UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

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

MODELLING AND CONTROL OF MANIPULATORS

First Assignment

Equivalent representations of orientation matrices

Author:

Khadka Chhetri Rubin

Student ID:

s6558048

Professors:

Giovanni Indiveri
Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Georgii Kurshakov
Luca Tarasi

October 31, 2024

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 and Euler Angles) will be reviewed.

The first assignment is **mandatory** and consists of 5 different exercises. You are asked to:

- Download the .zip file called MCM-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*", "*AngleAxisToRot.m*", "*RotToAngleAxis.m*", "*YPRToRot.m*" and "*RotToYPR.m*".
- Write a report motivating the answers for each exercise, following the predefind format on this document.

1.1 Exercise 1 - Angle-Axis to Rotation Matrix

A particularly interesting minimal representation of 3D rotation matrices is the so-called angle-axis representation, where a rotation is represent by the axis of rotation **h** and the angle θ. Any rotation matrix can be represented by its equivalent angle-axis representation by applying the Rodrigues Formula.

Q1.1 Given an angle-axis pair $(\mathbf{h}, \theta)$, implement on MATLAB the Rodrigues formula, computing the equivalent rotation matrix, **WITHOUT** using built-in matlab functions. The function signature will be

function R = AngleAxisToRot(h,theta)

Then test it for the following cases and briefly comment the results obtained:

- **Q1.2** $\mathbf{h} = [1, 0, 0]^T$ and $\theta = 90^\circ$
- **Q1.3** $\mathbf{h} = [0, 0, 1]^T$ and $\theta = \pi/3$
- **Q1.4** $\rho = [-\pi/3, -\pi/6, \pi/3]$;

Note that $\rho = \mathbf{h}\theta$.

1.2 Exercise 2 - Rotation Matrix to Angle-Axis

Given a rotation matrix R, the problem of finding the corresponding angle-axis representation $(\mathbf{h}, \theta)$ is called the Inverse Equivalent Angle-Axis Problem.

Q2.1 Given a rotation matrix R, implement on MATLAB the Equivalent Angle-Axis equations **WITHOUT** using built-in matlab functions. The function signature will be

function [h,theta] = RotToAngleAxis(R)

You **MUST** check that the input is a valid rotation matrix. Test it for the following cases and briefly comment the results obtained:

- **Q2.2** $R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$
- **Q2.3** $R = \begin{pmatrix} 0.5 & -\sqrt{3}/2 & 0 \\ \sqrt{3}/2 & 0.5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- **Q2.4** $R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- **Q2.5** $R = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

1.3 Exercise 3 - Euler Angles to Rotation Matrix

Any orientation matrix can be expressed in terms of three elementary rotations in sequence. Consider the Yaw Pitch Roll (YPR) representation, where the sequence of the rotation axes is Z-Y-X.

Q3.1 Given a triplet of YPR angles (ψ, θ, ϕ), compute the equivalent rotation matrix representation **WITHOUT** using built-in matlab functions. The function signature will be

function R = YPRToRot(psi, theta, phi)

Then test it for the following cases and briefly comment the results obtained:

- **Q3.2** $\psi = \theta = 0$, $\phi = \pi/2$
- **Q3.3** $\phi = \theta = 0$, $\psi = 60°$
- **Q3.4** $\psi = \pi/3$, $\theta = \pi/2$, $\phi = \pi/4$
- **Q3.5** $\psi = 0$, $\theta = \pi/2$, $\phi = -\pi/12$

1.4 Exercise 4 - Rotation Matrix to Euler Angles

Given a rotation matrix R, it is possible to compute an equivalent triplet of YPR angles (ψ, θ, ϕ), provided that the configuration is not singular (that is, $\cos\theta \neq 0$).

Q4.1 Given a rotation matrix R, implement in MATLAB the equivalent YPR angles, **WITHOUT** using built-in matlab functions. The function signature will be

function [psi,theta,phi] = rotToYPR(R)

You **MUST** check that the input is a valid rotation matrix. Test it for the following cases and briefly comment the results obtained:

- **Q4.2** $R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$
- **Q4.3** $R = \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- **Q4.4** $R = \begin{pmatrix} 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 0.5 & \frac{\sqrt{2}\sqrt{3}}{4} & \frac{\sqrt{2}\sqrt{3}}{4} \\ -\frac{\sqrt{3}}{2} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \end{pmatrix}$

1.5 Exercise 5 - Frame tree

Figure 1 shows the frame tree for the 7 joints of the Franka robot. With reference to the figure, use the geometric definition of the transformation matrix to compute by hand the following matrices.

- **Q5.1** ${}^{0}_{1}T$
- **Q5.2** ${}^{1}_{2}T$
- **Q5.3** ${}^{2}_{3}T$
- **Q5.4** ${}^{3}_{4}T$
- **Q5.5** ${}^{4}_{5}T$
- **Q5.6** ${}^{5}_{6}T$
- **Q5.7** ${}^{6}_{7}T$
- **Q5.8** ${}^{7}_{e}T$

You **MUST** compute the matrices **WITHOUT** using mathematical software.

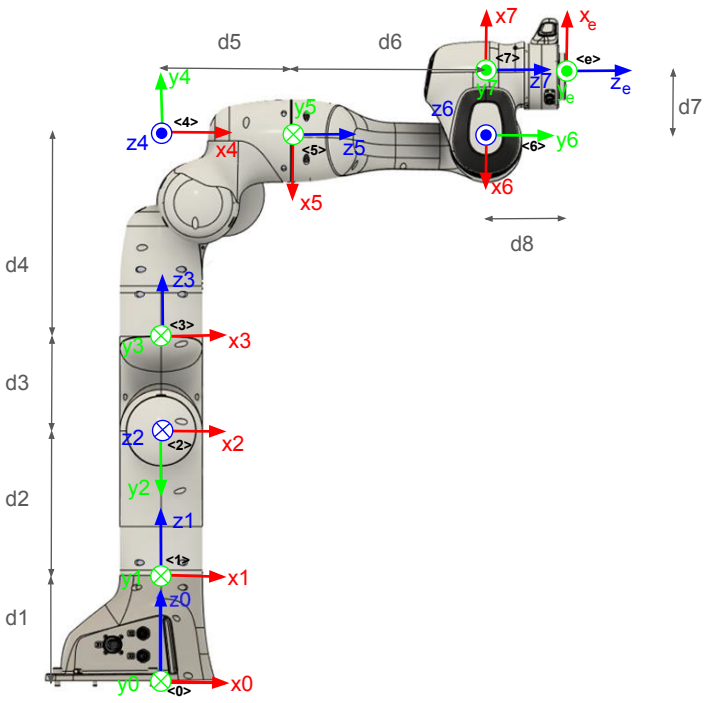

Figure 1: exercise 5 frames

2 Exercise 1 - Angle-Axis to Rotation Matrix

In this exercise, the conversion from angle-axis representation to a rotation matrix is implemented using the Rodrigues formula, expressed as:

$$R(\mathbf{h}, \theta) = I_{3\times3} + [\mathbf{K}] \sin\theta + [\mathbf{K}]^2(1 - \cos\theta) \tag{1}$$

In this formula, R represents the rotation matrix, $I_{3\times3}$ is the identity matrix, θ denotes the angle of rotation, and $[\mathbf{K}]$ is the skew-symmetric matrix derived from the axis of rotation $\mathbf{h}$. This method allows the representation of a rotation in three-dimensional space using a given axis of rotation and angle.

2.1 Q 1.1 - Implementation of Angle-Axis to Rotation Matrix

A MATLAB function `AngleAxisToRot` was implemented to convert an angle-axis representation into a rotation matrix using the Rodrigues formula. The function accepts two inputs: the axis of rotation $\mathbf{h}$ and the angle of rotation θ in radians. The following steps are followed in the function:

1. **Normalization of the Axis Vector:** The input axis vector $\mathbf{h}$ is normalized to ensure it has a unit length. This normalization is crucial because it preserves the direction of the axis while setting its magnitude to 1.

2. **Skew-Symmetric Matrix Construction:** A skew-symmetric matrix $[\mathbf{K}]$ is then constructed from the components of the normalized axis vector. This matrix is used in the Rodrigues formula to compute the rotation matrix:
$$[\mathbf{K}] = \begin{bmatrix} 0 & -h_3 & h_2 \\ h_3 & 0 & -h_1 \\ -h_2 & h_1 & 0 \end{bmatrix} \tag{2}$$

3. **Identity Matrix:** The identity matrix $I_{3\times3}$ (a 3x3 identity matrix) is created using the `eye(3)` function to serve as the foundation for the rotation matrix calculation.

4. **Rotation Matrix Calculation:** Finally, the rotation matrix R is computed using the Rodrigues formula (Equation 1). This combines the identity matrix, the skew-symmetric matrix multiplied by the sine of the rotation angle, and the skew-symmetric matrix squared multiplied by $(1 - \cos(\theta))$. In MATLAB, this calculation is executed as:

```
R = I + sin(theta) * K + (1 - cos(theta)) * (K * K);
```

The complete function code can be found in Appendix B.

2.2 Q 1.2 - Test Case 1

For the test case with axis $\mathbf{h} = [1, 0, 0]^T$ and angle $\theta = 90^\circ$ (or $\theta = \frac{\pi}{2}$ radians), the rotation matrix obtained is as follows:

$$R = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & 0.0000 & -1.0000 \\ 0 & 1.0000 & 0.0000 \end{bmatrix}$$

- The first row $[1.0000, 0, 0]$ confirms that the x-coordinate remains unchanged after the rotation.
- The second row $[0, 0.0000, -1.0000]$ shows that the original y-axis rotates downward to align with the negative z-axis.
- The third row $[0, 1.0000, 0.0000]$ indicates that the original z-axis moves upward to align with the positive y-axis.

This result demonstrates that points initially in the yz-plane are now positioned in the xz-plane with a 90-degree rotation about the x-axis.

2.3 Q 1.3 - Test Case 2

For the test case with axis $\mathbf{h} = [0, 0, 1]^T$ and angle $\theta = \frac{\pi}{3}$ radians, the rotation matrix obtained is as follows:

$$R = \begin{bmatrix} 0.5000 & -0.8660 & 0 \\ 0.8660 & 0.5000 & 0 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

- The third row $[0, 0, 1.0000]$ shows that the z-coordinate remains unchanged.
- The first row $[0.5000, -0.8660, 0]$ shows that a point originally located on the x-axis now lies in the xy-plane, positioned at an angle of 60° from the x-axis.
- The second row $[0.8660, 0.5000, 0]$ indicates that a point initially on the y-axis rotates to align with the positive x-axis.

Thus, points initially in the xy-plane are now at a 60° angle relative to the x-axis with the rotation about the z-axis.

2.4 Q 1.4 - Test Case 3

For the test case with $\rho = [-\frac{\pi}{3}, -\frac{\pi}{6}, \frac{\pi}{3}]$, the rotation matrix obtained is:

$$R = \begin{bmatrix} 0.4444 & -0.4444 & -0.7778 \\ 0.8889 & 0.1111 & 0.4444 \\ -0.1111 & -0.8889 & 0.4444 \end{bmatrix}$$

In this case, the angle θ and axis h were computed from ρ as follows:

$$\mathbf{h} = \frac{\rho}{\|\rho\|}, \quad \theta = \|\rho\| \tag{3}$$

Here, $\|\rho\|$ is the norm of ρ.

- The first row $[0.4444, -0.4444, -0.7778]$ shows how points originally along the x-axis are transformed, indicating a rotation influenced by both the x and y axes.
- The second row $[0.8889, 0.1111, 0.4444]$ suggests that points initially aligned with the y-axis are rotated, primarily influenced by the positive x and z components.
- The third row $[-0.1111, -0.8889, 0.4444]$ reveals how points on the z-axis are affected, indicating a rotation that pushes them towards the negative x-direction while being influenced positively along the y-direction.

3 Exercise 2 - Rotation Matrix to Angle-Axis

In this exercise, the conversion from a rotation matrix to its equivalent angle-axis representation is implemented. Given a rotation matrix R, the angle of rotation θ and the corresponding axis of rotation $\mathbf{h}$ are determined.

The angle-axis representation is derived from the rotation matrix using the following relationships:

$$\theta = \cos^{-1}\left(\frac{\text{tr}(R) - 1}{2}\right) \tag{4}$$

where tr(R) denotes the trace of the matrix R. The axis of rotation $\mathbf{h}$ is calculated using:

$$\mathbf{h} = \frac{1}{\sqrt{1 + R_{11} + R_{22} + R_{33}}} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} \tag{5}$$

This method enables the orientation of a rotation to be expressed as a vector and an angle.

3.1 Q 2.1 - Implementation of Rotation Matrix to Angle-Axis

A MATLAB function, `RotToAngleAxis`, was implemented to convert a rotation matrix R into its angle-axis representation $(\mathbf{h}, \theta)$. The function accept rotation matrix R as input and operates as follows:

1. **Validation of Rotation Matrix:** The input rotation matrix R is first verified as a valid rotation matrix through several checks:
   - *Size Check:* The dimensions of the matrix are confirmed to be 3×3 by verifying both dimensions equal 3. If this condition is not met, an error is raised.
   - *Orthogonality Check:* The orthogonality of R is checked by computing the product RR^T and comparing it to the identity matrix I. This step ensures that the rows (and columns) of R are orthonormal vectors.
   - *Determinant Check:* The determinant of the matrix is calculated, and it is verified to equal 1.
2. **Angle of Rotation:** The angle θ of rotation is then calculated from the trace of the rotation matrix using Equation 4. This angle represents the amount of rotation about the axis $\mathbf{h}$.
3. **Axis Vector Calculation:** The axis of rotation $\mathbf{h}$ is calculated from a skew-symmetric matrix $\mathbf{K}$, derived from the rotation matrix R. The skew-symmetric matrix is calculated using the following formula:

   $$\mathbf{K} = \frac{R - R^T}{2\sin(\theta)} \tag{6}$$

   The corresponding vector is then extracted using the `vex` function, which converts the skew-symmetric matrix $\mathbf{K}$ into a vector that represents the rotation axis. Specifically, the components from the matrix are extracted as follows:

   $$\mathbf{h} = \begin{bmatrix} K_{32} \\ K_{13} \\ K_{21} \end{bmatrix} \tag{7}$$

   where K_{ij} denotes the element in the i-th row and j-th column of the matrix $\mathbf{K}$. This results in a standard 3D vector format that represents the rotation axis.

   The axis vector $\mathbf{h}$ is normalized to ensure it is a unit vector:

   $$\mathbf{h} = \frac{\mathbf{h}}{\|\mathbf{h}\|} \tag{8}$$

   where $\|\mathbf{h}\|$ represents the norm (magnitude) of the vector $\mathbf{h}$.
4. **Output:** Finally, the calculated axis vector $\mathbf{h}$ and the angle θ in radians are returned.

The Matlab function code can be found in Appendix C.

3.2 Q 2.2 - Test Case 1

The rotation matrix used in Test Case 1 is:

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

The output obtained from Test Case 1 is as follows:

$$\text{Rotation axis } (\mathbf{h}) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Angle } (\theta) = 1.5708 \text{ radians (or } 90^\circ)$$

These results indicate that the rotation axis is aligned with the x-axis, and the angle of rotation is 90°. It is confirmed that the rotation represented by the matrix R corresponds to a 90° rotation about the x-axis, which aligns with the properties of the given matrix.

3.3 Q 2.3 - Test Case 2

The rotation matrix used in Test Case 2 is:

$$R = \begin{bmatrix} 0.5 & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The output obtained from Test Case 2 is as follows:

$$\text{Rotation axis } (\mathbf{h}) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{Angle } (\theta) = 1.0472 \text{ radians (or } 60^\circ)$$

These results indicate that the rotation axis is aligned with the z-axis, and the angle of rotation is 60°. It is confirmed that the rotation represented by the matrix R corresponds to a 60° rotation about the z-axis, which aligns with the properties of the given matrix.

3.4 Q 2.4 - Test Case 3

The rotation matrix used in Test Case 3 is:

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

The output obtained from Test Case 3 is as follows:

$$\text{Rotation axis } (\mathbf{h}) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Angle } (\theta) = 0 \text{ radians}$$

These results show that the rotation axis is undefined because the input matrix represents the identity transformation, which corresponds to no rotation. The angle of rotation being 0 confirms that there is no change in orientation, consistent with the properties of the identity matrix.

3.5 Q 2.5 - Test Case 4

The rotation matrix used in Test Case 4 is:

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

The output obtained from Test Case 4 is as follows:

$$\text{Rotation axis } (\mathbf{h}) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Angle } (\theta) = 3.1416 \text{ radians (or } 180^\circ)$$

These results indicate that the rotation axis is a zero vector, which suggests that the transformation represented by the matrix R corresponds to a reflection across the origin rather than a standard rotation about an axis. The angle of rotation is 180°, consistent with the properties of the matrix.

4 Exercise 3 - Euler Angles to Rotation Matrix

In this exercise, the conversion from Yaw-Pitch-Roll (YPR) angles to a rotation matrix is conducted. The rotation matrix is computed using individual rotation matrices for yaw, pitch, and roll, which are sequentially combined to form a 3×3 matrix. This matrix represents the orientation of an object in three-dimensional space based on the specified YPR angles.

4.1 Q 3.1 - Implementation of Yaw-Pitch-Roll to Rotation Matrix

The MATLAB function for this conversion is defined as follows:

```
function R = YPRToRot(psi, theta, phi)
```

This function accepts three inputs: the yaw (ψ), pitch (θ), and roll (ϕ) angles. It then calculates the corresponding rotation matrix R by defining and sequentially combining the individual rotation matrices.

1. **Calculation of Individual Rotation Matrices:**

   The rotation matrices for yaw, pitch, and roll are derived as follows:

   - *Yaw Rotation (Z-axis):*

   $$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

   - *Pitch Rotation (Y-axis):*

   $$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{10}$$

   - *Roll Rotation (X-axis):*

   $$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{11}$$

2. **Combination of Rotation Matrices:**

   The overall rotation matrix R is obtained by combining the rotation matrices in sequence:

   $$R = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi) \tag{12}$$

3. **Output:**

   The rotation matrix R, a 3×3 matrix representing the combined rotation corresponding to the YPR angles, is returned by the function.

The complete Matlab function code is provided in Appendix D.

4.2 Q 3.2 - Test Case 1

The rotation matrix obtained when Yaw $\psi = 0$, Pitch $\theta = 0$, and Roll $\phi = \frac{\pi}{2}$ is:

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

The result indicates a 90° rotation about the X-axis, as determined by the Roll angle $\phi = \frac{\pi}{2}$. This rotation matrix R displays a transformation where the Y and Z axes are rotated, with the Y-axis moved to the Z position and the Z-axis to the negative Y position. This confirms a counterclockwise 90° rotation in the YZ-plane, consistent with a rotation about the X-axis.

4.3 Q 3.3 - Test Case 2

The rotation matrix obtained when Yaw $\psi = 60°$, Pitch $\theta = 0$, and Roll $\phi = 0$ is:

$$R = \begin{bmatrix} 0.5000 & -0.8660 & 0 \\ 0.8660 & 0.5000 & 0 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

This rotation matrix R represents a $60°$ rotation around the Z-axis, consistent with the specified Yaw angle $\psi = 60°$. The X-axis is rotated towards the Y-axis, and the Y-axis towards the X-axis, while the Z-axis remains unchanged. This transformation effectively rotates points in the XY-plane by $60°$.

4.4 Q 3.4 - Test Case 3

The rotation matrix obtained when Yaw $\psi = \frac{\pi}{3}$, Pitch $\theta = \frac{\pi}{2}$, and Roll $\phi = \frac{\pi}{4}$ is:

$$R = \begin{bmatrix} 0.0000 & -0.2588 & 0.9659 \\ 0.0000 & 0.9659 & 0.2588 \\ -1.0000 & 0.0000 & 0.0000 \end{bmatrix}$$

This matrix represents the cumulative effect of a 60° Yaw, a 90° Pitch, and a 45° Roll. The primary alignment of the transformation is along the Z-axis, with modifications to the X and Y axes. This rotation matrix demonstrates a significant tilt due to the Pitch rotation, resulting in an almost vertical position, with the Yaw slightly shifting it along the X-axis.

4.5 Q 3.5 - Test Case 4

The rotation matrix obtained when Yaw $\psi = 0$, Pitch $\theta = \frac{\pi}{2}$, and Roll $\phi = -\frac{\pi}{12}$ is:

$$R = \begin{bmatrix} 0.0000 & -0.2588 & 0.9659 \\ 0 & 0.9659 & 0.2588 \\ -1.0000 & 0.0000 & 0.0000 \end{bmatrix}$$

This matrix reflects the combined effect of the specified angles. The first row indicates minimal movement along the x-axis, with a negative contribution in the y-direction and a strong positive contribution in the z-direction. The second row shows a significant alignment along the y-axis due to the 90° Pitch rotation, while the third row's negative x-axis value demonstrates substantial rotation in 3D space. This configuration captures the influence of the Yaw, Pitch, and Roll rotations on the object's orientation.

5 Exercise 4 - Rotation Matrix to Euler Angles

In this exercise, the conversion from a rotation matrix to equivalent Yaw-Pitch-Roll (YPR) angles is implemented. Yaw, pitch, and roll angles represent rotations around the z, y, and x axes, respectively. The relationship between the rotation matrix R and the YPR angles (ψ, θ, ϕ) is examined.

5.1 Q 4.1 - Implementation of Rotation Matrix to Yaw-Pitch-Roll

The MATLAB function `RotToYPR` is implemented to convert a given rotation matrix into the corresponding Euler angles using the Yaw-Pitch-Roll (YPR) convention. This function produces three outputs: yaw (ψ), pitch (θ), and roll (ϕ) angles.

1. **Input Validation:** The function first verifies that the input R is a valid rotation matrix. Similar to Exercise 2, this involves performing the following checks:
   - *Size Check:* Ensures that R is 3×3 by verifying that both dimensions equal 3. If this condition is not met, an error is raised.
   - *Orthogonality Check:* Confirms that R is orthogonal by computing RR^T and comparing it to the identity matrix I. This ensures that the rows (and columns) of R are orthonormal vectors.
   - *Determinant Check:* Checks that the determinant of R is 1, confirming it as a proper rotation matrix.
2. **Angle Calculation:** After the input validation, the function calculates the Euler angles:
   - The pitch angle θ is calculated as:
   $$\theta = -\arcsin(R_{13}) \tag{13}$$
   where R_{13} is the element in the first row and third column of R.
   - The yaw angle ψ and roll angle ϕ are conditionally calculated based on the cosine of θ to handle potential singularities:
     - If $\cos(\theta) > 1 \times 10^{-6}$ (indicating a non-singular case), yaw and roll are computed as:
     $$\psi = \arctan 2\left(\frac{R_{23}}{\cos(\theta)}, \frac{R_{33}}{\cos(\theta)}\right) \tag{14}$$
     $$\phi = \arctan 2\left(\frac{R_{12}}{\cos(\theta)}, \frac{R_{11}}{\cos(\theta)}\right) \tag{15}$$
     where R_{23} and R_{33} are elements in the second and third rows of R, respectively, for yaw, and R_{12} and R_{11} are elements in the first row for roll.
     - If $\cos(\theta)$ is close to zero (indicating a singularity), yaw is set to 0 (as it becomes indeterminate), and roll is computed as:
     $$\psi = 0 \tag{16}$$
     $$\phi = \arctan 2(R_{21}, R_{22}) \tag{17}$$
     where R_{21} and R_{22} are elements in the rotation matrix. This approach maintains stability even in singular cases.

The Matlab function code of this exercise is added in Appendix E.

5.2 Q 4.2 - Test Case 1

In this test case, the rotation matrix is given as:

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

The calculated Yaw-Pitch-Roll (YPR) angles are:

- Yaw (ψ): -1.5708 radians
- Pitch (θ): 0 radians
- Roll (ϕ): 0 radians

The yaw angle of -1.5708 radians indicates a 90-degree rotation to the left about the z-axis. The pitch and roll angles being zero signify that the object remains level along the y-axis and upright along the x-axis, respectively.

5.3 Q 4.3 - Test Case 2

In this test case, the rotation matrix is given as:

$$R = \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The calculated Yaw-Pitch-Roll (YPR) angles are:

- Yaw (ψ): 0 radians
- Pitch (θ): 0 radians
- Roll (ϕ): -1.0472 radians (which is equivalent to $-\frac{\pi}{3}$)

The results indicate that the yaw and pitch angles are both zero, suggesting no rotation around these axes, while the roll angle of -1.0472 radians represents a rotation of $-\frac{\pi}{3}$ radians about the x-axis.

5.4 Q 4.4 - Test Case 3

In this test case, the rotation matrix is defined as:

$$R = \begin{pmatrix} 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 0.5 & \frac{\sqrt{6}}{4} & \frac{\sqrt{6}}{4} \\ -\frac{\sqrt{3}}{2} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \end{pmatrix}$$

The calculated Yaw-Pitch-Roll (YPR) angles are:

- Yaw (ψ): 1.0472 radians
- Pitch (θ): -0.7854 radians
- Roll (ϕ): -1.5708 radians

The yaw angle of 1.0472 radians indicates a rotation of approximately 60° around the z-axis, suggesting a rightward turn. The pitch angle of -0.7854 radians corresponds to a downward tilt of about -45°, showing a descending rotation around the y-axis. The roll angle of -1.5708 radians represents a clockwise rotation of approximately -90° about the x-axis, indicating a significant tilt.

6 Exercise 5 - Frame Tree

In this exercise, the transformation matrices for the 7 joints of the Franka robot, as shown in Figure 1, are computed. These calculations are performed using the geometric definition of the transformation matrix.

6.1 Q 5.1: - Transformation matrix of frame 1 with respect to frame 0

$$^{0}_{1}T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

6.2 Q 5.2: - Transformation matrix of frame 2 with respect to frame 1

$$^{1}_{2}T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

6.3 Q 5.3: - Transformation matrix of frame 3 with respect to frame 2

$$^{2}_{3}T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

6.4 Q 5.4: - Transformation matrix of frame 4 with respect to frame 3

$$ {}^{3}_{4}T = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

6.5 Q 5.5: - Transformation matrix of frame 5 with respect to frame 4

$$ {}^{4}_{5}T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

6.6 Q 5.6: - Transformation matrix of frame 6 with respect to frame 5

$$ {}^{5}_{6}T = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & d_6 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

6.7 Q 5.7: - Transformation matrix of frame 7 with respect to frame 6

$$ {}^{6}_{7}T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & d_7 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

6.8 Q 5.8: - Transformation matrix of end effector e with respect to frame 7

$$ {}^{7}_{e}T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_8 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

7 Appendix

This section includes the MATLAB function codes and additional support materials.

7.1 Appendix A - Tolerance Checks in Numerical Computations

Tolerance checks are incorporated into numerical algorithms to handle small deviations due to rounding errors, precision limits, or floating-point arithmetic inaccuracies. In this context, a tolerance threshold (e.g., 1×10^{-6}) is used to determine whether a calculated value sufficiently approximates the expected result, rather than requiring an exact match. For example, in rotation matrices, conditions such as orthogonality ($R \cdot R^T = I$) and determinant checks ($\det(R) = 1$) ensure accuracy in representing rotations. However, due to numerical precision, these checks allow slight deviations from their theoretical values, captured by tolerance limits.

7.2 Appendix B - Angle-Axis to Rotation Matrix

```
function R = AngleAxisToRot(h, theta)
    % Normalize the rotation axis
    h = h / norm(h);

    % Skew-symmetric matrix
    K = [0, -h(3), h(2);
         h(3), 0, -h(1);
         -h(2), h(1), 0];

    % Create identity matrix
    I = eye(3);

    % Rodrigues' rotation formula
    R = I + sin(theta) * K + (1 - cos(theta)) * (K * K);
end
```

7.3 Appendix C - Rotation Matrix to Angle-Axis

```
function [h, theta] = RotToAngleAxis(R)
    % Check if R is a 3x3 matrix
    if size(R, 1) ~= 3 || size(R, 2) ~= 3
        error('Input must be a 3x3 matrix.');
    end

    % Check if R is a proper orthogonal matrix
    if abs(det(R) - 1) > 1e-6 || norm(R' * R - eye(3)) > 1e-6
        error('Input matrix is not a valid rotation matrix.');
    end

    % Calculate the angle theta
    theta = acos((trace(R) - 1) / 2);

    % Calculate the rotation axis h
    if theta == 0
        h = [0; 0; 0]; % No rotation
    else
        % Calculate the skew-symmetric part
        S = (R - R') / (2 * sin(theta));
        h = vex(S); % Convert skew-symmetric matrix to vector
    end
end

function a = vex(S_a)
    a = [S_a(3, 2); S_a(1, 3); S_a(2, 1)];
end
```

7.4 Appendix D - Euler Angles to Rotation Matrix

```
function R = YPRToRot(psi, theta, phi)
    % Calculate individual rotation matrices
    Rz = [cos(psi), -sin(psi), 0;
          sin(psi), cos(psi), 0;
          0, 0, 1];

    Ry = [cos(theta), 0, sin(theta);
          0, 1, 0;
          -sin(theta), 0, cos(theta)];

    Rx = [1, 0, 0;
          0, cos(phi), -sin(phi);
          0, sin(phi), cos(phi)];

    % Combine the rotation matrices: R = Rz * Ry * Rx
    R = Rz * Ry * Rx;
end
```

7.5 Appendix E - Rotation Matrix to Euler Angles

```
function [psi, theta, phi] = RotToYPR(R)

    % Check if R is a valid rotation matrix
    if size(R, 1) ~= 3 || size(R, 2) ~= 3
        error('Input must be a 3x3 matrix.');
    end

    % Check for orthogonality
    if norm(R * R' - eye(3)) > 1e-6
        error('Input must be an orthogonal matrix (R * R'' should be the identity matrix).
    end

    % Check for determinant
    if abs(det(R) - 1) > 1e-6
        error('Input must be a proper rotation matrix (det(R) should be 1).');
    end

    % Calculate the angles
    theta = -asin(R(1, 3)); % Pitch (Y-axis)

    if cos(theta) > 1e-6 % Check for singularity
        psi = atan2(R(2, 3) / cos(theta), R(3, 3) / cos(theta)); % Yaw (Z-axis)
        phi = atan2(R(1, 2) / cos(theta), R(1, 1) / cos(theta)); % Roll (X-axis)
    else
        psi = 0; % Indeterminate yaw; can be set to any value
        phi = atan2(R(2, 1), R(2, 2)); % Roll is now determined
    end
end
```