Trabalho 07

Nome da Atividade: Lista de Exercícios

Nome e Matrícula: Lucas Lima do Nascimento - 12111ECP024

1. Um sistema de referência móvel UVW foi rotacionado de 90 graus em torno do eixo Z do sistema de referência global XYZ. Calcular as coordenadas do vetor rXYZ quando rUVW = [-3, 4, -11].

Para calcular as coordenadas do vetor rXYZ a partir do vetor rUVW, é necessário utilizar uma matriz de rotação para aplicar a rotação de 90 graus no eixo Z.

A matriz de rotação para uma rotação de 90 graus em torno do eixo Z é:

[cos(90), -sin(90), 0]
[sin(90), cos(90), 0]
[0, 0, 1]

Ou, em termos de uma matriz 3x3:

[0, -1, 0]
[1, 0, 0]
[0, 0, 1]

Para calcular as coordenadas do vetor rXYZ, basta multiplicar essa matriz pela matriz coluna formada pelas coordenadas do vetor rUVW:

[-3]
[4]
[-11]

O resultado será:

[4]
[-3]
[-11]

Portanto, as coordenadas do vetor rXYZ são [4, -3, -11].

2. Calcular o vetor resultante da translação do vetor rXYZ = [4, 4, 11] segundo o vetor p=[6, 3, 8].

Para calcular o vetor resultante da translação do vetor rXYZ pelo vetor p, basta adicionar os dois vetores componente a componente:

[4 + 6]
[4 + 3]
[11 + 8]

O resultado será:

[10]
[7]
[19]

Portanto, o vetor resultante da translação é [10, 7, 19].

---

3. Um sistema de referência móvel UVW foi transladado um vetor p=[8, -4, 5] e, posteriormente, rotacionado de 90 graus em torno do eixo X do sistema de referência global XYZ. Calcular as coordenadas do vetor rXYZ sendo que rUVW = [7, 3,12].

Para calcular as coordenadas do vetor rXYZ a partir do vetor rUVW, primeiro é preciso aplicar a translação pelo vetor p. Isso pode ser feito adicionando o vetor p componente a componente:

[7 + 8]
[3 - 4]
[12 + 5]

O resultado será:

[15]
[-1]
[17]

Agora, para rotacionar esse vetor em torno do eixo X do sistema de referência global XYZ, precisamos utilizar uma matriz de rotação. A matriz de rotação para uma rotação de 90 graus

em torno do eixo X é:

[1, 0, 0]
[0, cos(90), -sin(90)]
[0, sin(90), cos(90)]

Ou, em termos de uma matriz 3x3:

[1, 0, 0]
[0, 0, -1]
[0, 1, 0]

Para calcular as coordenadas do vetor rXYZ, basta multiplicar essa matriz pela matriz coluna formada pelas coordenadas do vetor rUVW:

[15]
[-1]
[17]

O resultado será:

[15]
[17]
[-1]

Portanto, as coordenadas do vetor rXYZ são [15, 17, -1].

---

4. Translação: escrever um programa de computador que recebe as coordenadas do ponto P a ser transladado. A translação deve ser de 2 unidades em x, 3 unidades em y, e 5 unidades em z. Apresentar a matriz de transformação e plotar as novas coordenadas.

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

# Cria a matriz de translação
T = np.array([[1, 0, 0, 2],
              [0, 1, 0, 3],
              [0, 0, 1, 5],
              [0, 0, 0, 1]])
```

```
# Cria a matriz coluna com as coordenadas do ponto P
P = np.array([[2],
              [3],
              [5],
              [1]])

# Calcula a matriz resultante da translação
result = T.dot(P)

# Configura o ambiente de plotagem
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot o ponto P antes da translação (em azul)
ax.scatter(2, 3, 5, c='b', marker='o')

# Plot o ponto P após a translação (em vermelho)
ax.scatter(result[0], result[1], result[2], c='r', marker='o')

# Exibe o gráfico
plt.show()
```

---

5. Translações sucessivas: escrever um programa que recebe x0, y0 e z0 da matriz de transformação T0, x1, y1 e z1 da matriz de transformação T1. Realizar a transformação sucessiva T0.T1 e a transformação T1.T0 sobre o ponto P=(3,4,7). Apresentar os resultados destas duas transformações sucessivas.

```
import numpy as np

# Lê os valores de x0, y0, z0, x1, y1 e z1
x0 = float(input("Insira o valor de x0: "))
y0 = float(input("Insira o valor de y0: "))
z0 = float(input("Insira o valor de z0: "))

x1 = float(input("Insira o valor de x1: "))
y1 = float(input("Insira o valor de y1: "))
z1 = float(input("Insira o valor de z1: "))

# Cria as matrizes de transformação T0 e T1
T0 = np.array([[1, 0, 0, x0],
               [0, 1, 0, y0],
               [0, 0, 1, z0],
               [0, 0, 0, 1]])

T1 = np.array([[1, 0, 0, x1],
               [0, 1, 0, y1],
               [0, 0, 1, z1],
               [0, 0, 0, 1]])

# Cria a matriz coluna com as coordenadas do ponto P
```

```
P = np.array([[3],
              [4],
              [7],
              [1]])

# Realiza as transformações sucessivas e imprime os resultados
result1 = T0.dot(T1).dot(P)
print("Transformação T0.T1:", result1)

result2 = T1.dot(T0).dot(P)
print("Transformação T1.T0:", result2)
```

---

6. Rotação: Escrever um programa que recebe o ângulo teta a ser rotacionado em torno do eixo z e calcula as novas coordenadas do ponto P=(4,5,6) após sua rotação. Plotar as novas coordenadas.

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

# Lê o valor do ângulo teta
teta = float(input("Insira o valor do ângulo teta: "))

# Cria a matriz de rotação
R = np.array([[cos(teta), -sin(teta), 0, 0],
              [sin(teta), cos(teta), 0, 0],
              [0, 0, 1, 0],
              [0, 0, 0, 1]])

# Cria a matriz coluna com as coordenadas do ponto P
P = np.array([[4],
              [5],
              [6],
              [1]])

# Calcula a matriz resultante da rotação
result = R.dot(P)

# Configura o ambiente de plotagem
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot o ponto P antes da rotação (em azul)
ax.scatter(4, 5, 6, c='b', marker='o')

# Plot o ponto P após a rotação (em vermelho)
ax.scatter(result[0], result[1], result[2], c='r', marker='o')

# Exibe o gráfico
plt.show()
```

---

7. Rotações sucessivas: escrever um programa que recebe o ângulo teta da primeira rotação R1, e o ângulo alfa da segunda rotação R2 a serem aplicados sobre o ponto P=(6,6,8). Plotar as novas coordenadas.

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

# Lê os valores dos ângulos teta e alfa
teta = float(input("Insira o valor do ângulo teta: "))
alfa = float(input("Insira o valor do ângulo alfa: "))

# Cria as matrizes de rotação
R1 = np.array([[cos(teta), -sin(teta), 0, 0],
               [sin(teta), cos(teta), 0, 0],
               [0, 0, 1, 0],
               [0, 0, 0, 1]])

R2 = np.array([[1, 0, 0, 0],
               [0, cos(alfa), -sin(alfa), 0],
               [0, sin(alfa), cos(alfa), 0],
               [0, 0, 0, 1]])

# Cria a matriz coluna com as coordenadas do ponto P
P = np.array([[6],
              [6],
              [8],
              [1]])

# Calcula a matriz resultante da rotação
result = R1.dot(R2).dot(P)

# Configura o ambiente de plotagem
fig = plt.figure()
ax = fig.
```

7. Translação e rotação sucessivas: escrever um programa que recebe x e y da matriz de translação T e o ângulo teta da matriz de rotação R a serem aplicados sobre o ponto P=(4,5,7). Plotar o resultado da transformação sucessiva T.R e o resultado da transformação sucessiva R.T.

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

# Lê os valores de x e y da translação e o ângulo teta da rotação
x = float(input("Insira o valor de x: "))
y = float(input("Insira o valor de y: "))
```

```
teta = float(input("Insira o valor do ângulo teta: "))

# Cria as matrizes de transformação
T = np.array([[1, 0, 0, x],
              [0, 1, 0, y],
              [0, 0, 1, 0],
              [0, 0, 0, 1]])

R = np.array([[cos(teta), -sin(teta), 0, 0],
              [sin(teta), cos(teta), 0, 0],
              [0, 0, 1, 0],
              [0, 0, 0, 1]])

# Cria a matriz coluna com as coordenadas do ponto P
P = np.array([[4],
              [5],
              [7],
              [1]])

# Calcula as matrizes resultantes das transformações
result1 = T.dot(R).dot(P)
result2 = R.dot(T).dot(P)

# Configura o ambiente de plotagem
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot o ponto P antes da transformação (em azul)
ax.scatter(4, 5, 7, c='b', marker='o')

# Plot o ponto P após a transformação T.R (em vermelho)
ax.scatter(result1[0][0], result1[1][0], result1[2][0], c='r', marker='o')

# Plot o ponto P após a transformação R.T (em verde)
ax.scatter(result2[0][0], result2[1][0], result2[2][0], c='g', marker='o')

# Exibe o gráfico
plt.show()
```