Modelling and Control of Manipulators - Assignment 1

GROUP 16

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY,
BIOENGINEERING, ROBOTICS AND SYSTEM ENGINEERING

MODELLING AND CONTROL OF MANIPULATORS

First Assignment

Equivalent representations of orientation matrices

Author:

Tateyama Kohei
Hidayat Hanif Furqon
Milous Ouassim

Student ID:

s6427214
s5807387
s5938924

Professors:

Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Francesco Giovinazzo
George Kurshakov

October 31, 2023

Contents

Mathematical expression	Definition	MATLAB expression
$< w >$	World Coordinate Frame	w
${}^{a}_{b}R$	Rotation matrix of frame $< b >$ with respect to frame $< a >$	aRb
${}^{a}_{b}T$	Transformation matrix of frame $< b >$ with respect to frame $< a >$	aTb

Table 1: Nomenclature Table

1 Assignment description

The first assignment of Modelling and Control of Manipulators focuses on the geometric fundamentals and algorithmic tools underlying any robotics application. The concepts of transformation matrix, orientation matrix and the equivalent representations of orientation matrices (Equivalent angle-axis representation, Euler Angles and Quaternions) will be reviewed.

The first assignment is **mandatory** and consists of 4 different exercises. You are asked to:

- Download the .zip file called MOCOM-LAB1 from the Aulaweb page of this course.
- Implement the code to solve the exercises on MATLAB by filling the predefined files called "*main.m*", "*ComputeAngleAxis.m*", "*ComputeInverseAngleAxis.m*", and "*QuatToRot.m*".
- Write a report motivating the answers for each exercise, following the predefind format on this document.

1.1 Exercise 1 - Equivalent Angle-Axis Representation (Exponential representation)

A particularly interesting minimal representation of 3D rotation matrices is the so-called "*angle-axis representation*" or "*exponential representation*". Given two frames $< a >$ and $< b >$, initially coinciding, let's consider an applied geometric unit vector $(\mathbf{v}, O_a) = (\mathbf{v}, O_b)$, passing through the common origin of the two frames, whose initial projection on $< a >$ is the same of that on $< b >$. Then let's consider that frame $< b >$ is purely rotated around **v** of an angle θ, even negative, accordingly with the right-hand rule. We note that the axis-line defined by $(\mathbf{v}, O_a) = (\mathbf{v}, O_b)$ remains common to both the reference systems of the two frames $< a >$ and $< b >$ and we obtain that the orientation matrix constructed in the above way is said to be represented by its equivalent angle-axis representation that admits the following equivalent analytical expression, also known as Rodrigues Formula:

$$\mathbf{R}(^*\mathbf{v}, \theta) = e^{[^*\mathbf{v}\times]\theta} = e^{[\rho\times]} = \mathbf{I}_{3x3} + [^*\mathbf{v}\times]\sin(\theta) + [^*\mathbf{v}\times]^2(1 - \cos(\theta))$$

Q1.1 Given two generic frames $< a >$ and $< b >$, given the geometric unit vector $(\mathbf{v}, O_a) = (\mathbf{v}, O_b)$ and the angle θ, implement on MATLAB the Rodrigues formula, computing the rotation matrix ${}^a_bR$ of frame $< b >$ with respect to $< a >$.

Then test it for the following cases and comment the results obtained, including some sketches of the frames configurations:

- **Q1.2** $\mathbf{v} = [1, 0, 0]$ and $\theta = 45°$
- **Q1.3** $\mathbf{v} = [0, 1, 0]$ and $\theta = \pi/6$
- **Q1.4** $\mathbf{v} = [0, 0, 1]$ and $\theta = 3\pi/4$
- **Q1.5** $\mathbf{v} = [0.3202, 0.5337, 0.7827]$ and $\theta = 2.8$
- **Q1.6** $\boldsymbol{\rho} = [0, 2\pi/3, 0];$
- **Q1.7** $\boldsymbol{\rho} = [0.25, -1.3, 0.15];$
- **Q1.8** $\boldsymbol{\rho} = [-\pi/4, -\pi/3, \pi/6];$

Note that $\boldsymbol{\rho}$ is $\boldsymbol{\rho} = \mathbf{v} * \theta$

1.2 Exercise 2 - Inverse Equivalent Angle-Axis Problem

Given two reference frames $< a >$ and $< b >$, referred to a common world coordinate system $< w >$, their orientation with respect to the world frame $< w >$ is expressed in Figure 1.

Q2.1 Compute the orientation matrix ${}^a_bR$, by inspection of Figure 1, without using the Rodriguez formula.
Q2.2 Solve the Inverse Equivalent Angle-Axis Problem for the orientation matrix ${}^a_bR$.
Q2.3 Given the following Transformation matrix:

$${}^w_cT = \begin{bmatrix} 0.835959 & -0.283542 & -0.46986 & 0 \\ 0.271321 & 0.957764 & -0.0952472 & -1.23 \\ 0.47703 & -0.0478627 & 0.877583 & 14 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solve the Inverse Equivalent Angle-Axis Problem for the orientation matrix ${}^c_bR$.

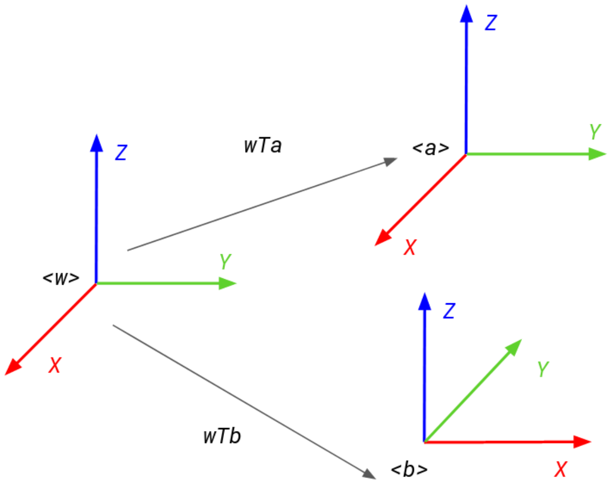


Figure 1: exercise 2 frames

1.3 Exercise 3 - Euler angles (Z-X-Z) vs Tait-Bryan angles (Yaw-Pitch-Roll)

Any orientation matrix can be expressed in terms of three elementary rotations in sequence. These can occur either about the axes of a fixed coordinate system (extrinsic rotations), or about the axes of a rotating coordinate system (intrinsic rotations) initially aligned with the fixed one. Then we can distinguish:

- Proper Euler angles: X-Z-X, Y-Z-Y, ...
- Tait-Bryan angles: Z-Y-X, X-Y-Z, ...

Q3.1 Given two generic frames $< w >$ and $< b >$, define the elementary orientation matrices for frame $< b >$ with respect to frame $< w >$, knowing that:

- $< b >$ is rotated of $45°$ around the z-axis of $< w >$
- $< b >$ is rotated of $60°$ around the y-axis of $< w >$
- $< b >$ is rotated of $-30°$ around the x-axis of $< w >$

Q3.2 Compute the equivalent angle-axis representation for each elementary rotation
Q3.3 Compute the z-y-x (yaw,pitch,roll) representation and solve the Inverse Equivalent Angle-Axis Problem for the obtained orientation matrix
Q3.4 Compute the z-x-z representation and solve the Inverse Equivalent Angle-Axis Problem for the obtained orientation matrix

1.4 Exercise 4 - Quaternions

Given the following quaternion: $q = 0.1647 + 0.31583i + 0.52639j + 0.77204k$ expressing how a reference frame $< b >$ is rotated with respect to $< a >$:
Q4.1 Compute the equivalent rotation matrix, **WITHOUT** using built-in matlab functions.
Q4.2 Solve the Inverse Equivalent Angle-Axis Problem for the obtained orientation matrix

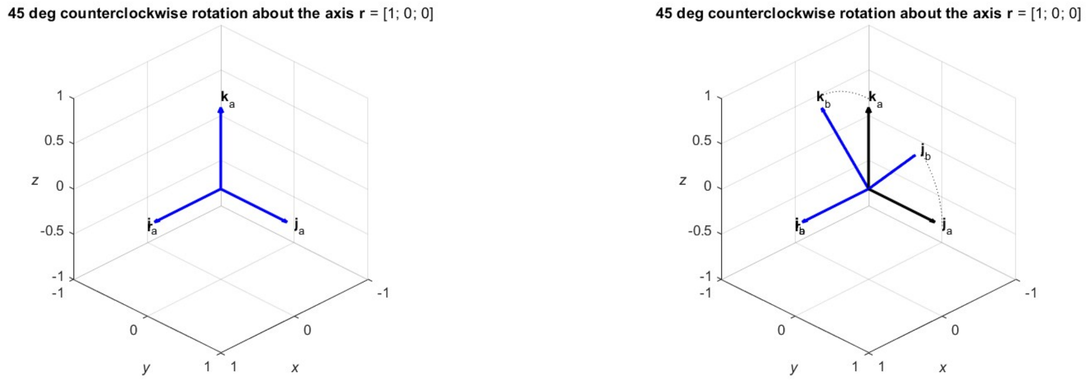
2 Exercise 1

2.1 Q1.1

in order to compute the rotation matrix of the given Angle-Axis representatinos we can use the Rodrigues formula. for that we write the **ComputeAngleAxis()** function.

after using the function we wrote, we get the following results:

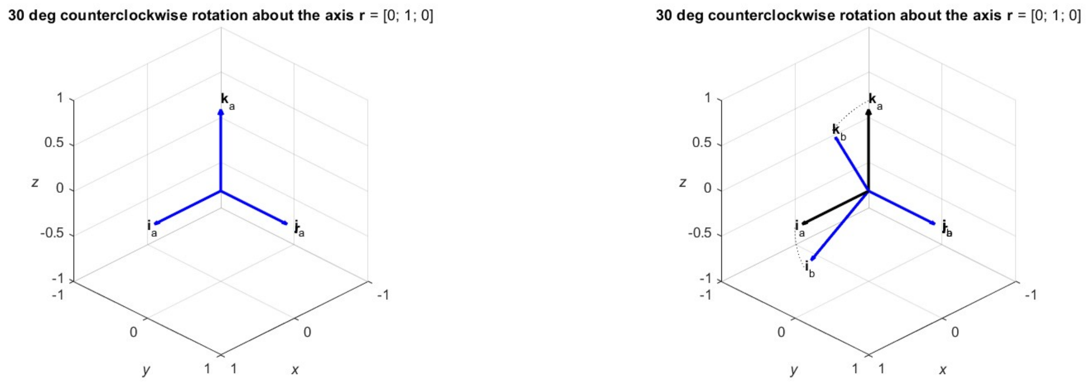
2.2 Q1.2



(a) Initial frame configuration (b) Final frame configuration

Figure 2: Exercise 1.2

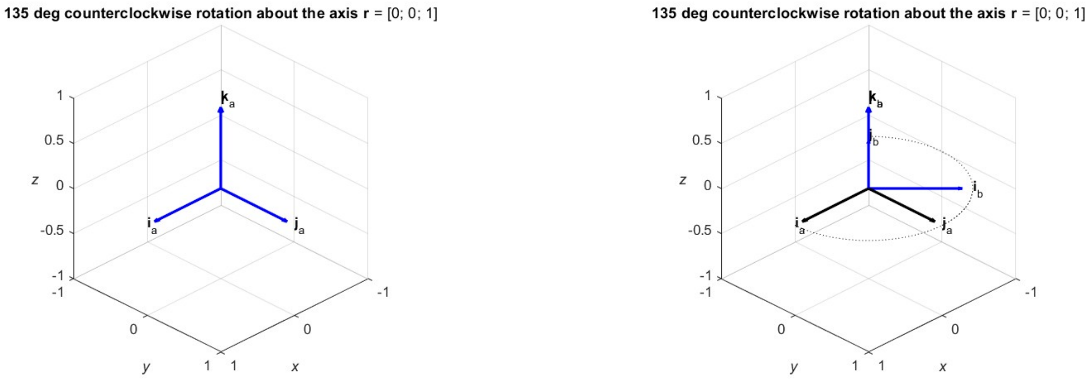
2.3 Q1.3



(a) Initial frame configuration (b) Final frame configuration

Figure 3: Exercise 1.3

2.4 Q1.4



(a) Initial frame configuration

(b) Final frame configuration

Figure 4: Exercise 1.4

2.5 Q1.5

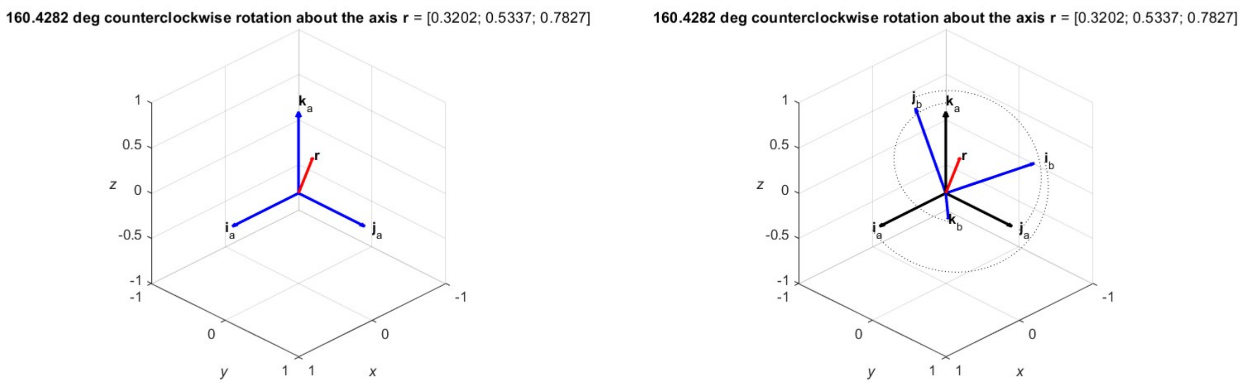
for the last 3 questions we are given

$\rho = v\theta$

for that, we wrote a second function we called **ro()** to be able to get the respected V and θ so we can directly implement our first **ComputeAngleAxis()** function.

$$
\begin{aligned}
let \\
\rho = (a, b, c) \\
v = (x, y, z) \\
so \\
\rho = v\theta \\
\rho = (x\theta, y\theta, z\theta) \\
x^2 + y^2 + z^2 = 1 \\
\theta^2 x^2 + \theta^2 y^2 + \theta^2 z^2 = \theta^2 \\
\sqrt{a^2 + b^2 + c^2} = \theta \\
Thus: \\
\theta = \sqrt{a^2 + b^2 + c^2} \\
v = [\theta a, \theta b, \theta c];
\end{aligned}
\tag{1}
$$

by implementing the following results into the **ro()** code, we get the following results:

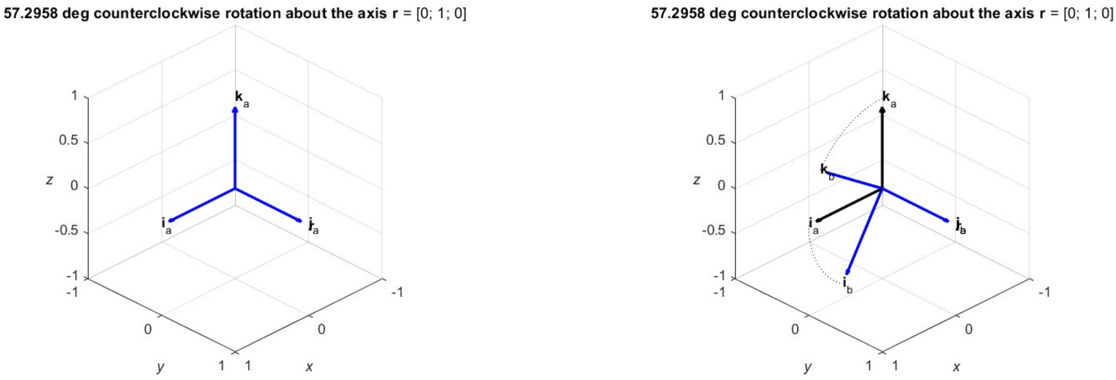


(a) Initial frame configuration

(b) Final frame configuration

Figure 5: Exercise 1.5

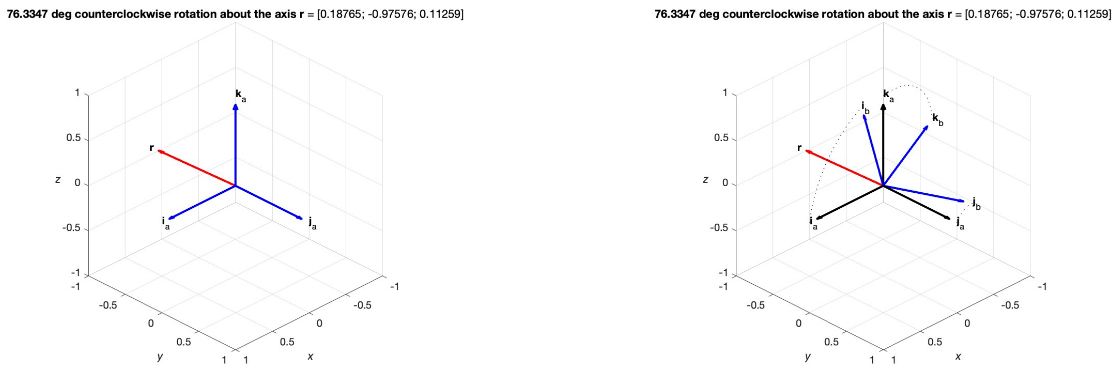
2.6 Q1.6



(a) Initial frame configuration (b) Final frame configuration

Figure 6: Exercise 1.6

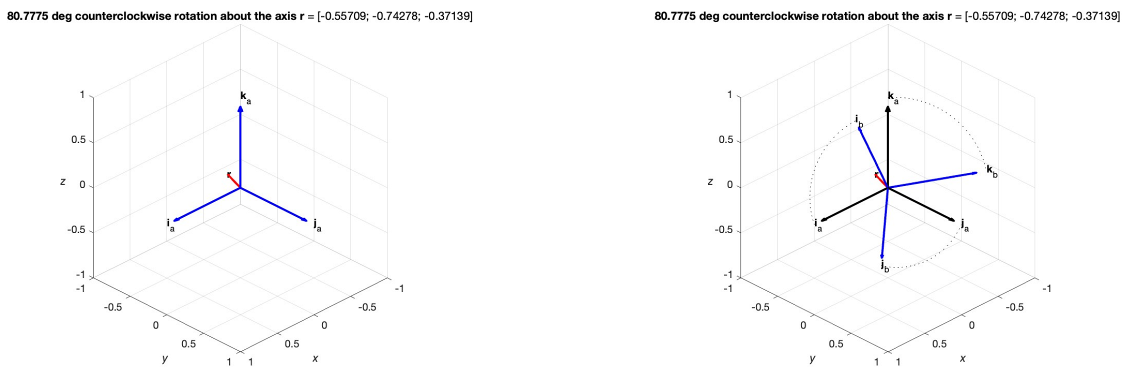
2.7 Q1.7



(a) Initial frame configuration (b) Final frame configuration

Figure 7: Exercise 1.7

2.8 Q1.8



(a) Initial frame configuration (b) Final frame configuration

Figure 8: Exercise 1.8

[Comment] For each exercise include an image of the initial frames configuration, with the applied geometric unit vector and an image of the final configuration

[Comment] For each exercise report the results obtained and provide an explanation of the result obtained (even though it might seem trivial)

3 Exercise 2

3.1 Q2.1

Orientation matrix ${}^a_bR$ can also be described as ${}^a_bR$ = ${}^a_wR\,{}^b_wR^T$. Therefore, we first use the figure 1 to get the two orientation matrices ${}^a_wR$ and ${}^b_wR$. As frame $< a >$ and frame $< w >$ have the same configuration, the rotational matrix ${}^a_wR$ can be described as below.

$${}^a_wR = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In order to get the orientation matrix ${}^b_wR$, we need to compare each axis of frame $< b >$ and frame $< w >$. The x axis of frame $< b >$ is in the direction of y axis of frame $< w >$. And the y axis of frame $< b >$ is in the direction of -x axis of frame $< w >$. So the orientation matrix ${}^b_wR$ can be written as below.

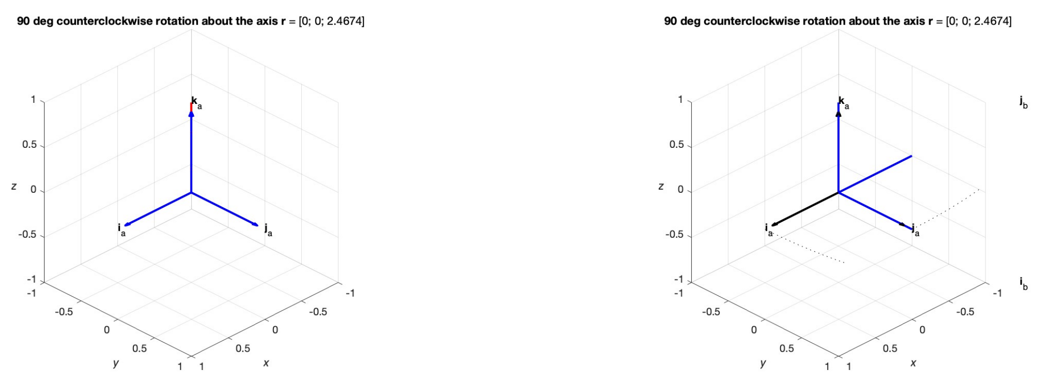
$${}^b_wR = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Using these two orientation matrices, ${}^a_bR$ can be computed.

$${}^a_bR = {}^a_wR\,{}^b_wR^T$$

$${}^a_bR = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The initial frame configuration and the frame of the final configuration is shown below in figure 9



(a) Initial frame configuration of 2.1

(b) Final configuration of 2.1

Figure 9: Exercise 2.1

3.2 Q2.2

In order to solve the inverse equivalent angel-axis problem, we wrote a function called **ComputeInverseAngleAxis()** that gets a rotation matrix R as an input, and calculates the angle of rotation θ and the axis of rotation v.

In this function, we first check if the given matrix is a proper orthogonal matrix. Then, using the formula below, the angle of rotation can be calculated.

$\theta = acos((tr(R) - 1)/2)$

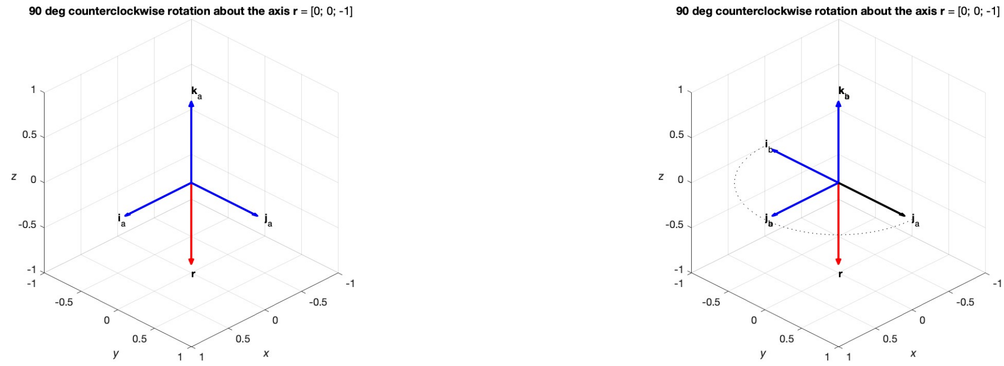
In order to calculate the axis of rotation, we first need to compute the skew-symmetric matrix M of the rotation matrix R using the equation below.

$M = ((R - R')/2)$

Now, using the angle of rotation θ and the skew-symmetric matrix M, we can calculate the axis of rotation v using the formula below. The formula normalizes v by dividing it by $2 * \sin(\theta)$.

$v = [(M(3,2) - M(2,3)), (M(1,3) - M(3,1)), (M(2,1) - M(1,2))]/(2 * sin(theta))$

The initial frame configuration and the frame of the final configuration is shown below in figure 10



(a) Initial frame configuration of 2.2 (b) Final configuration of 2.2

Figure 10: Exercise 2.2

3.3 Q2.3

The transformation matrix ${}^{w}_{c}T$ is given, as below.

$${}^{w}_{c}T = \begin{bmatrix} 0.835959 & -0.283542 & -0.46986 & 0 \\ 0.271321 & 0.957764 & -0.0952472 & -1.23 \\ 0.47703 & -0.0478627 & 0.877583 & 14 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

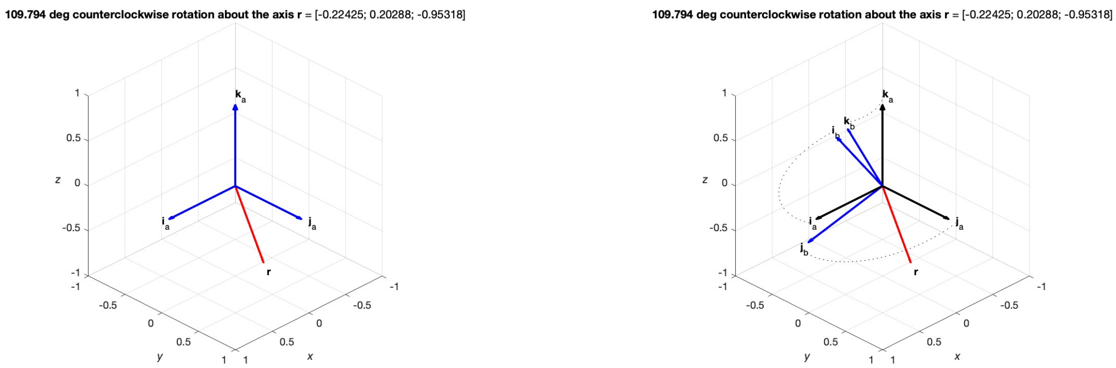
From this transformation matrix, the orientation matrix can be extracted as below.

$${}^{w}_{c}R = \begin{bmatrix} 0.835959 & -0.283542 & -0.46986 \\ 0.271321 & 0.957764 & -0.0952472 \\ 0.47703 & -0.0478627 & 0.877583 \end{bmatrix}$$

Using this matrix and the ones that we have got in exercise 2.1, orientation matrix ${}^{c}_{b}R$ can be solved.

${}^{c}_{b}R = {}^{w}_{c}R^{T}\ {}^{b}_{w}R^{T}$

The orientation matrix ${}^{c}_{b}R$ will then be inputted in the function that we created in exercise 2.2, to solve the inverse equivalent angle-axis. The initial frame configuration and the frame of the final configuration is shown below in figure 11



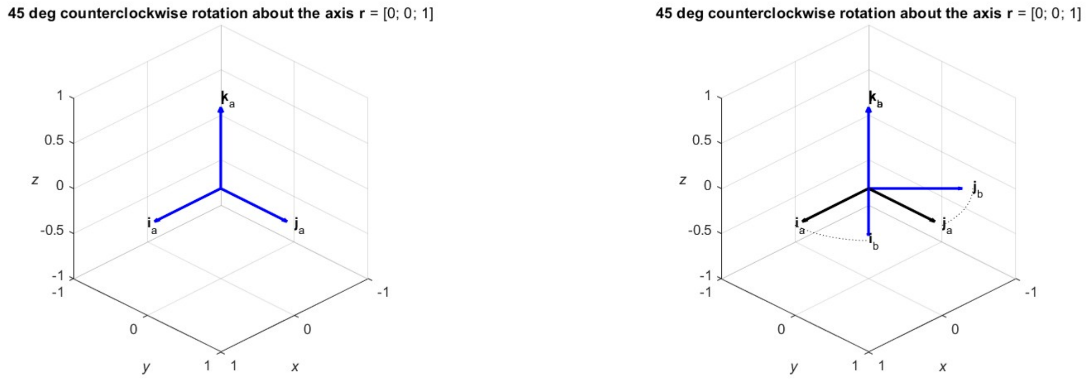
(a) Initial frame configuration of 2.3 (b) Final configuration of 2.3

Figure 11: Exercise 2.3

4 Exercise 3

4.1 Q3.1

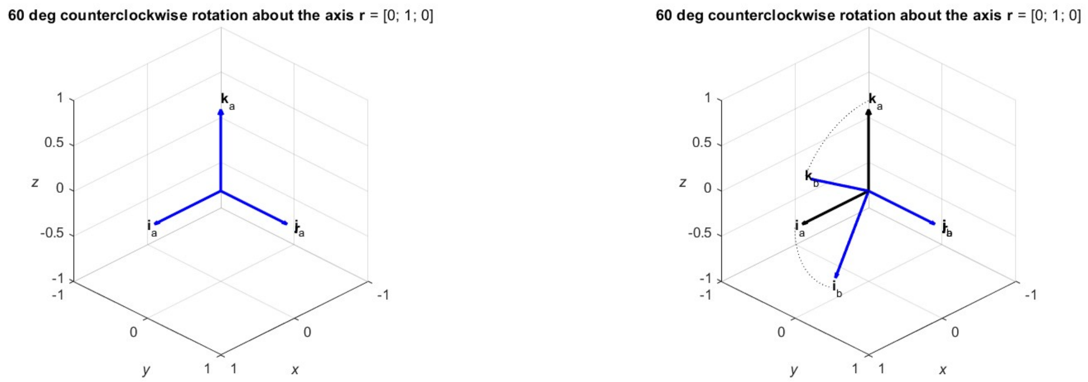
The rotation matrix of $45°$ rotation of frame $< b >$ around the z-axis of $< w >$ is ${}^{w}_{b}R = \begin{bmatrix} \text{cos}(\pi/4) & \text{-sin}(\pi/4) & 0 \\ \text{sin}(\pi/4) & \text{cos}(\pi/4) & 0 \\ 0 & 0 & 1 \end{bmatrix}$



(a) Initial frame configuration (b) Final frame configuration

Figure 12: Exercise 3.1 (a)

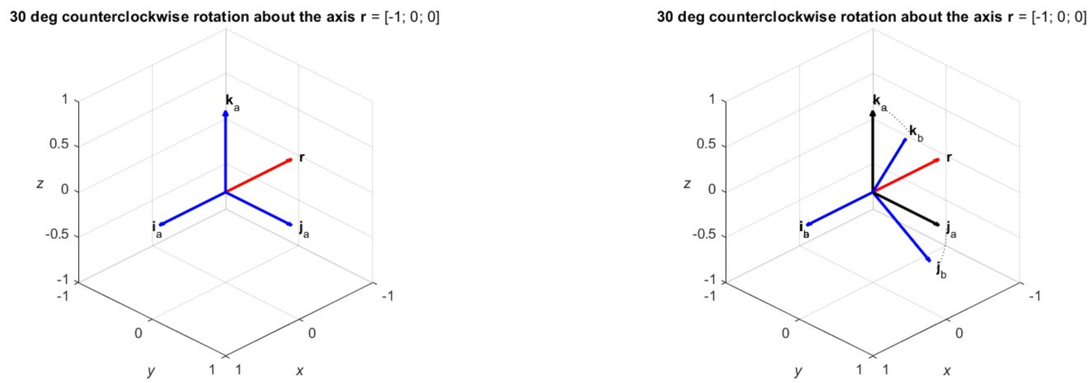
The rotation matrix of $60°$ rotation of frame $< b >$ around the y-axis of $< w >$ is ${}^{w}_{b}R = \begin{bmatrix} \cos(\pi/3) & 0 & \sin(\pi/3) \\ 0 & 1 & 0 \\ -\sin(\pi/3) & 0 & \cos(\pi/3) \end{bmatrix}$



(a) Initial frame configuration (b) Final frame configuration

Figure 13: Exercise 3.1 (b)

The rotation matrix of $-30°$ rotation of frame $< b >$ around the x-axis of $< w >$ is ${}^{w}_{b}R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\pi/6) & -\sin(-\pi/6) \\ 0 & \sin(-\pi/6) & \cos(-\pi/6) \end{bmatrix}$



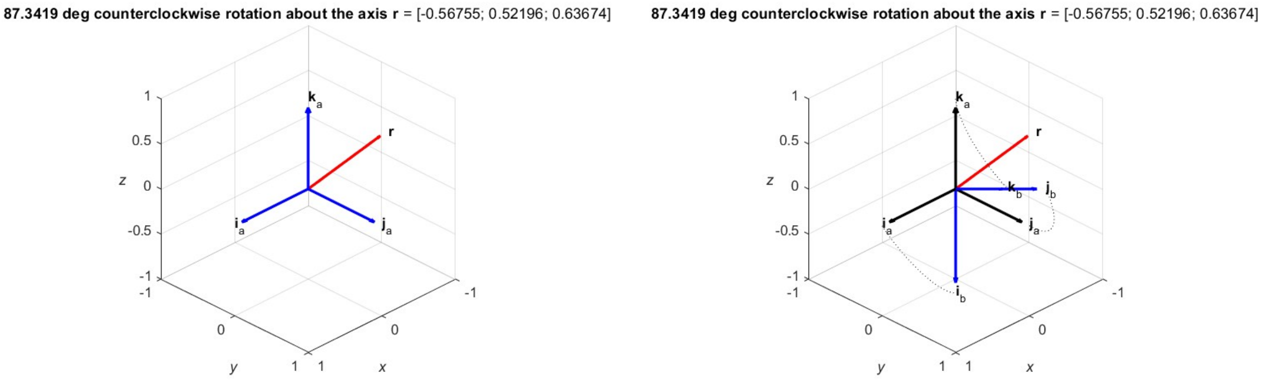
(a) Initial frame configuration (b) Final frame configuration

Figure 14: Exercise 3.1 (c)

4.2 Q3.2

The rotation matrix R_{xyz} is a product of:

$$\begin{bmatrix} \cos(\pi/4) & \text{-}\sin(\pi/4) & 0 \\ \sin(\pi/4) & \cos(\pi/4) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\pi/3) & 0 & \sin(\pi/3) \\ 0 & 1 & 0 \\ \text{-}\sin(\pi/3) & 0 & \cos(\pi/3) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\pi/6) & \text{-}\sin(-\pi/6) \\ 0 & \sin(-\pi/6) & \cos(-\pi/6) \end{bmatrix}$$



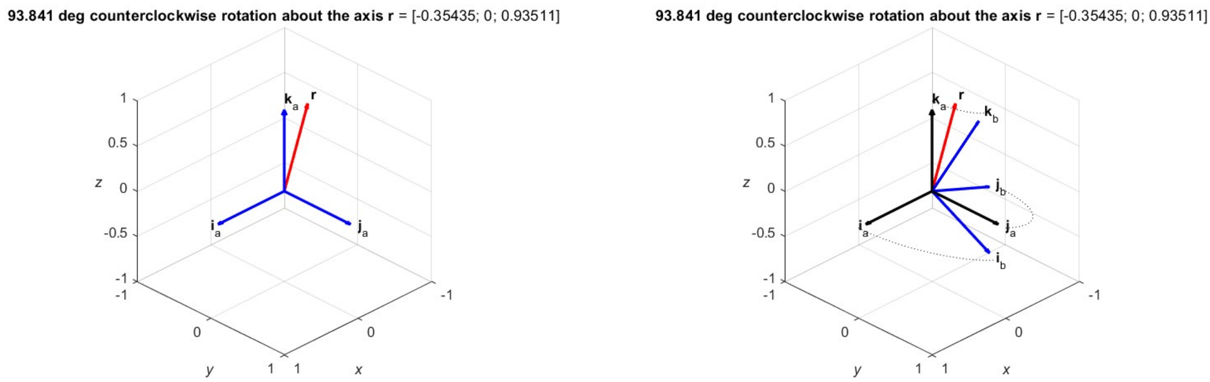
(a) Initial frame configuration (b) Final frame configuration

Figure 15: Exercise 3.2

4.3 Q3.3

The rotation matrix R_{zxz} is a product of:

$$\begin{bmatrix} \cos(\pi/4) & \text{-}\sin(\pi/4) & 0 \\ \sin(\pi/4) & \cos(\pi/4) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\pi/6) & \text{-}\sin(-\pi/6) \\ 0 & \sin(-\pi/6) & \cos(-\pi/6) \end{bmatrix} \begin{bmatrix} \cos(\pi/4) & \text{-}\sin(\pi/4) & 0 \\ \sin(\pi/4) & \cos(\pi/4) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



(a) Initial frame configuration (b) Final frame configuration

Figure 16: Exercise 3.3

5 Exercise 4

first of all we are given:

$$q = 0.1647 + 0.31583i + 0.52639j + 0.77204k$$

Q4.1 Here we are asked to compute the equivalent rotation matrix. to do so we can either use the built-in Matlab function **quat2rotm()** or, we will need to write our own function **quatToRot()** given that according to the book Robotics, Modelling, Planning and Control by Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe

Oriolo, page 55 equation 2.33, we find that the rotation matrix is equal to the following

$$R(\eta, \epsilon) = \begin{bmatrix} 2(\eta^2 + \epsilon_x^2) - 1 & 2(\epsilon_x\epsilon_y - \eta\epsilon_z) & 2(\epsilon_x\epsilon_z + \eta\epsilon_y) \\ 2(\epsilon_x\epsilon_y + \eta\epsilon_z) & 2(\eta^2 + \epsilon_y^2) - 1 & 2(\epsilon_y\epsilon_z - \eta\epsilon_x) \\ 2(\epsilon_z\epsilon_z - \eta\epsilon_y) & 2(\epsilon_y\epsilon_z + \eta\epsilon_x) & 2(\eta^2 + \epsilon_z^2) - 1 \end{bmatrix}$$

after writing the code we get the following result using our function:

```
rot matrix es 4.1
   -0.7463    0.0782    0.6611
    0.5868   -0.3916    0.7088
    0.3143    0.9168    0.2463
```

Figure 17: quat2rotm()'s output

and the following result using the matlab funciton:

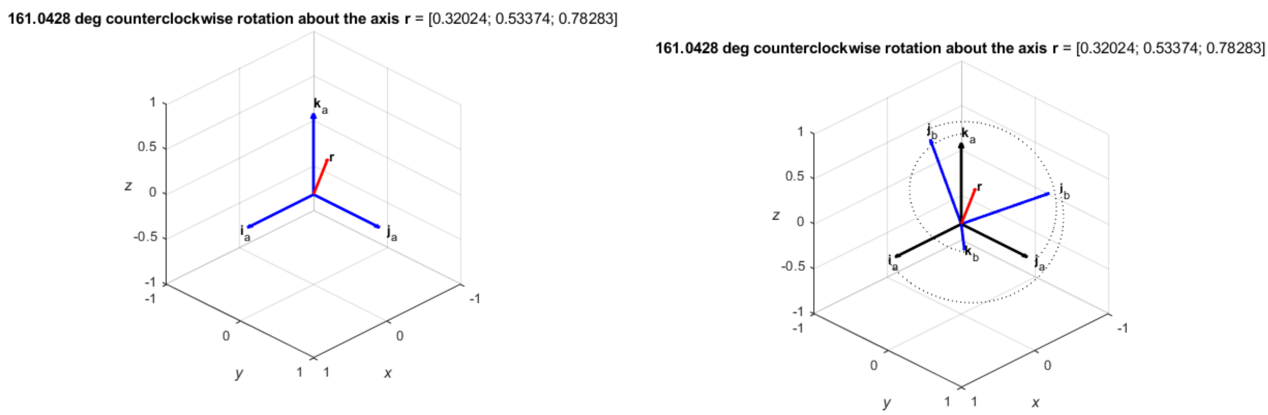
```
rot matrix es 4.1 using matlab function
   -0.7463    0.0782    0.6611
    0.5868   -0.3916    0.7088
    0.3143    0.9168    0.2463
```

Figure 18: quatToRot()'s output

we notice that the results are the same.

Q4.2 in this question we are going to solve the Inverse equivalent Angle-Axis Problem for the orientation matrix. For that, we can use either the **quat2axang()** matlab funciton, or the **ComputeInverseAngleAxis()** function that we wrote.

the ComputeInverseAngleAxis function gives the following result:

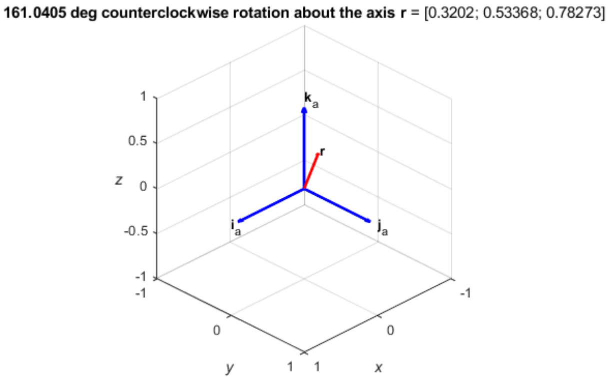


(a) Initial frame configuration of ComputeInverseAngleAxis()'s output

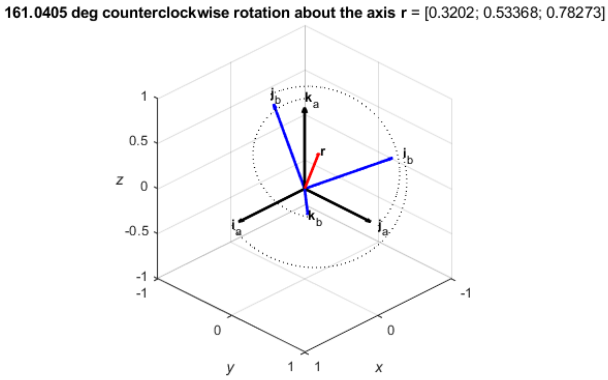
(b) Final configuration of ComputeInverseAngleAxis()'s output

Figure 19: ComputeInverseAngleAxis()'s output

the quat2axang function gives the following result:



(a) Initial frame configuration of quat2axang()'s output



(b) Final configuration of quat2axang()'s output

Figure 20: quat2axang()'s output