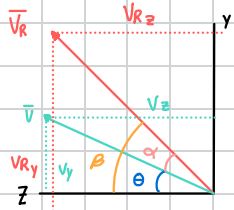
# Tarea 2: Matrices de rotación y representación en Python

## Deducir $R_x(\beta)$




Por proyección del vector z.

$V_z = -|V| \cos\theta$

$V_y = |V| \sin\theta$

Fórmula de desglose de ángulo:

Inverso

$\cos(a \pm b) = \cos(a)\cos(b) \mp \sin(a)\sin(b)$

$\sin(a \pm b) = \sin(a)\cos(b) \pm \cos(a)\sin(b)$

$V_{Rz} = -|V| \cos\beta$

$V_{Rz} = -|V| \cos(\alpha + \theta)$

$V_{Rz} = -|V| [\cos\alpha \cdot \cos\theta - \sin\alpha \cdot \sin\theta]$

$V_{Rz} = -|V| \cos\alpha \cdot \cos\theta + |V| \sin\alpha \cdot \sin\theta$

$V_{Rz} = V_z \cdot \cos\alpha + V_y \cdot \sin\alpha$

$V_{Ry} = |V| \sin\beta$

$V_{Ry} = |V| \sin(\alpha + \theta)$

$V_{Ry} = |V| [\sin\alpha \cdot \cos\theta + \cos\alpha \cdot \sin\theta]$

$V_{Ry} = |V| \sin\alpha \cdot \cos\theta + |V| \cos\alpha \cdot \sin\theta$

$V_{Ry} = -V_z \cdot \sin\alpha + V_y \cdot \cos\alpha$

We miss the negative of $V_z$

$$\begin{bmatrix} V_{Ry} \\ V_{Rz} \end{bmatrix} = \begin{bmatrix} -V_z \sin\alpha + V_y \cdot \cos\alpha \\ V_z \cos\alpha + V_y \cdot \sin\alpha \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} V_{Ry} \\ V_{Rz} \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & +\cos\alpha \end{bmatrix} \begin{bmatrix} V_y \\ V_z \end{bmatrix}$$

$$\downarrow$$

$$R_x(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & +\cos\alpha \end{bmatrix}$$

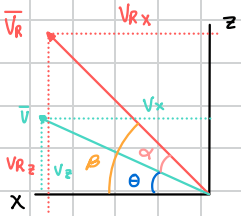
## Código de Animación Python [$R_x(\beta)$]:

```
# Import libraries and packages
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np

# create the fig and ax objects to handle figure and axes of the fixed frame
fig,ax = plt.subplots()

# Use 3d view
ax = plt.axes(projection = "3d")

def setaxis(x1, x2, y1, y2, z1, z2):
    ax.set_xlim3d(x1,x2)
    ax.set_ylim3d(y1,y2)
    ax.set_zlim3d(z1,z2)
    ax.view_init(elev=30, azim=40)

def fix_system(axis_length):
    x = [-axis_length, axis_length]
    y = [-axis_length, axis_length]
    z = [-axis_length, axis_length]
    zp = [0, 0]
    ax.plot3D(x, zp, zp, color='red')
    ax.plot3D(zp, y, zp, color='blue')
    ax.plot3D(zp, zp, z, color='green')

def sind(t):
    res = np.sin(t*np.pi/180)
    return res

def cosd(t):
    res = np.cos(t*np.pi/180)
```

```
    return res
def RotZ(t):
    Rz = np.array(([1,0,0],[0,cosd(t),-sind(t)],[0,sind(t),cosd(t)]))
    return Rz
def drawVector(v):
    deltaX = [0, v[0]]
    deltaY = [0, v[1]]
    deltaZ = [0, v[2]]
    ax.plot3D(deltaX, deltaY, deltaZ,color='orange')
    #plt.draw()
    #plt.pause(0.001)

def rotate(t):
    n = 0
    while n < t:
        ax.cla()

        # Set the view
        setaxis(-1,1,-1,1,-1,1)

        # plot the axis
        fix_system(1)

        # draw vector1
        v1 = np.array([0,1,1])
        drawVector(v1)

        # draw vector2
        v2 = RotZ(n).dot(v1)
        drawVector(v2)

        n = n + 1
        plt.draw()
```

```
        plt.pause(0.001)

rotate(45)

# show image.
plt.draw()
plt.show()
```

# Deducir $R_y(\beta)$




$V_x = -|V| \cdot \cos\theta$

$V_z = |V| \cdot \sin\theta$

Fórmula de desglose de ángulo:

Inverso

$\cos(a \pm b) = \cos(a)\cos(b) \mp \sin(a)\sin(b)$

$\sin(a \pm b) = \sin(a)\cos(b) \pm \cos(a)\sin(b)$

$V_{Rx} = -|V| \cdot \cos\beta$

$V_{Rx} = -|V| \cdot \cos(\theta+\alpha)$

$V_{Rx} = -|V| [\cos\theta \cdot \cos\alpha - \sin\theta \cdot \sin\alpha]$

$V_{Rx} = -|V| \cos\theta \cdot \cos\alpha + |V| \sin\theta \cdot \sin\alpha$

$V_{Rx} = V_x \cdot \cos\alpha + V_z \cdot \sin\alpha$

$V_{Rz} = |V| \cdot \sin\beta$

$V_{Rz} = |V| \cdot \sin(\theta+\alpha)$

$V_{Rz} = |V| [\sin\theta\cos\alpha + \cos\theta\sin\alpha]$

$V_{Rz} = |V| \sin\theta\cos\alpha + |V|\cos\theta\sin\alpha$

$V_{Rz} = V_z \cdot \cos\theta - V_x \cdot \sin\alpha$

$$\begin{bmatrix} V_{Rx} \\ V_{Rz} \end{bmatrix} = \begin{bmatrix} V_x\cos\alpha + V_z\sin\alpha \\ -V_x\sin\alpha + V_z\cos\alpha \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} V_{Rx} \\ V_{Rz} \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} V_x \\ V_z \end{bmatrix}$$

$$\downarrow$$

$$R_y(\beta) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

## Código de Animación Python [$R_y(\beta)$]:

```
# Import libraries and packages
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np

# create the fig and ax objects to handle figure and axes of the fixed frame
fig,ax = plt.subplots()

# Use 3d view
ax = plt.axes(projection = "3d")

def setaxis(x1, x2, y1, y2, z1, z2):
    ax.set_xlim3d(x1,x2)
    ax.set_ylim3d(y1,y2)
    ax.set_zlim3d(z1,z2)
    ax.view_init(elev=30, azim=40)

def fix_system(axis_length):
    x = [-axis_length, axis_length]
    y = [-axis_length, axis_length]
    z = [-axis_length, axis_length]
    zp = [0, 0]
    ax.plot3D(x, zp, zp, color='red')
    ax.plot3D(zp, y, zp, color='blue')
    ax.plot3D(zp, zp, z, color='green')

def sind(t):
    res = np.sin(t*np.pi/180)
    return res
def cosd(t):
    res = np.cos(t*np.pi/180)
    return res
```

```
def RotZ(t):
    Rz = np.array(([cosd(t),0,sind(t)],[0,1,0],[-sind(t),0,cosd(t)]))
    return Rz

def drawVector(v):
    deltaX = [0, v[0]]
    deltaY = [0, v[1]]
    deltaZ = [0, v[2]]
    ax.plot3D(deltaX, deltaY, deltaZ,color='orange')

def rotate(t):
    n = 0
    while n < t:
        ax.cla()

        # Set the view
        setaxis(-1,1,-1,1,-1,1)

        # plot the axis
        fix_system(1)

        # draw vector1
        v1 = np.array([1,0,1])
        drawVector(v1)

        # draw vector2
        v2 = RotZ(n).dot(v1)
        drawVector(v2)

        n = n + 1
        plt.draw()
        plt.pause(0.001)
```

```
rotate(45)

# show image.
plt.draw()
plt.show()
```

## Transpuesta de $R_z(\beta)$

$$R_z(\beta) = \begin{bmatrix} \cos\alpha & -\operatorname{sen}\alpha & 0 \\ \operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{T} R_z^T(\beta) = \begin{bmatrix} \cos\alpha & \operatorname{sen}\alpha & 0 \\ -\operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## $R_z(\beta) \times R_z^T(\beta)$

$$\underbrace{\begin{bmatrix} \cos\alpha & -\operatorname{sen}\alpha & 0 \\ \operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_z(\beta)} \times \underbrace{\begin{bmatrix} \cos\alpha & \operatorname{sen}\alpha & 0 \\ -\operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_z^T(\beta)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$C_{1,1} = (\cos\alpha)(\cos\alpha) + (-\operatorname{sen}\alpha)(-\operatorname{sen}\alpha) + (0)(0) = \boxed{\cos^2\alpha + \operatorname{sen}^2\alpha = 1}$ Identidad trigonométrica

$C_{1,2} = (\cos\alpha)(\operatorname{sen}\alpha) + (-\operatorname{sen}\alpha)(\cos\alpha) + (0)(0) = \operatorname{sen}\alpha\cos\alpha - \operatorname{sen}\alpha\cos\alpha = 0$

$C_{1,3} = (\cos\alpha)(0) + (-\operatorname{sen}\alpha)(0) + (0)(1) = 0$

$C_{2,1} = (\operatorname{sen}\alpha)(\cos\alpha) + (\cos\alpha)(-\operatorname{sen}\alpha) + (0)(0) = \operatorname{sen}\alpha\cos\alpha - \operatorname{sen}\alpha\cos\alpha = 0$

$C_{2,2} = (\operatorname{sen}\alpha)(\operatorname{sen}\alpha) + (\cos\alpha)(\cos\alpha) + (0)(0) = \cos^2\alpha + \operatorname{sen}^2\alpha = 1$

$C_{2,3} = (\operatorname{sen}\alpha)(0) + (\cos\alpha)(0) + (0)(1) = 0$

$C_{3,1} = (0)(\cos\alpha) + (0)(-\operatorname{sen}\alpha) + (1)(0) = 0$

$C_{3,2} = (0)(\operatorname{sen}\alpha) + (0)(\cos\alpha) + (1)(0) = 0$

$C_{3,3} = (0)(0) + (0)(0) + (1)(1) = 1$

**Conclusión:** al multiplicar una matriz original o estándar por su transpuesta nos dará como resultado una matriz primitiva debido que se cancelarán una de la inversa y cuando estas se eliminen son igual a 1.