Representing pose in 3D

Daniel Stangeland
s331489

September 18, 2020

In Figure 1, 2, 3, 4, we can see examples of different poses in 3 dimensions.

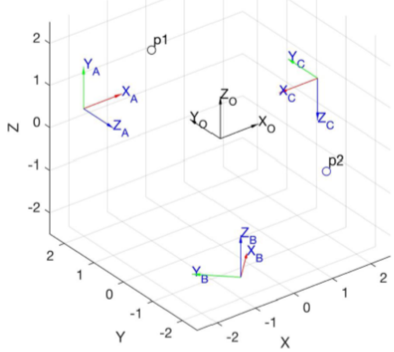


Figure 1: 3D view

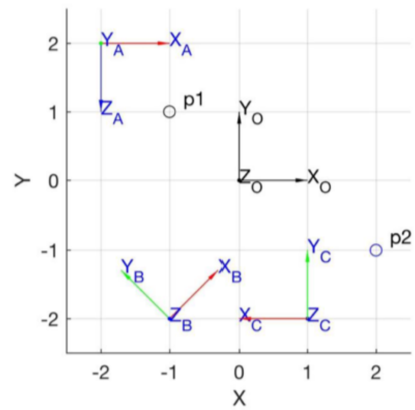


Figure 2: Top view

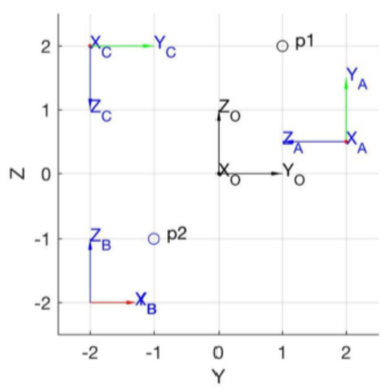


Figure 3: Projection on X axis

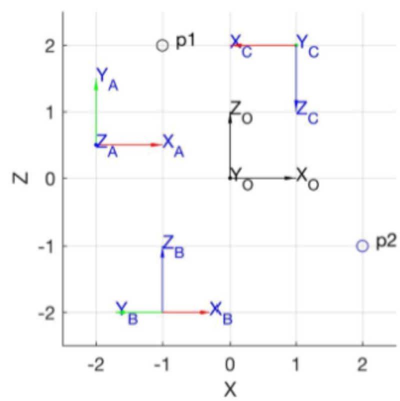


Figure 4: Projection on Y axis

Contents

1 Exercise 1

1.1 a) Determine positions of two points

Using figures 1, 2, 3, 4, determine the positions of the points 1 and 2, denoted by circles in the following reference frames:

$$^{O}p_1, ^{A}p_1, ^{B}p_1, ^{C}p_1$$
$$^{O}p_2, ^{A}p_2, ^{B}p_2, ^{C}p_2$$

We know that a point in reference to a known frame is denoted as

$$^{A}p = \begin{bmatrix} ^{A}p_x \\ ^{A}p_y \\ ^{A}p_z \end{bmatrix} \tag{1}$$

A representing the reference frame

With this information combined with figures we denote the points in the following way

$$^{O}p_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix} \qquad ^{O}p_2 = \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix}$$

$$^{A}p_1 = \begin{bmatrix} 1 \\ \frac{3}{2} \\ 1 \end{bmatrix} \qquad ^{A}p_2 = \begin{bmatrix} 4 \\ -\frac{3}{2} \\ 3 \end{bmatrix}$$

$$^{B}p_1 = \begin{bmatrix} \frac{3\sqrt{2}}{2} \\ \frac{3\sqrt{2}}{2} \\ 4 \end{bmatrix} \qquad ^{B}p_2 = \begin{bmatrix} 2\sqrt{2} \\ -\sqrt{2} \\ 1 \end{bmatrix}$$

$$^{C}p_1 = \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix} \qquad ^{C}p_2 = \begin{bmatrix} -1 \\ 1 \\ 3 \end{bmatrix}$$

1.2 b) Determine the homogeneous transformation matrices ${}^{O}T_A, {}^{O}T_B, {}^{O}T_C$

The homogeneous transformation matrix is as follows

$$ {}^{A}T_B = \begin{pmatrix} {}^{A}\boldsymbol{R}_B & {}^{A}\boldsymbol{t}_B \\ \boldsymbol{0}_{1X3} & 1 \end{pmatrix} \tag{2} $$

$$ {}^{A}\boldsymbol{R}_B = \begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} & \boldsymbol{c} \end{bmatrix} \in \mathbb{R}^{3X3} \quad {}^{A}\boldsymbol{t}_B = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \boldsymbol{0}_{1X3} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \tag{3} $$

The total rotation matrix is determined by looking at one rotation at a time, from A to B, where each rotation has its own matrix denoted R_x, R_y, R_z. and are commonly known as the Elementary Rotational Matrices.

$$ R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix} \quad R_y = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \quad R_z = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4} $$

To find the total rotation $\boldsymbol{R}$ each of these rotational matrices are matrix multiplied together.The order of the multiplication is critical since rotations in 3D are non commutative, meaning $R_1 \cdot R_2 \neq R_2 \cdot R_1$.

$$ \boldsymbol{R} = R_x \cdot R_y \cdot R_z \tag{5} $$

${}^{O}T_A$ has a positive $\theta = \frac{\pi}{2}$ rotation around the X axis with translation ${}^{O}\boldsymbol{t}_A = \begin{bmatrix} -2 \\ 2 \\ \frac{1}{2} \end{bmatrix}$

$$ {}^{O}T_A = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & cos(\frac{\pi}{2}) & -sin(\frac{\pi}{2}) & 2 \\ 0 & sin(\frac{\pi}{2}) & cos(\frac{\pi}{2}) & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 0 & -1 & 2 \\ 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

${}^{O}T_B$ has a positive $\theta = \frac{\pi}{4}$ rotation around the Z axis with translation ${}^{O}\boldsymbol{t}_B = \begin{bmatrix} -1 \\ -2 \\ -2 \end{bmatrix}$

$$ {}^{O}T_B = \begin{bmatrix} cos(\frac{\pi}{4}) & -sin(\frac{\pi}{4}) & 0 & -1 \\ sin(\frac{\pi}{4}) & cos(\frac{\pi}{4}) & 0 & -2 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & -1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & -2 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

${}^{O}T_C$ has a positive $\theta = \pi$ rotation around the Y axis with translation ${}^{O}\boldsymbol{t}_C = \begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix}$

$$ {}^{O}T_C = \begin{bmatrix} cos(\pi) & 0 & sin(\pi) & 1 \\ 0 & 1 & 0 & -2 \\ -sin(\pi) & 0 & cos(\pi) & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

1.3 c) Determine the homogeneous transformation matrices ${}^{B}T_{C}, {}^{C}T_{A}, {}^{A}T_{B}$

${}^{B}T_{C}$ has its first rotation of $\theta_{Y_B} = \pi$ and its second rotation of $\theta_{Z_B} = \frac{\pi}{4}$ with a translation ${}^{B}\boldsymbol{t}_{C} = \begin{bmatrix} \sqrt{2} \\ -\sqrt{2} \\ 4 \end{bmatrix}$. Computing $R = R_y(\pi) \cdot R_z(\frac{\pi}{4})$ gives the following homogeneous transform matrix:

$$
{}^{B}T_{C} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & \sqrt{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & -\sqrt{2} \\ 0 & 0 & -1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

${}^{C}T_{A}$ has its first rotation of $\theta_{Y_C} = \pi$ and its second rotation of $\theta_{X_C} = \frac{\pi}{2}$ with a translation ${}^{C}\boldsymbol{t}_{A} = \begin{bmatrix} 3 \\ 4 \\ \frac{3}{2} \end{bmatrix}$. Computing $R = R_y(\pi) \cdot R_x(\frac{\pi}{2})$ gives the following homogeneous transform matrix:

$$
{}^{C}T_{A} = \begin{bmatrix} -1 & 0 & 0 & 3 \\ 0 & 0 & -1 & 4 \\ 0 & -1 & 0 & \frac{3}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

${}^{A}T_{B}$ has its first rotation of $\theta_{X_A} = -\frac{\pi}{2}$ and its second rotation of $\theta_{Z_A} = \frac{\pi}{4}$ with a translation ${}^{A}\boldsymbol{t}_{B} = \begin{bmatrix} 1 \\ -\frac{5}{2} \\ 4 \end{bmatrix}$. Computing $R = R_x(-\frac{\pi}{2}) \cdot R_z(\frac{\pi}{4})$ gives the following homogeneous transform matrix:

$$
{}^{A}T_{B} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 1 \\ 0 & 0 & 1 & -\frac{5}{2} \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

1.4 d) Using the obtained transformation matrices ${}^{O}T_A, {}^{O}T_B, {}^{O}T_C$ and ${}^{O}p_1, {}^{O}p_2$ from the diagram determine position vectors:

$$
{}^{A}p_1, {}^{B}p_1, {}^{C}p_1
$$
$$
{}^{A}p_2, {}^{B}p_2, {}^{C}p_2
$$

To find the position vectors we use the following relation

$$
{}^{A}\tilde{p}_1 = {}^{A}T_O \cdot {}^{O}\tilde{p}_1 = ({}^{O}T_A)^{-1} \cdot {}^{O}\tilde{p}_1 \tag{6}
$$

Where $\tilde{p}$ denotes the homogeneous representation of vector p

$$
p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \longrightarrow \tilde{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{7}
$$

To compute this we use python. The results from each computation should be equal to the results found in Exercise 1a.

[6]:
```python
import numpy as np
import robotteknikk as rob
```

[21]:
```python
# Exercise 1.d

# Define homogeneous transformation matrices

oTa = np.array([[1, 0, 0, -2],
                [0, 0, -1, 2],
                [0, 1, 0, 1/2],
                [0, 0, 0, 1]])

oTb = np.array([[np.sqrt(2)/2, -np.sqrt(2)/2, 0, -1],
                [np.sqrt(2)/2, np.sqrt(2)/2, 0, -2],
                [0, 0, 1, -2],
                [0, 0, 0, 1]])

oTc = np.array([[-1, 0, 0, 1],
               [0, 1, 0, -2],
               [0, 0, -1, 2],
               [0, 0, 0, 1]])

# Define vectors
Op1 = np.array([-1, 1, 2])
Op2 = np.array([2, -1, -1])
```

[26]:
```python
# Determine Ap1

Ap1_h = np.linalg.inv(oTa).dot(rob.e2h(Op1))
Ap1 = rob.h2e(Ap1_h)
Ap1
```

[26]:
```
array([1. , 1.5, 1. ])
```

[27]:
```python
# Determine Bp1

Bp1_h = np.linalg.inv(oTb).dot(rob.e2h(Op1))
Bp1 = rob.h2e(Bp1_h)
Bp1
```

[27]:
```
array([2.12132034, 2.12132034, 4.        ])
```

[28]:
```python
# Determine Cp1

Cp1_h = np.linalg.inv(oTc).dot(rob.e2h(Op1))
Cp1 = rob.h2e(Cp1_h)
Cp1
```

[28]:
```
array([2., 3., 0.])
```

[29]:
```python
# Determine Ap2

Ap2_h = np.linalg.inv(oTa).dot(rob.e2h(Op2))
Ap2 = rob.h2e(Ap2_h)
Ap2
```

[29]:
```
array([ 4. , -1.5,  3. ])
```

[30]:
```python
# Determine Ap2

Bp2_h = np.linalg.inv(oTb).dot(rob.e2h(Op2))
Bp2 = rob.h2e(Bp2_h)
Bp2
```

[30]:
```
array([ 2.82842712, -1.41421356,  1.        ])
```

[31]:
```python
# Determine Ap2

Cp2_h = np.linalg.inv(oTc).dot(rob.e2h(Op2))
Cp2 = rob.h2e(Cp2_h)
Cp2
```

[31]:
```
array([-1.,  1.,  3.])
```

Results from the python script is equal to those found in Exercise 1a.

2 Exercise 2

Determine if the following are valid rotation matrices

$$R_a = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_b = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_c = \begin{bmatrix} -0.5 & 0 & 0.866 \\ 0 & 1 & 0 \\ 0.866 & 0 & -0.5 \end{bmatrix}$$

The following properties of a rotational matrix are used to determine if the matrix is valid or not.

$$det(R) = +1 \quad R^T R = I \tag{8}$$

[45]:
```
import numpy as np
import robotteknikk as rob
```

[33]:
```
def isRotMat(R):
    Rdim = R.shape
    RtR = R.T.dot(R)
    tol = 1e-5
    detR = np.linalg.det(R)

    #checks if RtR and detR is close to the identity matrix and 1.
    if np.allclose(RtR,np.eye(Rdim[0]), rtol=tol, atol=tol) and np.
 ↪isclose(detR,1,rtol=tol, atol=tol):

        print("det(R) = ",detR,"\n")
        print("R^TR = \n",RtR.round(3))
        print("\nR =\n",R,"\n\nIs a valid rotation matrix\n")
    else:

        print("det(R) = ",detR,"\n")
        print("R^TR = \n",RtR.round(3))
        print("\nR =\n", R,"\n\nIs NOT avalid rotation matrix\n")
```

[34]:
```
Ra = np.array([[1, 1, 0],
               [0, 0, 0],
               [0, 0, 1]])

isRotMat(Ra)
```

```
det(R) =  0.0

R^TR =
 [[1 1 0]
 [1 1 0]
 [0 0 1]]

R =
 [[1 1 0]
 [0 0 0]
 [0 0 1]]

Is NOT avalid rotation matrix
```

[44]:
```
Rb = np.array([[np.sqrt(2)/2, -np.sqrt(2)/2, 0],
               [np.sqrt(2)/2, np.sqrt(2)/2, 0],
               [0,0,1]])
isRotMat(Rb)
```

```
det(R) =  1.0

R^TR =
 [[ 1. -0.  0.]
 [-0.  1.  0.]
 [ 0.  0.  1.]]

R =
 [[ 0.70710678 -0.70710678  0.        ]
 [ 0.70710678  0.70710678  0.        ]
 [ 0.          0.          1.        ]]

Is a valid rotation matrix
```

[56]:
```
Rc = np.array([[-0.5, 0, 0.866],
               [0, 1, 0],
               [0.866,0,-0.5]])
isRotMat(Rc)
```

```
det(R) =  -0.499956

R^TR =
 [[ 1.     0.    -0.866]
 [ 0.     1.     0.   ]
 [-0.866  0.     1.   ]]

R =
 [[-0.5    0.     0.866]
 [ 0.     1.     0.   ]
 [ 0.866  0.    -0.5  ]]

Is NOT avalid rotation matrix
```

3 Exercise 3

Consider the following rotation matrix

$$R(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3.1 a) Show that R(0) = I_3 where I_3 is the identity matrix of dimensions 3

We know that

$$sin0 = 0 \quad \text{and} \quad cos0 = 1$$

Plugging this ($\theta = 0$) into the rotation matrix $R(\theta)$

$$R(0) = \begin{bmatrix} cos0 & -sin0 & 0 \\ sin0 & cos0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_3$$

3.2 b) Show that $det(R(\theta)) = +1$ for any angle θ

To show that $det(R(\theta)) = +1$ for any angle the following trigonometric property in combination with the formula for finding the determinant of a 3x3-matrix is used.

$$sin^2\theta + cos^2\theta = 1$$

and

$$\begin{aligned} det(A) = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} &= a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ &= a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}) \end{aligned}$$

The determinant for the given rotational matrix is therefore as follows

$$\begin{aligned} det(R) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} &= cos\theta \begin{vmatrix} cos\theta & 0 \\ 0 & 1 \end{vmatrix} - (-sin\theta) \begin{vmatrix} sin\theta & 0 \\ 0 & 1 \end{vmatrix} + 0 \begin{vmatrix} sin\theta & cos\theta \\ 0 & 0 \end{vmatrix} \\ &= cos\theta(cos\theta - 0) + sin\theta(sin\theta - 0) + 0 \\ &= cos^2\theta + sin^2\theta \\ &= 1 \end{aligned}$$

As shown above the determinate of a rotational 3x3-matrix is always 1.

3.3 c) Show that $R(\theta)^T R(\theta) = I_3$

First we compute $R(\theta)^T$

$$R(\theta)^T = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} \cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then we multiply $R(\theta)^T R(\theta)$

$$\begin{aligned} R(\theta)^T \cdot R(\theta) &= \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} (cos^2\theta + sin^2\theta) & (-sin\theta cos\theta + sin\theta cos\theta) & 0 \\ (-sin\theta cos\theta + sin\theta cos\theta) & (sin^2\theta + cos^2\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= I_3 \end{aligned}$$

3.4 d) Show that the columns of $R(\theta)$ are orthonormal

Two columns being orthonormal means they are orthogonal to each other and their length corresponds to the length of a unit-vector (unit norm).

We describe the rotational matrix R as follows:

$$R(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} & \boldsymbol{c} \end{bmatrix} \in \mathbb{R}^{3X3}$$

To compute the length of each column the following formula is used. In this case the unit-vector should equal 1.

$$||\boldsymbol{a}||^2 = \sqrt{(a_1)^2 + (a_2)^2 + (a_3)^2} = 1$$
$$||\boldsymbol{b}||^2 = \sqrt{(b_1)^2 + (b_2)^2 + (b_3)^2} = 1$$
$$||\boldsymbol{c}||^2 = \sqrt{(a_1)^2 + (c_2)^2 + (c_3)^2} = 1$$

Two columns is said to be orthogonal if the dot product equals 0.

$$\boldsymbol{a^T b} = 0, \quad \boldsymbol{b^T c} = 0, \quad \boldsymbol{c^T a} = 0.$$

We apply this to the given rotational matrix and compute the length of each column and its dot products.

$$||\boldsymbol{a}||^2 = \sqrt{cos^2\theta + sin^2\theta + 0} = \sqrt{1} = 1$$
$$||\boldsymbol{b}||^2 = \sqrt{(-sin\theta)^2 + cos^2\theta + 0} = \sqrt{1} = 1$$
$$||\boldsymbol{c}||^2 = \sqrt{0 + 0 + 1^2} = \sqrt{1} = 1$$

$$\boldsymbol{a^T b} = \begin{bmatrix} cos\theta & sin\theta & 0 \end{bmatrix} \cdot \begin{bmatrix} -sin\theta \\ cos\theta \\ 0 \end{bmatrix} = \begin{bmatrix} -sin\theta cos\theta + sin\theta cos\theta + 0 \end{bmatrix} = 0$$
$$\boldsymbol{b^T c} = \begin{bmatrix} -sin\theta & cos\theta & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 + 0 + 0 \end{bmatrix} = 0$$
$$\boldsymbol{c^T a} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos\theta \\ sin\theta \\ 0 \end{bmatrix} = \begin{bmatrix} 0 + 0 + 0 \end{bmatrix} = 0$$

The results show that the columns of rotational matrix R are orthonormal.

3.5 e) Show that the rows of $R(\theta)$ are orthonormal

This proof is identical to Exercise 3.d, where the only difference is working with rows instead of columns.

4 Exercise 4

Consider the transforms

$$T_1 = \begin{bmatrix} R_1 & t_1 \\ 0_{1x3} & 1 \end{bmatrix} T_2 = \begin{bmatrix} R_2 & t_2 \\ 0_{1x3} & 1 \end{bmatrix}$$

Where

$$t_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad t_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \quad 0_{1x3} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Show that

$$T_1T_2 = \begin{bmatrix} R_1R_2 & t_1 + R_1t_2 \\ 0_{1x3} & 1 \end{bmatrix}$$

This can be shown by preforming matrix multiplication. To be able multiply the different "inner" matrices the dimensions of each multiplication need to be valid.

$$T_1T_2 = \begin{bmatrix} R_1 & t_1 \\ 0_{1x3} & 1 \end{bmatrix} \cdot \begin{bmatrix} R_2 & t_2 \\ 0_{1x3} & 1 \end{bmatrix} = \begin{bmatrix} R_1R_2 + (t_1 \cdot 0_{1X3}) & t_1 + R_1t_2 \\ (0_{1X3} \cdot R_2) + 0_{1X3} & (0_{1X3} \cdot t_2) + 1 \end{bmatrix}$$

Where

$$(t_1 0_{1X3}) = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$(0_{1X3} \cdot R_2) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

$$(0_{1X3} \cdot t_2) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = 0$$

With matrix addition we can now simplify T_1T_2

$$T_1T_2 = \begin{bmatrix} R_1R_2 + (t_1 \cdot 0_{1X3}) & t_1 + R_1t_2 \\ (0_{1X3} \cdot R_2) + 0_{1X3} & (0_{1X3} \cdot t_2) + 1 \end{bmatrix} = \begin{bmatrix} R_1R_2 & t_1 + R_1t_2 \\ 0_{1x3} & 1 \end{bmatrix}$$

5 Exercise 5

Show that $T^{-1} = \begin{bmatrix} R^T & -R^Tt \\ 0_{1X3} & 1 \end{bmatrix}$ is the inverse transform of $T = \begin{bmatrix} R & t \\ 0_{1X3} & 1 \end{bmatrix}$.

The inverse of a matrix A exists if the following is true

$$AA^{-1} = A^{-1}A = I_n \tag{9}$$

By using this property we can show that T^{-1} is the inverse of T.

$$\begin{aligned}
TT^{-1} &= \begin{bmatrix} R & t \\ 0_{1X3} & 1 \end{bmatrix} \cdot \begin{bmatrix} R^T & -R^Tt \\ 0_{1X3} & 1 \end{bmatrix} \\
&= \begin{bmatrix} RR^T + (t \cdot 0_{1X3}) & (R \cdot (-R^Tt) + t) \\ (0_{1X3} \cdot R^T) + 0_{1X3} & (0_{1X3} \cdot (-R^Tt) + 1) \end{bmatrix} \\
&= \begin{bmatrix} RR^T + (t \cdot 0_{1X3}) & (-RR^Tt) + t \\ (0_{1X3} \cdot R^T) + 0_{1X3} & (0_{1X3} \cdot -R^Tt + 1) \end{bmatrix}
\end{aligned}$$

Where

$$\begin{aligned}
RR^T &= \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} cos^2\theta + sin^2\theta + 0 & cos\theta sin\theta - cos\theta sin\theta + 0 & 0+0+0 \\ cos\theta sin\theta - cos\theta sin\theta & cos^2\theta + sin^2\theta & 0+0+0 \\ 0+0+0 & 0+0+0 & 0+0+1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_3 \\
RR^Tt = I_2t &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = t \\
(t_1 \cdot 0_{1X3}) &= \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
(0_{1X3} \cdot R^T) &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \\
(0_{1X3} \cdot -R^Tt) &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \cdot - \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} -xcos\theta - ysin\theta + 0 \\ xsin\theta - ycos\theta + 0 \\ 0+0+z \end{bmatrix} = 0
\end{aligned}$$

With matrix addition we can now simplify TT^{-1}

$$TT^{-1} = \begin{bmatrix} RR^T + (t \cdot 0_{1X3}) & (-RR^Tt) + t \\ (0_{1X3} \cdot R^T) + 0_{1X3} & (0_{1X3} \cdot -R^Tt + 1) \end{bmatrix} = \begin{bmatrix} I_3 & (-t+t) \\ 0_{1X3} & 1 \end{bmatrix} = \begin{bmatrix} I_3 & 0_{3X1} \\ 0_{1X3} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = I_3$$

This confirms that $T^{-1} = \begin{bmatrix} R^T & -R^Tt \\ 0_{1X3} & 1 \end{bmatrix}$ is the inverse transform of $T = \begin{bmatrix} R & t \\ 0_{1X3} & 1 \end{bmatrix}$

6 Exercise 6

Given the roll, pitch, yaw angles ($\theta_r, \theta_p, \theta_y$ the corrosponding rotation matrix is given by

$$R = R_x(\theta_r)R_y(\theta_p)R_z(\theta_y)$$

Where

$$R_x(\theta_r) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix} \quad R_y(\theta_p) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \quad R_z(\theta_y) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Write a function in python that returns this rotation R matrix with roll, pitch, yaw angles as arguments in the function

[4]:
```
import numpy as np
import robotteknikk as rob

def rpy2r(roll, pitch, jaw):
    """This fuction takes roll, pitch and jaw angles
        and converts them to a rotational matrix."""

    # Rotational matrix for roll angle (around X)
    Rx = np.array([[1, 0, 0],
                   [0, np.cos(roll), -np.sin(roll)],
                   [0, np.sin(roll), np.cos(roll)]])
    # Rotational matrix for pitch angle (around Y)
    Ry = np.array([[np.cos(pitch), 0, np.sin(pitch)],
                   [0, 1, 0],
                   [-np.sin(pitch), 0, np.cos(pitch)]])
    # Rotational matrix for jaw angle (around Z)
    Rz = np.array([[np.cos(yaw), -np.sin(yaw), 0],
                   [np.sin(yaw), np.cos(yaw), 0],
                   [0, 0, 1]])

    # Rotates around x-axis, then new y-axis and then z-axis
    R = Rx.dot(Ry).dot(Rz)

    return R
```

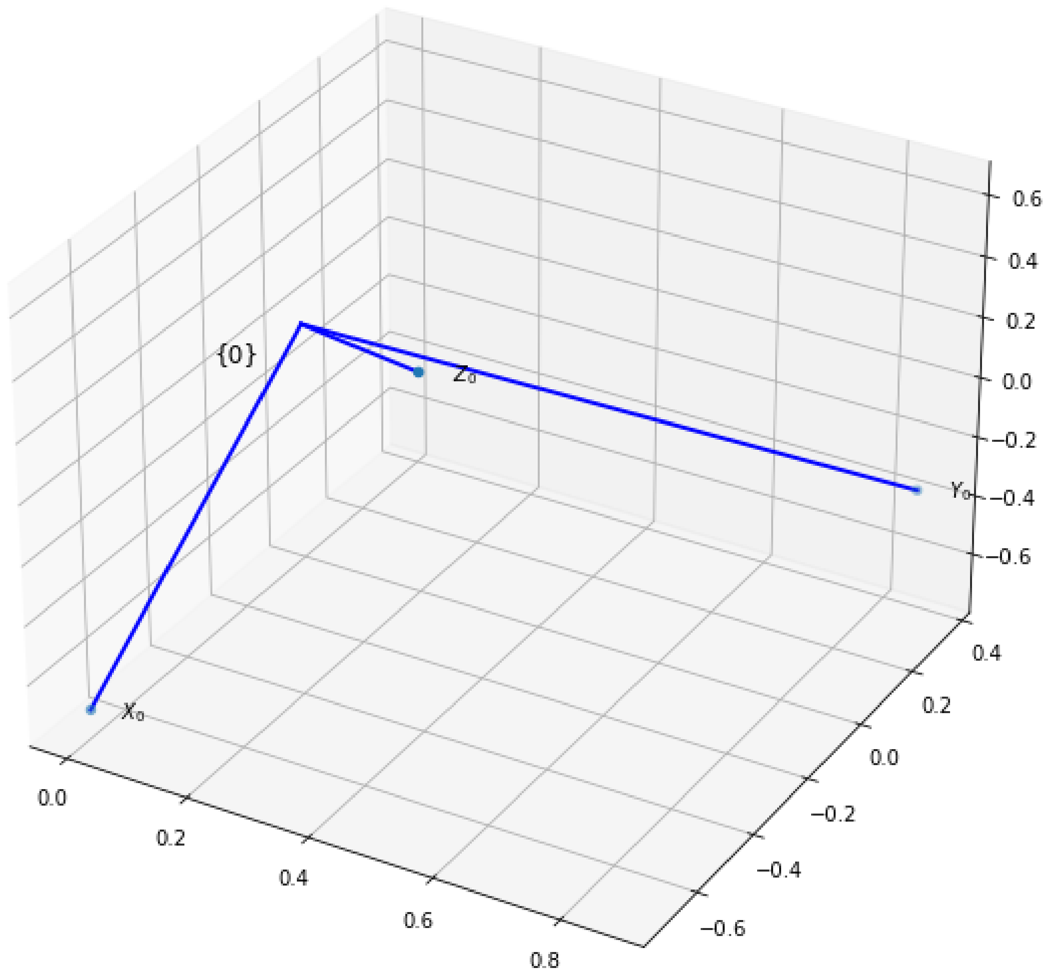
7 Exercise 7

Convert the following roll, pitch and yaw angels $(\theta_r, \theta_p, \theta_y) = (\frac{\pi}{4}, \frac{\pi}{6}, -\frac{\pi}{2})$

7.1 a) Determine equivalent rotation matrix and plot it

[44]:
```
import numpy as np
import matplotlib.pyplot as plt
import robotteknikk as rob

def tpr(R):
    """This function returns a homogeneous pure rotation matrix"""

    n = R.shape[0] # finds shape of first column, either 2 or 3.
    if n == 2:
        # Creates a identity matrix of 3 dimensions
        T = np.eye(3)
        # Append the identity matrix rows and columns with R input
        T[0:2,0:2] = R
    elif n == 3:
        # Creates a identity matrix of 4 dimensions
        T = np.eye(4)
        # Append the identity matrix rows and columns with R input
        T[0:3,0:3] = R
    else:
        print("Invalid rotation matrix. Please check that dimensions are correct")
    return T
```

[56]:
```
# Define the roll, pitch and yaw angles
roll = np.pi/4
pitch = np.pi/6
yaw = -np.pi/2

# Create a rotation matrix based on rpy
R = rpy2r(roll, pitch, yaw)

# To plot the rotational matrix it must be converted to the homogeneous pure rotation␣
 ↪matrix
T = tpr(R)

# Define the plot and use robotteknikk's plotting function
fig = plt.figure(1, figsize=(10,10))
ax = plt.axes(projection='3d')
rob.trplot3(ax,T,name="0",color="b")
```


{0}
Z_0
Y_0
X_0
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.0
0.2
0.4
0.6
0.8

7.2 b) Determine the equivalent unit quaternion

By using the following relationship between a rotational matrix and the unit quaternion

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \longrightarrow \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \frac{1}{4q_0} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{10}$$

Where

$$q_0 = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}} \tag{11}$$

A function can easily be made in python. The function takes a arbitrary rotational matrix as input and returns the equivalent unit quaternion.

[28]:
```
R =   ([[ 0.    ,  0.866,  0.5  ],
        [-0.707,  0.354, -0.612],
        [-0.707, -0.354,  0.612]])
```

[31]:
```
def r2quat(R, print=None):
    """
    This function converts the rotation matrix to
    its equvivalent unit quaternion expression

    A Quaternion is a hyper complex number:
            a + ib + jc + ke

    Represented as:
            q = s + v(i,j,k)

    Where s is the real part and v is the imaginary part
    """
    s = 1/2*np.sqrt(1+R[0,0]+R[1,1]+R[2,2])
    q = np.array([[R[2,1]-R[1,2]],
                  [R[0,2]-R[2,0]],
                  [R[1,0]-R[0,1]]])
    v = 1/(4*s) * q

    if print is not None:
        print(f"{s} <{q.T}>")
    return s, v

s, v = r2quat(R, print)
```

```
0.7010706098532444 <[[ 0.258  1.207 -1.573]]>
```

7.3 c) Determine the equivalent vector-angle representation

To determine the angle vector representation of a rotation matrix, the eigenvalues and eigenvectors of that matrix is used.

The eigenvector containing only real values is the vector representation $\boldsymbol{v}$. The angle representation θ is the angle of the eigenvalues, $\lambda = \cos\theta \pm i \sin\theta$.

This can be computed using python.

[28]:
```
R = ([[ 0.    ,  0.866,  0.5  ],
      [-0.707,  0.354, -0.612],
      [-0.707, -0.354,  0.612]])
```

[225]:
```
# Exercise 7.c

def r2angvec(R):
    """
    This function converts a rotation matrix to the equivalent
    vector-angle representation
    """
    # Finds the eigen values and vectors of R
    eigvalues, eigvectors = np.linalg.eig(R)

    # Extract imaginary parts from eigenvectors
    v0 = np.imag(eigvectors[:,0])
    v1 = np.imag(eigvectors[:,1])
    v2 = np.imag(eigvectors[:,2])

    # Creates an array of zeroes in the same dimension
    zeros = np.zeros(eigvectors.shape[0])

    # Identifies the vector with no imaginart parts.
    if (v0 == zeros).all():
        vector = np.real(eigvectors[:,0])
    elif (v1 == zeros).all():
        vector = np.real(eigvectors[:,1])
    elif (v2 == zeros).all():
        vector = np.real(eigvectors[:,2])
    else:
        print("No eigenvector with only real parts")

    # Identifies index of theta
    i = 0
    while np.angle(eigvalues[i]) <= 0:
        i = i+1

    angle = np.angle(eigvalues[i])

    return angle, vector

theta, v = r2angvec(R)
theta, v
```

[225]:
```
(1.587834237962121, array([-0.12942831, -0.603641  ,  0.78668027]))
```

8 Exercise 8

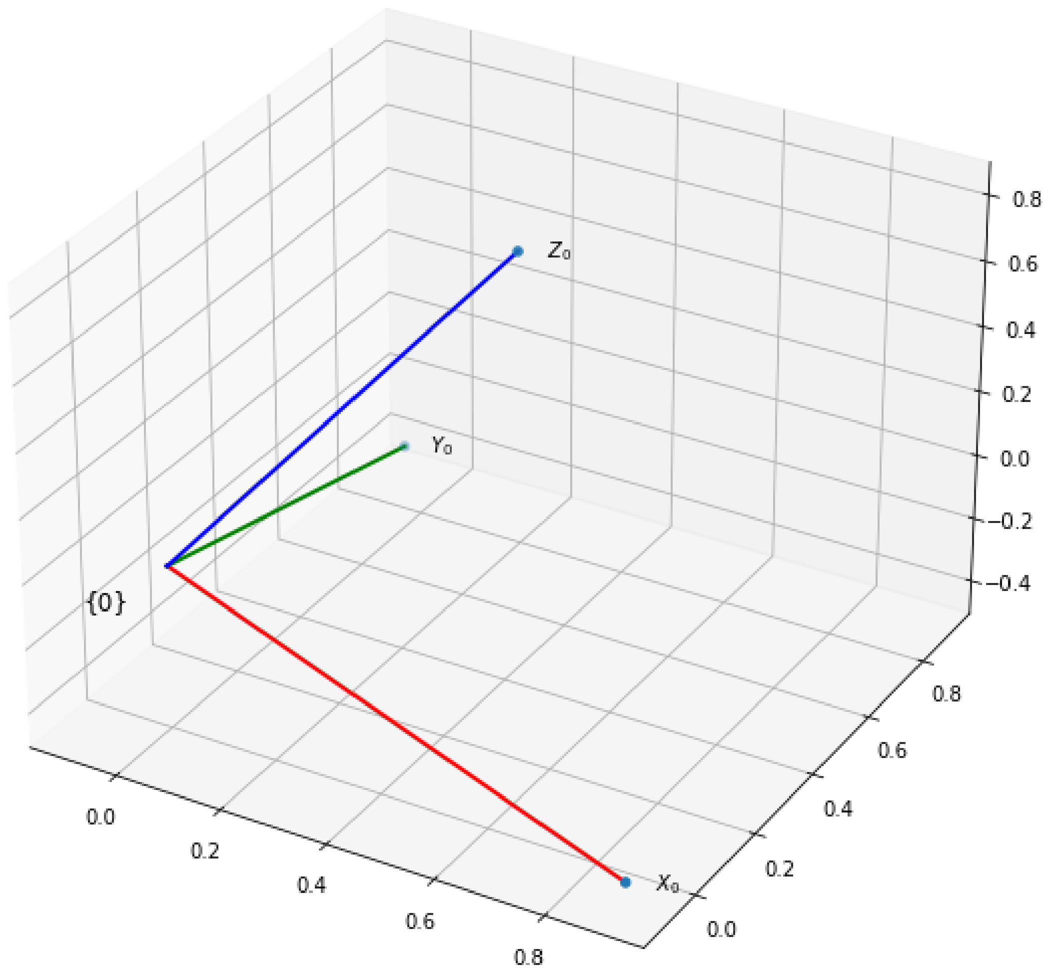
Given a rotation of $\theta = \frac{\pi}{5}$ along axis $(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0)$.

8.1 a) Determine the equivalent rotation matrix and plot it

[240]:
```
# Exercise 8.a

def skew(v):
    """
    Returns a skew symmetric matric from a vector
    w with dimensions 3x1
    """
    vx = v[0]
    vy = v[1]
    vz = v[2]
    skew = np.array([[0,-vz,vy],
                     [vz,0,-vx],
                     [-vy,vx,0]])
    return skew

def angvec2r(theta, v):
    """
    This function computes a rotational matrix from
    the angle vector.

    Theta is in radians and vector is of dimensions 3x1

    Using the Rodrigues formula:
        R = I_{3X3} + sin(theta) S(vector) + (1 - cos(theta))(vv.T - I_{3X3})
    Where S is the skew matrix
    """
    n = v.shape[0] # Defines the shape of vector array
    I = np.eye(n) # Creates an identity matrix of n-size

    s = np.sqrt(v[0]**2+v[1]**2+v[2]**2) # Computes the length of v
    # Can also use s = np.sqrt(v.dot(v))
    v = v/s # Renormalize v, it now has unit length
    S = skew(v) # Returns the skew matrix made form vector array

    # Computes the rotational matrix by using Rodrigues formula
    R = I + np.sin(theta) * S + (1 - np.cos(theta))*(np.outer(v,v) - I)
    return R

def plotR(R):
    # To plot the rotational matrix it must be converted to the homogeneous pure␣
↪rotation matrix
    T = tpr(R)

    # Define the plot and use robotteknikk's plotting function
    fig = plt.figure(1, figsize=(10,10))
    ax = plt.axes(projection='3d')
    rob.trplot3(ax,T,name="0")
```

```
theta = np.pi/5
v = np.array([-np.sqrt(2)/2, np.sqrt(2)/2,0])

R = angvec2r(theta, v)
plotR(R)
R.round(3)
```

[240]:
```
array([[ 0.905, -0.095,  0.416],
       [-0.095,  0.905,  0.416],
       [-0.416, -0.416,  0.809]])
```



8.2 b) Determine equivalent unit quaternion

[243]:
```python
R.round(3)
```

[243]:
```
array([[ 0.905, -0.095,  0.416],
       [-0.095,  0.905,  0.416],
       [-0.416, -0.416,  0.809]])
```

[241]:
```python
# Exercise 8.b

def r2quat(R, print=None):
    """
    This function converts the rotation matrix to
    its equuivalent unit quaternion expression

    A Quaternion is a hyper complex number:
            a + ib + jc + ke

    Represented as:
            q = s + v(i,j,k)

    Where s is the real part and v is the imaginary part
    """
    s = 1/2*np.sqrt(1+R[0,0]+R[1,1]+R[2,2])
    q = np.array([[R[2,1]-R[1,2]],
                  [R[0,2]-R[2,0]],
                  [R[1,0]-R[0,1]]])
    v = 1/(4*s) * q

    if print is not None:
        print(f"{s} <{q.T}>")
    return s, v

s, v = r2quat(R, print)
```

```
0.9510565162951535 <[[-0.83125388  0.83125388  0.        ]]>
```