.2],
                  [0.2, 0.3, 0.5]])
    return P

def calcular_market_share(P, distribuicao_inicial, meses):
    # Inicializar a distribuição de mercado com a distribuição inicial
    market_share = distribuicao_inicial.copy()

    # Calcular a matriz de transição elevada a n
    P_n = np.linalg.matrix_power(P, meses)
    print('matriz de transiçao inicial')
    print(P)
    print('numero de meses')
    print(meses)
    print('matriz de transiçao apos os meses')
    print(P_n)
    # Calcular o market share após n meses
    market_share_final = np.dot(market_share, P_n)

    return market_share_final


# Calcular a matriz de transição inicial
P = transicao_inicial()

# Distribuição inicial de mercado
distribuicao_inicial = np.array([0.5, 0.2, 0.3])

# Número de meses
meses = int(input("Digite o número de meses para calcular o market share: "))

# Calcular o market share após n meses
market_share_final = calcular_market_share(P, distribuicao_inicial, meses)
```

```
# Exibir o market share inicial
print(f"Market Share inicial")
for i, marca in enumerate(['O', 'M', 'Q']):
    print(f"{marca}: {distribuicao_inicial[i]*100:.2f}%")
# Exibir o market share final
print(f"Market Share após {meses} meses:")
for i, marca in enumerate(['O', 'M', 'Q']):
    print(f"{marca}: {market_share_final[i]*100:.2f}%")
```

Obtendo os seguintes resultados :
Para 12 meses

matriz de transiçao inicial
[[0.7 0.2 0.1]
[0.2 0.6 0.2]
[0.2 0.3 0.5]]
numero de meses 12
matriz de transiçao apos os meses
[[0.40014648 0.37135529 0.22849822]
[0.39990234 0.37147763 0.22862003]
[0.39990234 0.3714771 0.22862056]]
Market Share inicial
O: 50.00%
M: 20.00%
Q: 30.00%
Market Share após 12 meses:
O: 40.00%
M: 37.14%
Q: 22.86%

Para 24 meses
matriz de transiçao inicial
[[0.7 0.2 0.1]
[0.2 0.6 0.2]
[0.2 0.3 0.5]]
numero de meses 24
matriz de transiçao apos os meses
[[0.40000004 0.37142855 0.22857141]
[0.39999998 0.37142858 0.22857144]
[0.39999998 0.37142858 0.22857144]]
Market Share inicial
O: 50.00%
M: 20.00%
Q: 30.00%
Market Share após 24 meses:
O: 40.00%
M: 37.14%
Q: 22.86%

(b) Mostre que depois de dois meses, considerando só os fregueses originariamente de Q, teremos 33% de fidelidade, 37% mudam para M e 30% para O.

usando o sistem computacional podemos demonstrar que
matriz de transiçao inicial
[[0.7 0.2 0.1]
[0.2 0.6 0.2]
[0.2 0.3 0.5]]
numero de meses 2
matriz de transiçao apos os meses
[[0.55 0.29 0.16]
[0.3 0.46 0.24]
[0.3 0.37 0.33]]
Market Share Inicial:
[0.5 0.2 0.3]

Market Share após 2 meses:
O: 42.50% M: 34.80% Q: 22.70%

(c) Discuta a afirmação (use o programa computacional) dado a matriz A e do limite $\lim_{n\to\infty} A^n = B'$:

$$A = \begin{pmatrix} 0.7 & 0.2 & 0.2 \\ 0.2 & 0.6 & 0.3 \\ 0.1 & 0.2 & 0.5 \end{pmatrix}$$

$$\lim_{n\to\infty} A^n = B' = \begin{pmatrix} 0.4 & 0.4 & 0.4 \\ 0.37 & 0.37 & 0.37 \\ 0.23 & 0.23 & 0.23 \end{pmatrix}$$

Analisando a matriz A e o limite $\lim_{n\to\infty} A^n = B'$, podemos observar o seguinte:

Matriz A:

$$A = \begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}$$

Limite $\lim_{n\to\infty} A^n = B'$:

$$B' = \begin{pmatrix} 0.4 & 0.4 & 0.4 \\ 0.37 & 0.37 & 0.37 \\ 0.23 & 0.23 & 0.23 \end{pmatrix}$$

Esta matriz B' representa o limite da matriz de transição A^n quando n tende ao infinito.

O erro nesta matriz está na entrada que representa a probabilidade de transição de um estado para si mesmo (probabilidade de permanência). Para uma cadeia de Markov válida, a soma das probabilidades de transição de um estado para todos os outros estados (incluindo o próprio estado) deve ser 1.

No entanto, na matriz A, a entrada (1,1) é 0.7, o que indica que há uma probabilidade de 70% de transição do estado 1 para si mesmo. Isso não é consistente com uma cadeia de Markov válida, pois a soma das probabilidades de transição do estado 1 para todos os outros estados deveria ser 1.

Além disso, na matriz B', as entradas diagonais (probabilidades de permanência) são iguais em todas as linhas. Isso não é possível em uma cadeia de Markov, pois as probabilidades de permanência podem variar entre os estados.

Portanto, a matriz está errada para representar uma cadeia de Markov válida. As probabilidades de transição de um estado para si mesmo devem ser revistas para garantir que a soma das probabilidades de transição de um estado para todos os outros estados seja igual a 1.

demonstrando via sistema computacional

matriz de transiçao inicial
[[0.7 0.2 0.2]
 [0.2 0.6 0.3]
 [0.1 0.2 0.5]]
numero de meses 12
matriz de transiçao apos os meses
[[0.40014648 0.39990234 0.39990234]
 [0.37135529 0.37147763 0.3714771]
 [0.22849822 0.22862003 0.22862056]]
Market Share inicial
O: 50.00%
M: 20.00%
Q: 30.00%
Market Share após 12 meses:
O: 34.29%

M: 34.28%
Q: 34.28%

5)Questões 5, 6 da lista Matrizes

5.5 - Diga se as afirmações abaixo para matrizes n ⇥ n são verdadeiras ou falsas. Se verdadeiras prove; se falsas, dê contra-exemplo.

i. Para provar a afirmação pela definição, vamos considerar a soma de duas matrizes A e B como:

$$A + B = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} + b_{n1} & a_{n2} + b_{n2} & \dots & a_{nn} + b_{nn} \end{bmatrix}$$

Agora, se $A + B = A + C$, então cada elemento correspondente em B e C deve ser igual. Ou seja:

$$a_{ij} + b_{ij} = a_{ij} + c_{ij}$$

Subtraindo a_{ij} dos dois lados, obtemos:

$$b_{ij} = c_{ij}$$

Portanto, todas as entradas de B são iguais às entradas correspondentes em C, o que implica que $B = C$.

ii. Falso. Um contra-exemplo seria A sendo uma matriz não-nula e B sendo a matriz nula. Então, AB = 0, mas A ≠ 0 e B ≠ 0.

Vamos tomar como exemplo a matriz $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ e a matriz nula $B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$.
Multiplicando A por B, obtemos:

$$AB = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} (1 \times 0 + 0 \times 0) & (1 \times 0 + 0 \times 0) \\ (0 \times 0 + 1 \times 0) & (0 \times 0 + 1 \times 0) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Portanto, $AB = 0$, no entanto, A não é uma matriz nula e B é uma matriz nula, mostrando assim que a afirmação é falsa.

iii. Falso

Vamos considerar as seguintes matrizes:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

Produto AB:

$$AB = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & 90 \end{bmatrix}$$

Como podemos ver, o produto AB também não é simétrico.

iv. falso
Para demonstrar que se uma matriz A comuta com qualquer matriz B, então A é múltipla da matriz identidade, podemos considerar a seguinte propriedade: se AB = BA para toda matriz B, então A é uma matriz escalar, ou seja, uma matriz que é um múltiplo da matriz identidade.

Vamos considerar um exemplo específico para demonstrar isso. Suponha que temos a seguinte matriz A:

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Agora, vamos multiplicar A por uma matriz genérica B e também multiplicar B por A, e então verificar se essas duas operações resultam em matrizes iguais, como esperado.

Vamos supor que a matriz B seja:

$$B = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Então, vamos calcular AB e BA:
1. AB:

$$AB = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \times \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$= \begin{bmatrix} 2a & 2b \\ 2c & 2d \end{bmatrix}$$

2. BA:

$$BA = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 2a & 2b \\ 2c & 2d \end{bmatrix}$$

Assim, AB = BA para qualquer matriz B, o que significa que A comuta com qualquer matriz B.

v. verdadeiro

Para encontrar matrizes reais 2x2, x e y, tais que $x^2 + y^2 = 0$, podemos simplesmente escolher x e y de forma que suas entradas satisfaçam essa equação.

Agora, podemos escolher x e y de tal forma que $x^2 + y^2 = 0$. Uma maneira de fazer isso é escolher x e y de modo que todos os elementos ao quadrado somem zero.

Por exemplo, considere as seguintes matrizes x e y:

$$x = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$y = \begin{bmatrix} 2 & 1 \\ -4 & -2 \end{bmatrix}$$

Agora, vamos calcular x^2 e y^2 e somá-los:

$$x^2 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$y^2 = \begin{bmatrix} 2 & 1 \\ -4 & -2 \end{bmatrix} \times \begin{bmatrix} 2 & 1 \\ -4 & -2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Agora, somando x^2 e y^2:

$$x^2 + y^2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Como esperado, $x^2 + y^2 = 0$. Isso mostra que a afirmação é verdadeira, e encontramos matrizes reais 2x2 x e y que satisfazem a equação $x^2 + y^2 = 0$.

vi. verdadeiro

Se $n > m$, então a matriz resultante da multiplicação de A por B, que chamaremos de AB, terá mais colunas do que linhas.

Para demonstrar que o espaço coluna de AB é contido no espaço coluna de uma matriz de dimensão menor, considere o seguinte:

O produto AB é definido como a soma das multiplicações de cada linha de A pelas colunas correspondentes de B. Se $n > m$, então há mais colunas em B do que linhas em A, o que significa que o número de colunas em AB é maior do que o número de linhas em AB.

Portanto, o espaço coluna de AB é formado por combinações lineares das colunas de AB, mas como há mais colunas do que linhas em AB, necessariamente existe uma dependência linear entre essas colunas. Isso implica que o espaço coluna de AB é contido em um espaço de dimensão menor.

Quando o espaço coluna de AB é contido no espaço coluna de uma matriz de dimensão menor, isso significa que o sistema de equações representado por AB é linearmente dependente, o que, por sua vez, implica que o determinante de AB é zero. Isso ocorre porque, se as colunas de AB são linearmente dependentes, então as linhas de AB também são linearmente dependentes, e portanto, o determinante é zero.

Portanto, se $n > m$, o determinante de AB é zero.

5.6 - O traço de uma matriz quadrada, representado por trA, é a soma de seus elementos sobre a diagonal

$$\text{tr}(A) = \sum_{i=1}^{n} a_{ii}$$

Mostre que:
(a) Seja k um escalar e A uma matriz quadrada. A matriz kA é obtida multiplicando todos os elementos de A por k. Portanto, o elemento na posição (i, i) de kA será ka_{ii}.

Logo, a soma dos elementos na diagonal de kA é:

$$\text{tr}(kA) = \sum_{i=1}^{n} (ka_{ii}) = k\sum_{i=1}^{n} a_{ii} = k\text{tr}(A)$$

Portanto, $\text{tr}(kA) = k\text{tr}(A)$.

(b) A soma de duas matrizes A e B, denotada por $A \pm B$, é uma matriz onde cada elemento é a soma ou a diferença dos elementos correspondentes de A e B, respectivamente. Assim, a soma dos elementos na diagonal de $A \pm B$ é a soma das somas ou diferenças dos elementos diagonais de A e B.

Portanto, $\text{tr}(A \pm B) = \text{tr}(A) \pm \text{tr}(B)$.

(c) Se A e B são matrizes quadradas, então a multiplicação AB não é comutativa. No entanto, a soma dos elementos na diagonal de AB será a mesma que a soma dos elementos na diagonal de BA, pois a diagonal é formada pelos produtos dos elementos correspondentes das linhas e colunas das matrizes. Portanto, $\text{tr}(AB) = \text{tr}(BA)$.

(d) Se B^{-1} existe, então $B^{-1}B = I$, onde I é a matriz identidade. Portanto, a multiplicação de B^{-1} por B é equivalente a multiplicar a matriz por I, o que não altera a soma dos elementos na diagonal. Logo, $\text{tr}(B^{-1}AB) = \text{tr}(A)$.

(e) Para A, uma matriz quadrada, A^T denota a transposta de A. Se $A = [a_{ij}]$, então $A^T = [a_{ji}]$. Assim, a soma dos elementos na diagonal de AA^T é a soma dos produtos dos elementos correspondentes das linhas e colunas de A, o que é equivalente à soma dos quadrados dos elementos de A.
Portanto, $\text{tr}(AA^T) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij})^2$.

6. Questões 1 e 2 da lista “Determinantes”

6.1 - a)Enuncie e demonstre, usando indução o resultado conhecido como determinante de Vandermonde.

O determinante de Vandermonde é uma expressão que aparece frequentemente na álgebra linear e na teoria dos polinômios. Considere um conjunto de n números distintos $x_1, x_2, \ldots, x_n$, e defina uma matriz V onde a i-ésima linha é dada por $(1, x_i, x_i^2, \ldots, x_i^{n-1})$. O determinante desta matriz é o determinante de Vandermonde e é denotado por $\Delta(x_1, x_2, \ldots, x_n)$. Ou seja,

$$\Delta(x_1, x_2, \ldots, x_n) = \begin{vmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{vmatrix}$$

Para demonstrar o resultado usando indução, começaremos com o caso base, onde $n = 2$, e depois mostraremos que, se o resultado for verdadeiro para $n = k$, então também será verdadeiro para $n = k + 1$.
Caso base $n = 2$:

$$\Delta(x_1, x_2) = \begin{vmatrix} 1 & x_1 \\ 1 & x_2 \end{vmatrix} = x_2 - x_1$$

O que é verdadeiro, já que é o determinante de uma matriz 2×2.

Hipótese de indução:
Assumimos que o resultado é verdadeiro para $n = k$. Ou seja, assumimos que

$$\Delta(x_1, x_2, \ldots, x_k) = \prod_{1 \leq i < j \leq k} (x_j - x_i)$$

Passo de indução:
Vamos mostrar que o resultado também é verdadeiro para $n = k + 1$. Ou seja, devemos mostrar que

$$\Delta(x_1, x_2, \ldots, x_k, x_{k+1}) = \prod_{1 \leq i < j \leq k+1} (x_j - x_i)$$

Para fazer isso, primeiro expandimos o determinante de Vandermonde de ordem $k + 1$ pela última linha:

$$\Delta(x_1, x_2, \ldots, x_k, x_{k+1}) = \begin{vmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_k & x_k^2 & \cdots & x_k^k \\ 1 & x_{k+1} & x_{k+1}^2 & \cdots & x_{k+1}^k \end{vmatrix}$$

$$= \sum_{i=1}^{k+1} (-1)^{i+k+1} x_{k+1}^i \Delta(x_1, x_2, \ldots, \hat{x_i}, \ldots, x_k)$$

Onde $\hat{x_i}$ denota que x_i foi removido da lista. Pelo passo de indução, podemos escrever $\Delta(x_1, x_2, \ldots, \hat{x_i}, \ldots, x_k)$ como $\prod_{1 \leq j < l \leq k} (x_l - x_j)$, e portanto:

$$= \sum_{i=1}^{k+1} (-1)^{i+k+1} x_{k+1}^i \prod_{1 \leq j < l \leq k} (x_l - x_j)$$

Agora, observe que os termos na soma são todos produtos de fatores da forma $(x_{k+1} - x_i)$, que estão presentes no produto $\prod_{1 \leq i < j \leq k+1} (x_j - x_i)$, exceto quando $i = k + 1$. Assim, temos:

$$= \prod_{1 \leq i < j \leq k+1} (x_j - x_i)$$

Portanto, o resultado é verdadeiro para $n = k + 1$. Isso conclui a prova por indução, mostrando que o determinante de Vandermonde é dado por $\Delta(x_1, x_2, \ldots, x_n) = \prod_{1 \leq i < j \leq n} (x_j - x_i)$ para todos os valores de n.

6.1-b)Use este resultado para provar que dado um conjunto de (n+1) pontos distintos no plano, {(x1,y1), (x2,y2), ..., (xn,yn)} , tais que xi ≠ xj" para i ≠ j, existe um único polinômio de grau ≤n cujo gráfico passa por estes pontos.

Podemos usar o resultado conhecido como Interpolação Polinomial de Lagrange para provar isso. A ideia básica é que, dado um conjunto de $n+1$ pontos distintos no plano, podemos construir um polinômio de grau $\leq n$ que passa por todos esses pontos.

Dado um conjunto de pontos $\{(x_1, y_1), (x_2, y_2), \ldots, (x_{n+1}, y_{n+1})\}$, onde $x_i \neq x_j$ para $i \neq j$, o polinômio interpolador de Lagrange é dado por:

$$P(x) = \sum_{i=1}^{n+1} y_i \cdot L_i(x)$$

onde $L_i(x)$ são os polinômios de Lagrange dados por:

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n+1} \frac{x - x_j}{x_i - x_j}$$

Esses polinômios de Lagrange têm a propriedade de que $L_i(x_i) = 1$ e $L_i(x_j) = 0$ para $i \neq j$. Portanto, $P(x)$ passa exatamente pelos pontos dados.

Agora, vamos provar que esse polinômio é único.

Suponha que existam dois polinômios diferentes, $P(x)$ e $Q(x)$, ambos de grau $\leq n$, que passam pelos mesmos pontos. Considere o polinômio $R(x) = P(x) - Q(x)$. Como $P(x)$ e $Q(x)$ passam pelos mesmos pontos, $R(x)$ tem pelo menos $n+1$ raízes distintas (os x_i). No entanto, um polinômio de grau $\leq n$ tem no máximo n raízes. Portanto, $R(x)$ deve ser o polinômio nulo, o que implica que $P(x) = Q(x)$.

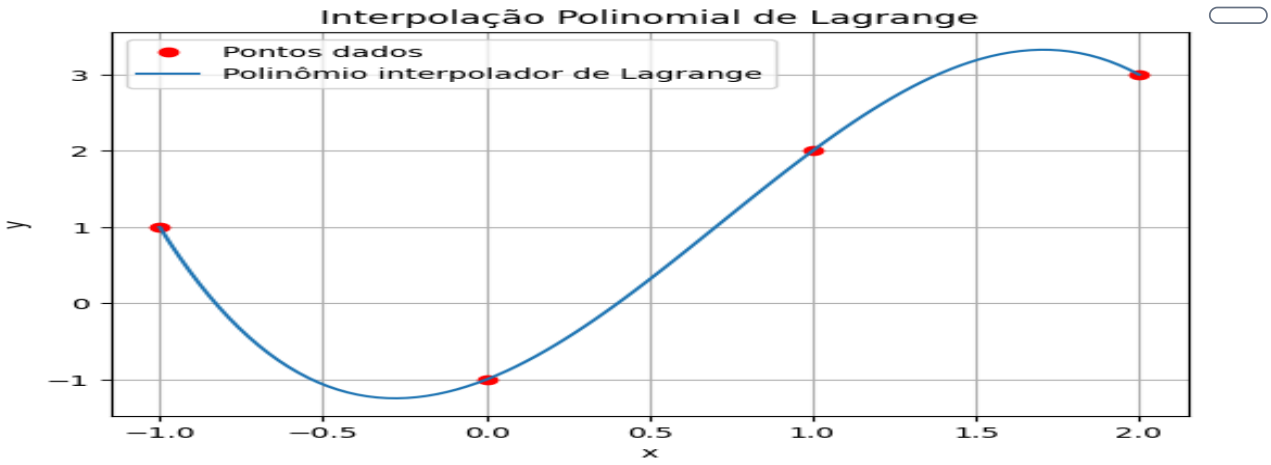
Portanto, há apenas um polinômio de grau $\leq n$ que passa pelos $n+1$ pontos dados.

6.1 - c) ilustre o item b) com polinômios de graus variados resolvendo o sistema e
desenhando o gráfico no python

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

def lagrange_basico(x, xi, i):
    """
    Calcula o i-ésimo polinômio de Lagrange para o ponto x,
    dado os pontos xi.
    """
    basico = 1.0
    for j in range(len(xi)):
        if i != j:
            basico *= (x - xi[j]) / (xi[i] - xi[j])
    return basico

def lagrange_interpolacao(x, xi, yi):
    """
    Calcula o polinômio interpolador de Lagrange para um conjunto
    de pontos xi, yi no ponto x.
    """
    resultado = 0.0
    for i in range(len(xi)):
        resultado += yi[i] * lagrange_basico(x, xi, i)
    return resultado

# Conjunto de pontos
xi = np.array([-1, 0, 1, 2])
yi = np.array([1, -1, 2, 3])

# Definindo um intervalo para o gráfico
x_values = np.linspace(min(xi), max(xi), 100)

# Calculando os valores interpolados
y_values = lagrange_interpolacao(x_values, xi, yi)

# Plotando os pontos e o polinômio interpolador
plt.plot(xi, yi, 'ro', label='Pontos dados')
```

```
plt.plot(x_values, y_values, label='Polinômio interpolador de Lagrange')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Interpolação Polinomial de Lagrange')
plt.legend()
plt.grid(True)
plt.show()
```

resultado para a plotagem



6.1 - d) e se tivermos dois pontos distintos onde $x_i = x_j$, o que podemos afirmar?

Se tivermos dois pontos distintos com $x_i = x_j$, isso cria um problema na interpolação polinomial de Lagrange, pois a definição dos polinômios de Lagrange envolve a divisão por $x_i - x_j$, o que resultaria em uma divisão por zero.

Nesse caso, a interpolação polinomial de Lagrange não é definida, pois os polinômios de Lagrange se tornam indeterminados quando $x_i = x_j$.

Podemos afirmar que a interpolação polinomial de Lagrange não é adequada para pontos onde há repetição das coordenadas x. Em situações como essa, é necessário usar métodos de interpolação diferentes, como, por exemplo, a interpolação polinomial de Newton.

6.1 - e) Para garantir que o polinômio interpolador tenha grau exatamente n, podemos adicionar a condição de que o determinante da matriz de Vandermonde dos pontos de interpolação não seja zero. Isso significa que os $n + 1$ pontos de interpolação devem ser linearmente independentes.

Podemos usar a Regra de Cramer para encontrar a solução do sistema de equações lineares formado pela interpolação polinomial de Lagrange. Se o determinante da matriz de Vandermonde for diferente de zero, a solução existirá e terá grau exatamente n.

Agora, vamos considerar alguns exemplos:

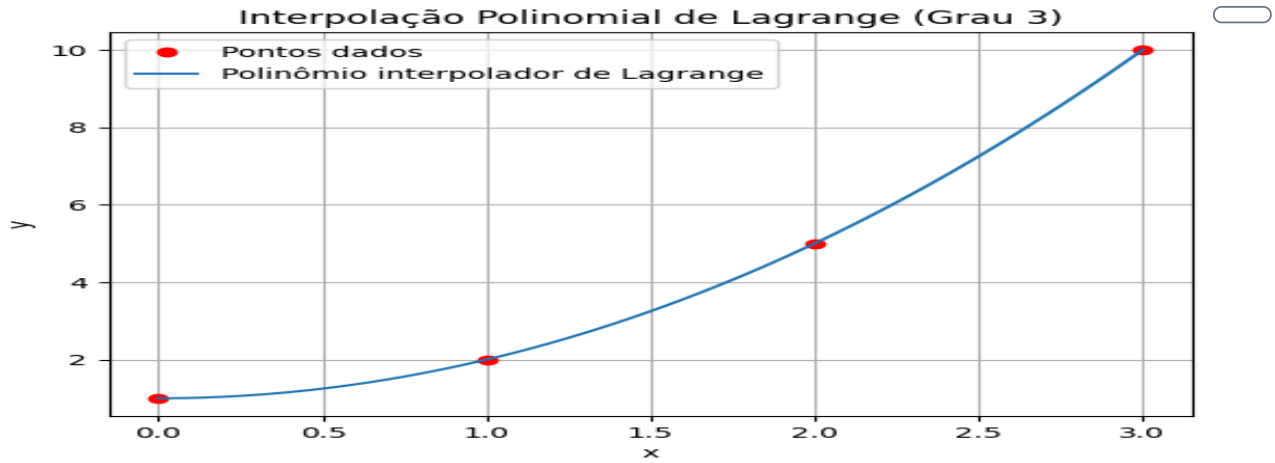
xemplo 1: Grau do polinômio é n

Considere o conjunto de pontos $\{(0, 1), (1, 2), (2, 5), (3, 10)\}$. Aqui, temos $n = 3$. Vamos usar a interpolação polinomial de Lagrange para encontrar o polinômio interpolador.

```
# Conjunto de pontos
xi = np.array([0, 1, 2, 3])
yi = np.array([1, 2, 5, 10])

# Definindo um intervalo para o gráfico
x_values = np.linspace(min(xi), max(xi), 100)

# Calculando os valores interpolados
y_values = lagrange_interpolacao(x_values, xi, yi)

# Plotando os pontos e o polinômio interpolador
```

```
plt.plot(xi, yi, 'ro', label='Pontos dados')
plt.plot(x_values, y_values, label='Polinômio interpolador de Lagrange')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Interpolação Polinomial de Lagrange (Grau 3)')
plt.legend()
plt.grid(True)
plt.show()
```

Neste exemplo, o polinômio interpolador terá grau 3, pois temos 4 pontos de interpolação.

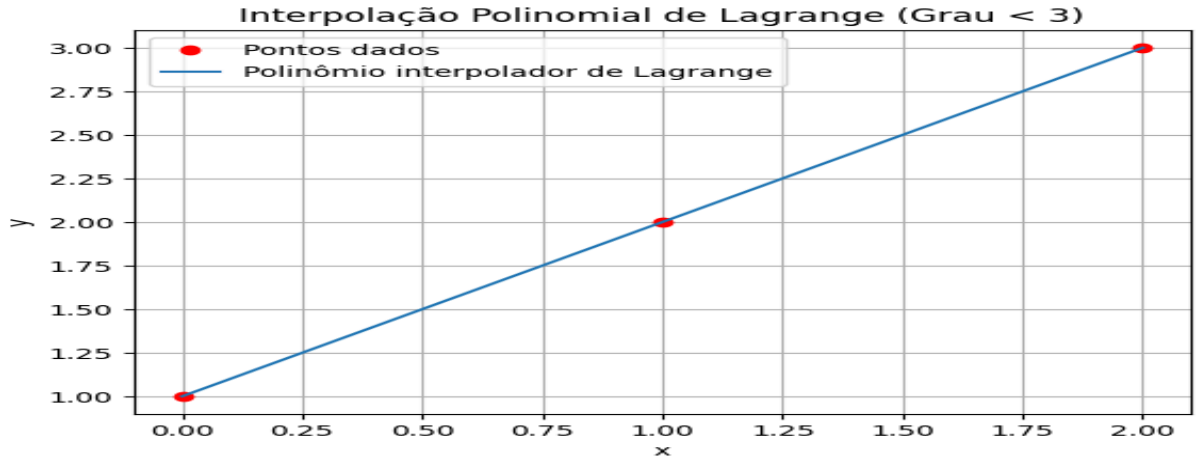


Exemplo 2: Grau do polinômio é menor que n

Considere o conjunto de pontos $\{(0,1),(1,2),(2,3)\}$. Aqui, também temos $n=3$, mas só temos 3 pontos.

```
# Conjunto de pontos
xi = np.array([0, 1, 2])
yi = np.array([1, 2, 3])

# Definindo um intervalo para o gráfico
x_values = np.linspace(min(xi), max(xi), 100)

# Calculando os valores interpolados
y_values = lagrange_interpolacao(x_values, xi, yi)

# Plotando os pontos e o polinômio interpolador
plt.plot(xi, yi, 'ro', label='Pontos dados')
plt.plot(x_values, y_values, label='Polinômio interpolador de Lagrange')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Interpolação Polinomial de Lagrange (Grau < 3)')
plt.legend()
plt.grid(True)
plt.show()
```

Neste exemplo, o polinômio interpolador terá grau 2, pois temos apenas 3 pontos de interpolação.



Observação para $n = 3$:

Para $n = 3$, a condição para garantir que o polinômio interpolador tenha grau exatamente 3 é que os 4 pontos dados não devem estar em uma linha reta. Isso significa que a área do triângulo formado por quaisquer três pontos não deve ser zero. Se os pontos estiverem em uma linha reta, a matriz de Vandermonde será singular e a interpolação polinomial de Lagrange não será definida para um polinômio de grau exatamente 3.

6.2 - a)

6.2 - b)

7 - Ilustre o teorema de Laplace generalizado.

O teorema de Laplace generalizado, é uma generalização do teorema de Laplace para o cálculo de determinantes de matrizes de ordem $n \times n$.

Ele afirma que o determinante de uma matriz A de ordem $n \times n$ pode ser expresso como a soma dos produtos de n elementos, onde cada produto é formado pela escolha de um elemento de cada linha e de cada coluna de A, multiplicado pelo cofator correspondente e pelo elemento escolhido.

De forma mais precisa, para uma matriz A de ordem $n \times n$, o determinante $|A|$ é dado por:

$$|A| = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} M_{ij}$$

Onde:
- a_{ij} é o elemento da i-ésima linha e j-ésima coluna de A,
- M_{ij} é o cofator correspondente a a_{ij},
- $(-1)^{i+j}$ é o fator de alternância que alterna o sinal da soma para cada posição (i, j) da matriz.

Esse teorema fornece uma maneira sistemática de calcular determinantes de matrizes de ordem $n \times n$, tornando o processo mais eficiente do que calcular diretamente a partir da definição de determinante.

Exemplo usando Program para demonstrar o teorema de Laplace em python

Exemplo 1 - 3x3

```
def cofator(matriz, linha, coluna):
    """Calcula o cofator de um elemento da matriz."""
    submatriz = [row[:coluna] + row[coluna+1:] for row in (matriz[:linha] + matriz[linha+1:])]
    print('Submatriz')
    print(submatriz)
    return (-1) ** (linha + coluna) * determinante(submatriz)

def determinante(matriz):
```

```
    """Calcula o determinante de uma matriz."""
    if len(matriz) == 1:  # Caso base para matriz 1x1
        return matriz[0][0]
    elif len(matriz) == 2:  # Caso base para matriz 2x2
        return matriz[0][0] * matriz[1][1] - matriz[0][1] * matriz[1][0]
    else:
        det = 0
        for j in range(len(matriz)):
            det += matriz[0][j] * cofator(matriz, 0, j)
        return det

# Exemplo de uso
matriz_exemplo = [
    [1, 2, 3,4],
    [4, 5, 6],
    [7, 8, 9]
]

print("Matriz de exemplo:")
for linha in matriz_exemplo:
    print(linha)

print("Determinante:", determinante(matriz_exemplo))
```

resultado:

```
Matriz de exemplo:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Submatriz
[[5, 6],
[8, 9]]
Submatriz [[4, 6]
, [7, 9]]
Submatriz
[[4, 5]
, [7, 8]]
Determinante: 0
```

Exemplo 2 : Matriz 4x4

```
# Exemplo de uso
matriz_exemplo = [
    [1, 2, 3, 4],
    [5, 6, 7, 8],
    [9, 10, 11, 12],
    [13, 14, 15, 16]
]
print("Matriz de exemplo:")
for linha in matriz_exemplo:
    print(linha)

print("Determinante:", determinante(matriz_exemplo))
```

Resultado :

```
Matriz de exemplo:
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, 16]
Submatriz
[[6, 7, 8], [10, 11, 12], [14, 15, 16]]
Submatriz
[[11, 12], [15, 16]]
Submatriz
[[10, 12], [14, 16]]
```

Submatriz
[[10, 11], [14, 15]]
Submatriz
[[5, 7, 8], [9, 11, 12], [13, 15, 16]]
Submatriz
[[11, 12], [15, 16]]
Submatriz
[[9, 12], [13, 16]]
Submatriz
[[9, 11], [13, 15]]
Submatriz
[[5, 6, 8], [9, 10, 12], [13, 14, 16]]
Submatriz
[[10, 12], [14, 16]]
Submatriz
[[9, 12], [13, 16]]
Submatriz
[[9, 10], [13, 14]]
Submatriz
[[5, 6, 7], [9, 10, 11], [13, 14, 15]]
Submatriz
[[10, 11], [14, 15]]
Submatriz
[[9, 11], [13, 15]]
Submatriz
[[9, 10], [13, 14]]
Determinante: 0

8. Escolha para resolver dois dos itens propostos em 5) da lista Espaço Rn , ilustrando quando possível no computador.

8.1. Desenho de curvas no plano e no espaçoo dadas na forma paramétrica, em particular segmentos de reta, elipses, curvas-gráfico, limaçons, hélices.
8.2. Desenho de superfícies no espaço dadas na forma paramétrica. Casos especiais: planos, superfícies -gráfico e superfícies de revolu ̧c ̃ao

Vamos começar com as descrições algébricas e geométricas de subconjuntos do espaço euclidiano $\mathbb{R}^n$.

Formas Implícitas:
-Equações e Inequações: Estas são representações que descrevem um subconjunto definindo as condições que seus pontos devem satisfazer. Por exemplo, uma equação $f(x_1, x_2, \ldots, x_n) = 0$ pode descrever uma superfície implícita no espaço $\mathbb{R}^n$, enquanto uma inequação $g(x_1, x_2, \ldots, x_n) \leq 0$ pode descrever uma região limitada.

Formas Paramétricas:
- Representação Paramétrica: Define um subconjunto em termos de parâmetros, onde as coordenadas dos pontos são expressas como funções dos parâmetros. Por exemplo, uma curva paramétrica no plano pode ser representada como $(x(t), y(t))$, onde $x(t)$ e $y(t)$ são funções dos parâmetros t.

Análise da Dimensão:
- Dimensão dos Objetos: Refere-se ao número de coordenadas necessárias para descrever um ponto no subconjunto. Por exemplo, uma linha reta no plano tem dimensão 1, um plano tem dimensão 2 e o espaço euclidiano $\mathbb{R}^n$ tem dimensão n.

Desenhos de Curvas:
1. Curvas no Plano e no Espaço: Podem ser desenhadas com base em suas representações paramétricas. Exemplos incluem segmentos de reta, elipses, curvas de gráfico de funções, limaçons e hélices.

Desenhos de Superfícies:
1. Superfícies no Espaço: Da mesma forma, podem ser desenhadas com base em suas representações paramétricas. Planos, superfícies de gráficos de funções e superfícies de revolução (geradas pela rotação de uma curva em torno de um eixo) são exemplos comuns.

No contexto das curvas e superfícies que acabamos de discutir, podemos relacionar esses conceitos com estruturas vetoriais de várias maneiras:
Curvas no Plano e no Espaço:
1. Vetores Tangentes: Em qualquer ponto de uma curva paramétrica, podemos definir um vetor tangente que indica a direção da curva naquele ponto. Este vetor tangente é dado pela derivada das funções paramétricas que descrevem a curva em relação ao parâmetro.

2. Vetores Normais: Para curvas no plano, podemos calcular um vetor normal em cada ponto da curva. Esse vetor é perpendicular à tangente da curva e é útil em problemas como cálculo de curvatura.

Superfícies no Espaço:
1. Planos Tangentes: Em cada ponto de uma superfície, podemos definir um plano tangente que é aproximadamente o mesmo que a superfície naquele ponto. Este plano tangente é útil em cálculos de tangentes e normais em superfícies.

2. Vetores Normais: Para superfícies no espaço, podemos calcular um vetor normal em cada ponto da superfície. Este vetor normal é perpendicular à superfície e é útil em cálculos de integrais de superfície e em problemas de física envolvendo campos vetoriais.

Estruturas Vetoriais:
1. Espaço Vetorial: O conjunto de todos os vetores tangentes, normais e outras quantidades vetoriais associadas a uma curva ou superfície forma um espaço vetorial. Esse espaço tem propriedades como adição de vetores e multiplicação por escalar.

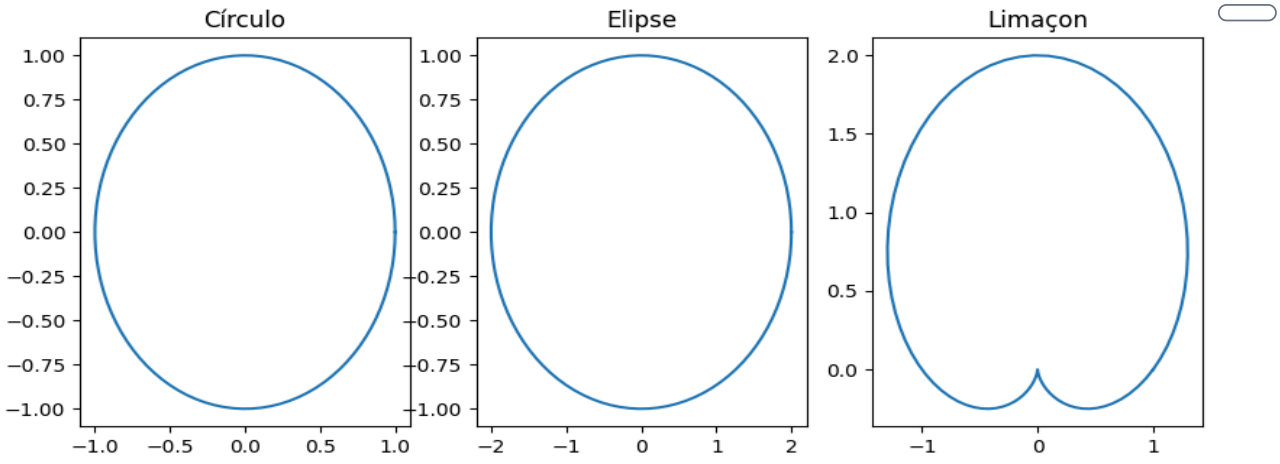
2. Operações Vetoriais: Podemos realizar operações vetoriais como soma, subtração, produto escalar e produto vetorial entre os vetores associados a curvas e superfícies. Essas operações são fundamentais em problemas de geometria diferencial e física matemática.

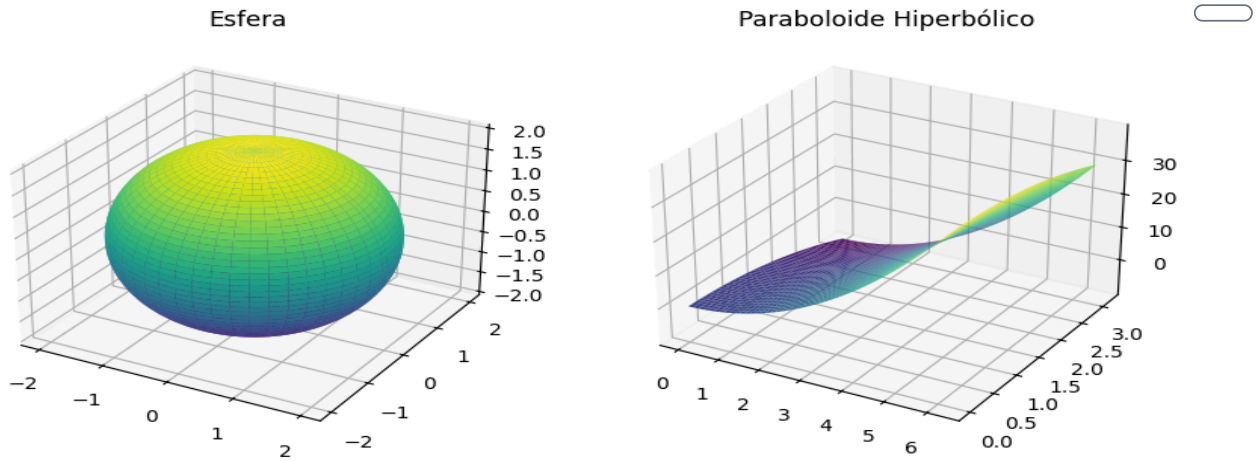
Portanto, as curvas e superfícies no plano e no espaço estão intrinsecamente relacionadas com as estruturas vetoriais, e os vetores desempenham um papel fundamental na análise e manipulação dessas entidades geométricas.

Usando Ferramentas Computacionais para plotagem de :

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Função para plotar curvas no plano
def plot_curvas():
    t = np.linspace(0, 2*np.pi, 100)  # Parâmetro para as curvas

    # Curva 1: círculo
    x1 = np.cos(t)
    y1 = np.sin(t)

    # Curva 2: elipse
    a = 2
    b = 1
    x2 = a * np.cos(t)
    y2 = b * np.sin(t)

    # Curva 3: limaçon
    r = 1 + np.sin(t)
    x3 = r * np.cos(t)
    y3 = r * np.sin(t)

    # Plotando as curvas
    plt.figure(figsize=(10, 4))
    plt.subplot(1, 3, 1)
    plt.plot(x1, y1)
    plt.title('Círculo')

    plt.subplot(1, 3, 2)
    plt.plot(x2, y2)
    plt.title('Elipse')

    plt.subplot(1, 3, 3)
    plt.plot(x3, y3)
    plt.title('Limaçon')

    plt.show()

# Função para plotar superfícies no espaço
def plot_superficies():
    u = np.linspace(0, 2 * np.pi, 100)
    v = np.linspace(0, np.pi, 100)
    U, V = np.meshgrid(u, v)
```

```
    # Superfície 1: esfera
    R = 2
    X1 = R * np.sin(V) * np.cos(U)
    Y1 = R * np.sin(V) * np.sin(U)
    Z1 = R * np.cos(V)

    # Superfície 2: paraboloide hiperbólico
    X2 = U
    Y2 = V
    Z2 = U**2 - V**2

    # Plotando as superfícies
    fig = plt.figure(figsize=(10, 5))

    ax1 = fig.add_subplot(1, 2, 1, projection='3d')
    ax1.plot_surface(X1, Y1, Z1, cmap='viridis')
    ax1.set_title('Esfera')

    ax2 = fig.add_subplot(1, 2, 2, projection='3d')
    ax2.plot_surface(X2, Y2, Z2, cmap='viridis')
    ax2.set_title('Paraboloide Hiperbólico')

    plt.show()

# Chamando as funções para plotar curvas e superfícies
plot_curvas()
plot_superficies()
```





9. Escolha um outro problema interessante para resolver envolvendo sistemas, matrizes e determinantes

Suponha que um banco oferece cinco tipos de investimentos para seus clientes: conservador (C), moderado (M), arrojado (A), agressivo (AG) e ultra-agressivo (UA). A cada mês, um cliente pode manter seu investimento atual ou trocar para outro tipo de investimento, de acordo com as seguintes probabilidades de transição:

Um cliente conservador tem 80% de chance de permanecer conservador, 10% de chance de mudar para um investimento moderado, 5% de chance de mudar para um investimento arrojado, 3% de chance de mudar para um investimento agressivo e 2% de chance de mudar para um investimento ultra-agressivo.

Um cliente moderado tem 70% de chance de permanecer moderado, 15% de chance de mudar para um investimento conservador, 10% de chance de mudar para um investimento arrojado, 3% de chance de mudar para um investimento agressivo e 2 % de chance de mudar para um investimento ultra-agressivo.

Um cliente arrojado tem 60% de chance de permanecer arrojado, 15% de chance de mudar para um investimento conservador, 10 % de chance de mudar para um investimento moderado, 10% de chance de mudar para um investimento agressivo e 5% de chance de mudar para um investimento ultra-agressivo.

Um cliente agressivo tem 50% de chance de permanecer agressivo, 20% de chance de mudar para um investimento conservador, 10% de chance de mudar para um investimento moderado, 10% de chance de mudar para um investimento arrojado e 10% de chance de mudar para um investimento ultra-agressivo.

Um cliente ultra-agressivo tem 40% de chance de permanecer ultra-agressivo, 20% de chance de mudar para um investimento conservador, 15% de chance de mudar para um investimento moderado, 15% de chance de mudar para um investimento arrojado e 10% de chance de mudar para um investimento agressivo.

O problema consiste em determinar a distribuição dos clientes entre os cinco tipos de investimento após um certo número de meses e calcular o market share de cada tipo de investimento.

Agora vamos desenvolver a solução matemática e implementar em Python.

```
import numpy as np

def transicao_inicial():
    # Definir a matriz de transição inicial
    P = np.array([[0.8, 0.1, 0.05, 0.03, 0.02],
                  [0.15, 0.7, 0.1, 0.03, 0.02],
                  [0.15, 0.1, 0.6, 0.1, 0.05],
                  [0.2, 0.1, 0.1, 0.5, 0.1],
                  [0.2, 0.15, 0.15, 0.1, 0.4]])
    return P

def calcular_market_share(P, distribuicao_inicial, meses):
    # Inicializar a distribuição de mercado com a distribuição inicial
    market_share = distribuicao_inicial.copy()

    # Calcular a matriz de transição elevada a n
    P_n = np.linalg.matrix_power(P, meses)
    print('matriz de transiçao inicial')
    print(P)
    print('numero de meses')
    print(meses)
    print('matriz de transiçao apos os meses')
    print(P_n)
    # Calcular o market share após n meses
    market_share_final = np.dot(market_share, P_n)

    return market_share_final


# Calcular a matriz de transição inicial
P = transicao_inicial()

# Distribuição inicial de mercado
distribuicao_inicial = np.array([0.2, 0.2, 0.2, 0.2, 0.2])

# Número de meses
meses = int(input("Digite o número de meses para calcular o market share: "))

# Calcular o market share após n meses
market_share_final = calcular_market_share(P, distribuicao_inicial, meses)
```

```
tipos_investimento = ['Conservador', 'Moderado', 'Arrojado', 'Agressivo', 'Ultra-agressivo']
# Exibir o market share inicial
print("Market Share inicial")
for i, tipo in enumerate(tipos_investimento):
    print(f"{tipo}: {distribuicao_inicial[i]*100:.2f}%")
print(f"Market Share após {meses} meses:")


for i, tipo in enumerate(tipos_investimento):
    print(f"{tipo}: {market_share_final[i]*100:.2f}%")
```

Vamos desenvolver as soluções matemáticas para os cenários de curto prazo (3 meses), médio prazo (9 meses) e longo prazo (24 meses) usando a cadeia de Markov.

Curto Prazo (3 meses):

Para o cenário de curto prazo, podemos calcular a distribuição dos clientes entre os tipos de investimento após 3 meses usando a matriz de transição elevada a essa potência. A matriz de transição P já foi definida anteriormente. Portanto, podemos calcular $P3$ para encontrar a distribuição dos clientes.

Médio Prazo (9 meses):

Para o cenário de médio prazo, calcularemos a distribuição dos clientes após 9 meses. Podemos fazer isso elevando a matriz de transição P à potência de 9: $P9$.

Longo Prazo (24 meses):

Para o cenário de longo prazo, calcularemos a distribuição dos clientes após 24 meses. Podemos fazer isso elevando a matriz de transição P à potência de 24: {P}24.

Essas soluções nos fornecerão as distribuições dos clientes entre os tipos de investimento para os cenários de curto, médio e longo prazo.

Resoluções dos cenários via python

```
matriz de transiçao inicial
[[0.8 0.1 0.05 0.03 0.02]
[0.15 0.7 0.1 0.03 0.02]
[0.15 0.1 0.6 0.1 0.05]
[0.2 0.1 0.1 0.5 0.1 ]
[0.2 0.15 0.15 0.1 0.4 ]]
numero de meses 3
matriz de transiçao apos os meses
[[0.59115 0.198175 0.11095 0.061255 0.03847 ]
[0.316775 0.41425 0.1609 0.06703 0.041045]
[0.325125 0.20075 0.288 0.119775 0.06635 ]
[0.3705 0.20405 0.16205 0.17835 0.08505 ]
[0.36325 0.235225 0.186475 0.11015 0.1049 ]]
Market Share inicial
Conservador: 20.00%
Moderado: 20.00%
Arrojado: 20.00%
Agressivo: 20.00%
Ultra-agressivo: 20.00%
Market Share após 3 meses:
Conservador: 39.34%
Moderado: 25.05%
Arrojado: 18.17%
Agressivo: 10.73%
Ultra-agressivo: 6.72%

Cenário Para Médio Prazo 9 meses
matriz de transiçao inicial
[[0.8 0.1 0.05 0.03 0.02]
[0.15 0.7 0.1 0.03 0.02]
[0.15 0.1 0.6 0.1 0.05]
```

[0.2 0.1 0.1 0.5 0.1]
[0.2 0.15 0.15 0.1 0.4]]
numero de meses 9
matriz de transiçao apos os meses
[[0.45745847 0.25329007 0.15660312 0.08268226 0.04996608]
[0.43758173 0.26360356 0.16294599 0.08483029 0.05103843]
[0.44080212 0.25436242 0.16486648 0.08747869 0.05249029]
[0.44492771 0.25464463 0.16185562 0.08652166 0.05205038]
[0.44337679 0.2561237 0.16254262 0.0861042 0.05185269]]
Market Share inicial
Conservador: 20.00%
Moderado: 20.00%
Arrojado: 20.00%
Agressivo: 20.00%
Ultra-agressivo: 20.00%
Market Share após 9 meses:
Conservador: 44.48%
Moderado: 25.64%
Arrojado: 16.18%
Agressivo: 8.55%
Ultra-agressivo: 5.15%

Cenário para Longo Prazo 24 meses

matriz de transiçao inicial
[[0.8 0.1 0.05 0.03 0.02]
[0.15 0.7 0.1 0.03 0.02]
[0.15 0.1 0.6 0.1 0.05]
[0.2 0.1 0.1 0.5 0.1]
[0.2 0.15 0.15 0.1 0.4]]
numero de meses 24
matriz de transiçao apos os meses
[[0.44792712 0.25636232 0.16029582 0.08449816 0.05091658]
[0.44790395 0.25636885 0.16030515 0.08450295 0.0509191]
[0.44791126 0.25636484 0.16030284 0.08450228 0.05091878]
[0.44791509 0.25636445 0.16030107 0.08450119 0.0509182]
[0.44791286 0.25636521 0.16030193 0.08450159 0.0509184]]
Market Share inicial
Conservador: 20.00%
Moderado: 20.00%
Arrojado: 20.00%
Agressivo: 20.00%
Ultra-agressivo: 20.00%
Market Share após 24 meses:
Conservador: 44.79%
Moderado: 25.64%
Arrojado: 16.03%
Agressivo: 8.45%
Ultra-agressivo: 5.09%

10. Estude e ilustre com exemplos o processo de inversão de matrizes usando matrizes elementares.

O processo de inversão de matrizes usando matrizes elementares consiste em realizar uma sequência de operações elementares nas linhas (ou colunas) de uma matriz para transformá-la na matriz identidade, enquanto aplica essas mesmas operações em uma matriz identidade correspondente. Quando a matriz original se torna a identidade, a matriz resultante da identidade se torna a inversa da matriz original.

Passos do processo de inversão de matriz usando matrizes elementares:

1. Forme uma matriz aumentada: Junte a matriz que você deseja inverter com a matriz identidade correspondente.

2. Aplique operações elementares: Realize operações elementares nas linhas da matriz aumentada até que a matriz original se torne a identidade. Aplique as mesmas operações na matriz identidade correspondente.

3. Resultado final: Quando a matriz original se tornar a identidade, a matriz inversa estará do outro lado da barra vertical, ou seja, a parte da matriz identidade que originalmente era separada da matriz original.

Exemplo:
Vamos considerar a seguinte matriz que desejamos inverter:

$$A = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}$$

Primeiro, criaremos uma matriz aumentada, juntando A com a matriz identidade I:

$$\left[\begin{array}{cc|cc} 2 & 1 & 1 & 0 \\ 4 & 3 & 0 & 1 \end{array}\right]$$

Agora, aplicaremos operações elementares para transformar a parte A da matriz aumentada na matriz identidade. Vamos começar subtraindo duas vezes a primeira linha da segunda linha para fazer o elemento a_{21} (que é 4) se tornar zero:

$$\left[\begin{array}{cc|cc} 2 & 1 & 1 & 0 \\ 0 & 1 & -2 & 1 \end{array}\right]$$

Agora, subtrairemos a segunda linha da primeira linha para tornar o elemento a_{12} (que é 1) zero:

$$\left[\begin{array}{cc|cc} 2 & 0 & 3 & -1 \\ 0 & 1 & -2 & 1 \end{array}\right]$$

Finalmente, dividimos a primeira linha por 2 para fazer o elemento a_{11} (que é 2) se tornar 1:

$$\left[\begin{array}{cc|cc} 1 & 0 & 3/2 & -1/2 \\ 0 & 1 & -2 & 1 \end{array}\right]$$

Agora, do lado direito da barra vertical, temos a inversa da matriz A:

$$A^{-1} = \begin{bmatrix} 3/2 & -1/2 \\ -2 & 1 \end{bmatrix}$$

Este é o processo básico de inversão de matriz usando matrizes elementares.

Demonstração em Sistema Computacional usando python

```
import numpy as np

def matriz_inversa(matriz):
  n = len(matriz)

  # Criar uma matriz aumentada [matriz | Identidade]
  matriz_aumentada = np.hstack([matriz, np.eye(n)])
  print('matriz aumentada')
  print(matriz_aumentada)
  # Aplicar operações elementares para transformar a parte esquerda em uma matriz identidade
  for i in range(n):
    # Dividir a linha i pelo elemento diagonal para tornar o elemento diagonal 1
    elemento_diagonal = matriz_aumentada[i][i]
    matriz_aumentada[i] /= elemento_diagonal

    # Zerar todos os elementos abaixo do elemento diagonal
    for j in range(i + 1, n):
      fator_diagonal = matriz_aumentada[j][i]
      matriz_aumentada[j] -= fator_diagonal * matriz_aumentada[i]
      print('fator diagonal')
      print(fator_diagonal)
      print('matriz aumentada')
      print(matriz_aumentada)


  # Zerar todos os elementos acima do elemento diagonal
  for i in range(n - 1, -1, -1):
    for j in range(i - 1, -1, -1):
      fator_diagonal = matriz_aumentada[j][i]
      matriz_aumentada[j] -= fator_diagonal * matriz_aumentada[i]
      print('fator diagonal')
      print(fator_diagonal)
      print('matriz aumentada')
```

```
        print(matriz_aumentada)


    # Extrair a parte direita (a inversa) da matriz aumentada
    inversa = matriz_aumentada[:, n:]

    return inversa

# Teste com a matriz fornecida no exemplo
matriz = np.array([[2, 1], [4, 3]])
inversa = matriz_inversa(matriz)
print("Matriz inversa:")
print(inversa)
```

Resultado do programa

```
matriz aumentada
[[2. 1. 1. 0.]
[4. 3. 0. 1.]]
fator diagonal 4.0
matriz aumentada
[[ 1. 0.5 0.5 0. ]
[ 0. 1. -2. 1. ]]
fator diagonal 0.5
matriz aumentada
[[ 1. 0. 1.5 -0.5]
[ 0. 1. -2. 1. ]]
Matriz inversa:
[[ 1.5 -0.5]
[-2. 1. ]]
```