Robotics Assignment

By Joseph Ashton, SID 27047440

UNIVERSITY OF
LINCOLN

Table of Contents

1. Discussion Questions

1a. Types of Robot

Question

1a. What are the main types of robotic systems? What tasks can they perform?

[5 marks]

This is a very open ended question, and so it must be refined before we can attempt to answer it. It can certainly be a useful practice to identify a system of taxonomy of a broad topic to gleam insight into how knowledge on the topic can be structured & organised. Recently having read Ultralearning by Scott Young I'm inclined to relate this to his idea of meta-learning, a practice of deliberately spending time to learn the lay of the land so to speak of a new topic to lay a stronger foundation from which to dive into more directed study. [1]

Taking a "robotic system" to mean a mechanical device that carries out tasks and it's associated peripherals, an example of which could be a 6DOF actuated arm with an end effector along with a power source and digital controller set up to move blocks over a wall. There are endless ways robotic systems could be classified into types, and arrangements capable of any conceivable task. For an engaging and imaginative list of robotic systems and their tasks have a look through this list of Star Wars droids: https://en.wikipedia.org/wiki/Droid_(Star_Wars)#List_of_droid_characters [2]

1b. Types of Joints

Question

1b. What are the types of joints in a robot? What motions can they provide?

[5 marks]

What are the types of joints in a robot? What motions can they provide?

To utilise actuation the parts of a traditional robot must be joined such that they can move by acting on each other but be constrained such that the motion is predictable and controllable.

The following types are broadly applicable and useful for describing almost any conceivable joint.

Joint Type	Motion	Degrees of Freedom
Rotary	Rotation	1
Linear	Translation	1
Cylindrical	Rotation & Translation	2
Spherical	Rotation	3
Planar	Translation	2
Universal	Rotational	2
Screw	Links Rotational to Transnational motion	1
Free	Rotation & Translation	6

However the limits of a Joint are also a vital consideration beyond degrees of freedom. For example a hinge and a wheel would both be "rotary" joints in this regime but their behaviour is not interchangeable given the hinge limits its rotation to 360° or less where a wheel would not. Similarly a cable would be classed as a "free" joint and still would be after it had snapped but the difference in system behaviour would likely be drastic.

MATLAB's Robotic System Toolbox has a class system for defining joints using the `rigidBodyJoint` object that considers both degrees of freedom with it's `type` and position limits aptly named `PositionLimits`. [3]

1c. Forward vs Inverse Kinematics

Question

1c. What are the differences between forward kinematics and inverse kinematics?

[5 marks]

Fundamentally forward kinematics works out final global position and orientation based on a set of system states, and inverse kinematics dose the opposite, figuring out what set of system states will produce a given global position and orientation.

Both are useful techniques for operating robotic systems, where inverse kinematics provides motion planning and forward kinematics can be used for validation and simulation.

Both are applications of kinematic models, and a suitable kinematic model can often pull double duty for use in both forward and inverse kinematics, however forward kinematics is typically much simpler. The forward case can be found as a deterministic result of a series of transformations, where the inverse requires solving/approximating nonlinear equations.

The use of forward kinematics to provide unsupervised learning data for machine learning based control approaches is a developing field in robotics [4] and especially in soft/continuum robotics where the lack of constraints makes kinematic particularly challenging. [5]

1d Types of Sensors

Question

1d. What are the common sensors in a typical robotic system?

[5 marks]

Sensors are vital to control of robotic systems as they provide observability of the system parameters that enable open loop control. In this case "system perimeters" may refer both to the states of controllable system components and to external stimuli.

The most common sensors deployed in robotics systems (based entirely on my own intuition) are likely:

1. **Switches**
   I believe the humble contact switch is almost certainly the most common sensor, not just in their user interface role but as limit switches, shock/bump sensors (yes I'm including reed switches), object detection and more. The simple solution is often the best solution.
2. **Rotary encoders**
   Bar switches I would expect hall effect based rotary encoders are the most common by far, given they provide highly accurate, precise, low latency measures at the point of actuation for electric motors which I would expect to me the most common form of actuator deployed in robotic systems. This is likely followed by other forms of rotary sensor i.e. potentiomiter, pulse, laser etc.
3. **Proximity / Distance**
   From autonomous vehicles to polishing grinding stone wheels, autonomous systems need to measure how close they are to things. Of this category I expect LIDAR/sonic to be most prominent.

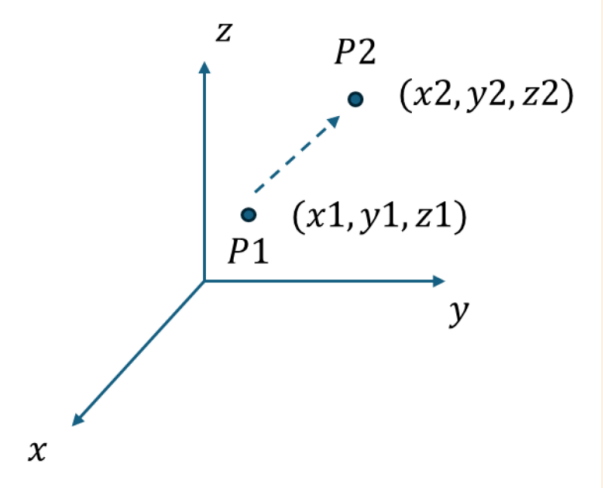
Transformations

This assignment utilises Homogeneous Transformation Matrices (HTM) to perform simple translation and simple rotations. This will be explained in general and applied to specific cases.

Question >

2. Consider the translation P1 to P2

[15 marks]



2a. Determine the homogeneous transformation matrix

Determine the homogeneous transformation matrix (4×4) to translate $P1$ to $P2$ as shown in figure (along x, y, z axes for distance $l(x, y, z)$).
[5 marks]

2b. Find case

$$\text{If} \quad l\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \text{and} \quad P1\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{then calculate} \quad P2\begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

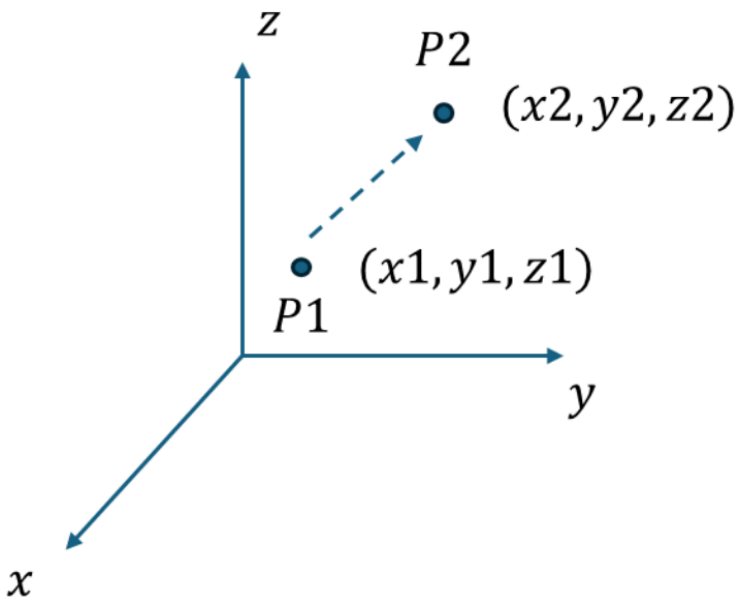
[5 marks]

2c. Do the calculations in MATLAB

Do the calculations in MATLAB and use rigidBody to visualise (both P1 and P2). Show your code (copy and paste here) and result (save as figure the insert here, not screen capture).
[5 marks]

2a. General Transformation Matrix for Simple Translation

The figure shows a simple translation.



This would be represented by the following Homogeneous Transformation Matrix (HTM):

$$\text{Simple Translation HTM:} \quad \begin{bmatrix} 1 & 0 & 0 & x_1 - x_2 \\ 0 & 1 & 0 & y_1 - y_2 \\ 0 & 0 & 1 & z_1 - z_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

That could be applied in the form:

$$P_2 = P_1 \begin{bmatrix} 1 & 0 & 0 & l_x \\ 0 & 1 & 0 & l_y \\ 0 & 0 & 1 & l_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where $l(x, y, z)$ is the vector $\vec{l} = P_2 - P_1$ and P_1 & P_2 are points represented by column vectors of homogeneous coordinates in the form:

$\begin{pmatrix} x/s \\ y/s \\ z/s \\ s \end{pmatrix}$ where $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ are cartesean coordinates denoting displacemnt from origin O.
and s is a scalar typically 1 or the lowest common denominator of (x, y, z).

2b. Specific Case

If the euclidean distance from points P_1 to P_2 is l where:

$$l\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \text{and} \quad P_1\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{then calculate} \quad P_2\begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

And could be be applied as follows to find P_2 :

$$P_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 1 \end{pmatrix}$$

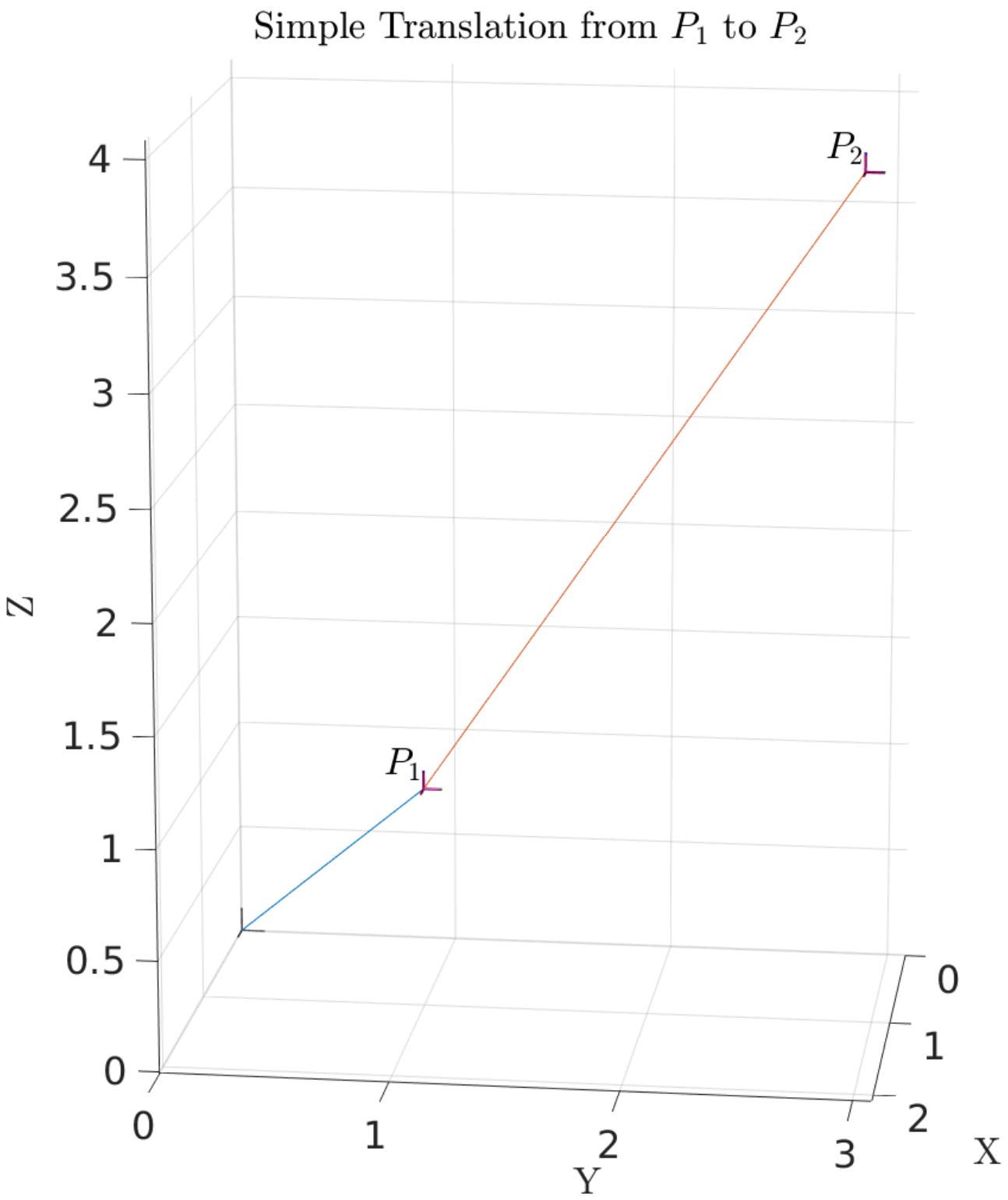
2c. MATLAB Implementation

This simple translation can be used to arrange the joints of of a robot in MATLABs robotics toolbox as demonstrated by the code below.

```
% Define P1 & l
P1 = [1, 1, 1];
l = [1,2,3];

% Generate transformation matricies
T_P1 = trvec2tform(P1);
T_l = trvec2tform(l);

% Create rigidBodyTree
TransformDiagram = rigidBodyTree;
TransformDiagram.BaseName = 'O';

% Add P1
P_1 = rigidBody('$P_{1}$');
D_P1 = rigidBodyJoint('$\vec{P_{1}}');
setFixedTransform(D_P1,T_P1);
P_1.Joint = D_P1;
addBody(TransformDiagram,P_1,'O')

% Add P2
P_2 = rigidBody('$P_{2}$');
D_P2 = rigidBodyJoint('$\vec{P_{2}}');
setFixedTransform(D_P2,T_l);
P_2.Joint = D_P2;
addBody(TransformDiagram,P_2,'$P_{1}$')

% Visualize using rigidBody
show(TransformDiagram);
axis equal;

% lables
P2=P1+l;
text(P1(1), P1(2), P1(3), '$P_1$', 'VerticalAlignment', 'bottom', 'HorizontalAlignment',
'right');
text(P2(1), P2(2), P2(3), '$P_2$', 'VerticalAlignment', 'bottom', 'HorizontalAlignment',
'right');
set(groot,'defaultTextInterpreter','latex'); % nicer lables
fontsize(24,"points"); % bigger lables

title('Simple Translation from $P_{1}$ to $P_{2}$');
view([2, 0.25, 0.5]); % side on view
```

Simple Translation Figure

This script generates the following figure.

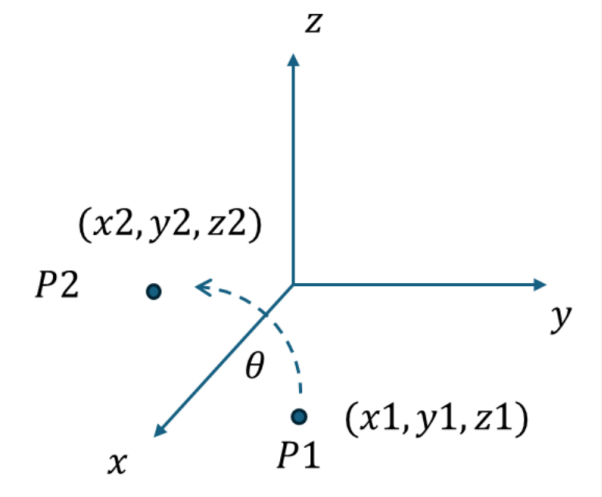


⊗ 1 Error in region

Question >

3. Consider the rotation about x of P1 to P2

[15 marks]



3a. Determine the homogeneous transformation matrix

Determine the homogeneous transformation matrix (4×4) to translate $P1$ to $P2$ around axis x for angle θ as shown in the figure above.
[5 marks]

3b. Find case

$$\text{If} \quad \theta = 30^\circ \quad \text{and} \quad P1 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{then calculate} \quad P2 \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

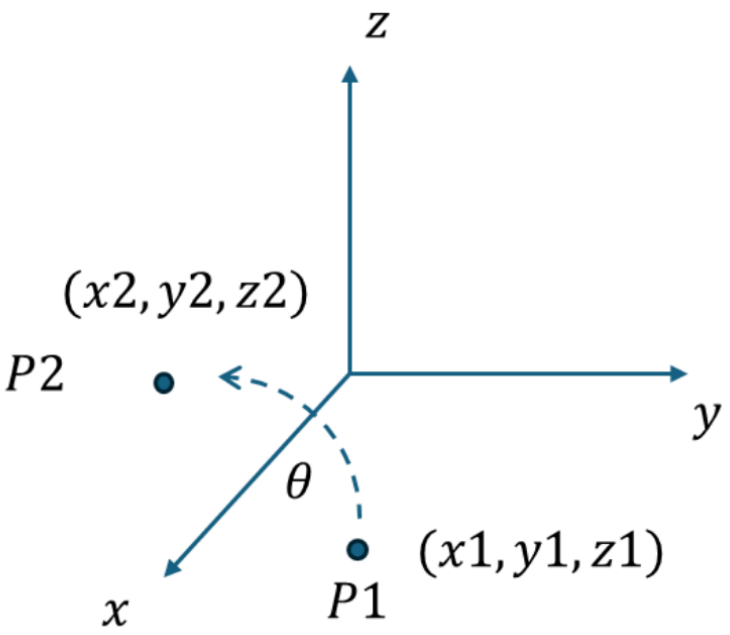
[5 marks]

3c. Do the calculations in MATLAB

Do the calculations in MATLAB and use rigidBody to visualise (both P1 and P2). Show your code (copy and paste here) and result (save as figure the insert here, not screen capture).
[5 marks]

3a. General Transformation Matrix for Simple Rotation

The figure shows a simple rotation.



The form of the Homogeneous Transformation Matrix (HTM) for a simple rotation about the axis depends on the axis about which the rotation takes place as seen in the equation below for the typical counter clockwise forms.

$$\text{HTM for Simple Rotation of Angle } \theta$$

$$\underset{\text{about } x}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \quad \underset{\text{about } y}{\begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \quad \underset{\text{about } z}{\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

As the case above is about the x axis the transformation can be applied as follows:

$$P_2 = P_1 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where θ is the angle of counter clockwise rotation about the x axis and P_1 & P_2 are points represented by column vectors of homogeneous coordinates in the form as previously discussed.

3a. Specific Case

The case stated in the brief is ambiguous about whether the rotation is to be take about the origin O or point $P1$ and as to the direction of the rotation, however it is taken to mean;

> *If P_1 is a point at $(1, 1, 1)$ and point P_2 is found 30° counter clockwise about the origin x axis from P_1 find the coordinates for P_2.*

The general form described above the transformation matrix could be applied as follows to find P_2 as follows:

$$P_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(30) & \sin(30) & 0 \\ 0 & -\sin(30) & \cos(30) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \approx \begin{pmatrix} 1 \\ -0.83 \\ 1.14 \\ 1 \end{pmatrix}$$

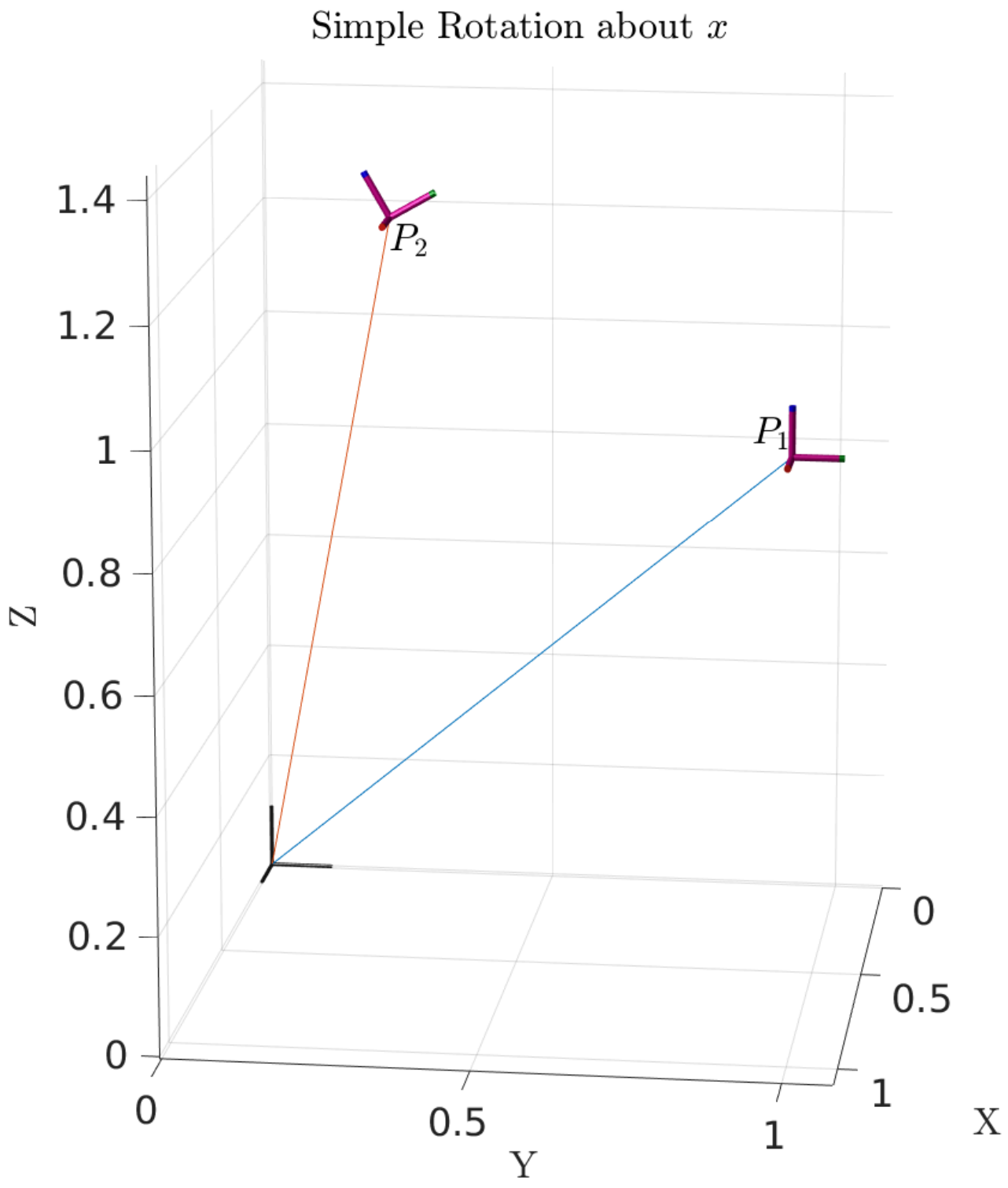
3c. MATLAB Implementation

This simple rotation can be used to arrange the joints of of a robot in MATLABs robotics toolbox as demonstrated by the code below.

```
% Define P1 & theta
P1 = [1, 1, 1];
theta_x = 30;
theta_y = 0;
theta_z = 0;

% Generate transformation matricies
T_P1 = trvec2tform(P1);
eul = deg2rad([theta_z,theta_y,theta_x]);
P2 = (eul2rotm(eul)*P1')';
T_P2 = se3(eul,"eul",P2);

% Create rigidBodyTree
TransformDiagram = rigidBodyTree;
TransformDiagram.BaseName = 'O';

% Add P1
P_1 = rigidBody('$P_{1}$');
D_P1 = rigidBodyJoint('$\vec{P_{1}}');
setFixedTransform(D_P1,T_P1);
P_1.Joint = D_P1;
addBody(TransformDiagram,P_1,'O')

% Add P2
P_2 = rigidBody('$P_{2}$');
D_P2 = rigidBodyJoint('$\vec{P_{2}}');
setFixedTransform(D_P2,T_P2);
P_2.Joint = D_P2;
addBody(TransformDiagram,P_2,'O')

% Visualize using rigidBody
show(TransformDiagram);
axis equal;

% lables
P2 = tform2trvec(P_2.Joint.JointToParentTransform);
text(P1(1), P1(2), P1(3), '$P_1$', 'VerticalAlignment', 'bottom', 'HorizontalAlignment',
'right');
text(P2(1), P2(2), P2(3), '$P_2$', 'VerticalAlignment', 'top', 'HorizontalAlignment', 'left');
title('Simple Rotation about $x$');
set(groot,'defaultTextInterpreter','latex'); % nicer lables
fontsize(24,"points"); % bigger lables

view([2, 0.25, 0.5]); % side on view
```

Simple Rotation about x Figure

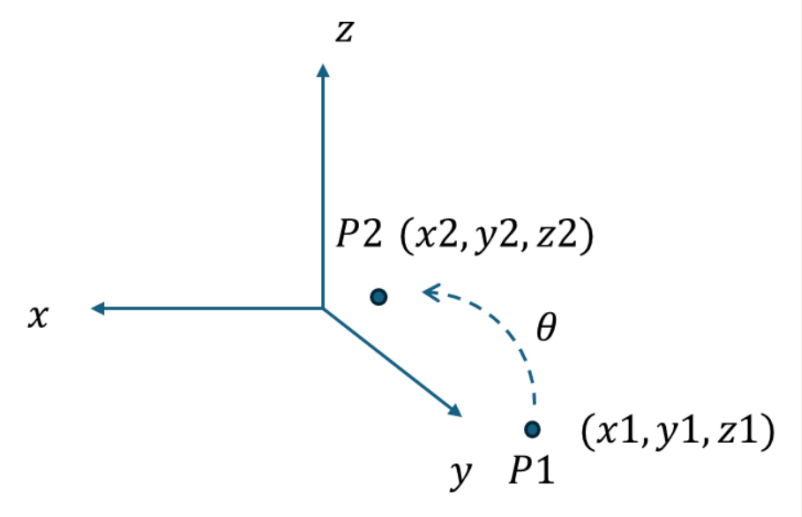
This script generates the following figure.



Question >

4. Consider the rotation about y of P1 to P2

[15 marks]



4a. Determine the homogeneous transformation matrix

Determine the homogeneous transformation matrix (4×4) to translate $P1$ to $P2$ around axis y for angle θ as shown in the figure above.
[5 marks]

4b. Find case

$$\text{If} \quad \theta = 45^\circ \quad \text{and} \quad P1 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{then calculate} \quad P2 \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

[5 marks]

3c. Do the calculations in MATLAB

Do the calculations in MATLAB and use rigidBody to visualise (both $P1$ and $P2$). Show your code (copy and paste here) and result (save as figure the insert here, not screen capture).
[5 marks]

4a. Transformation Matrix for Simple Rotation about y

The figure below (left) shows a simple rotation about y. As the case above is about the y axis the transformation can be applied as shown in the equation below (right). Where as before θ is the angle of rotation and P_1 & P_2 are points represented by column vectors of homogeneous coordinates in the form.

4b. Specific Case

The case stated in the brief once again is ambiguous about whether the rotation is to be take about the origin O or point $P1$ and as to the direction of the rotation, however for the sake of variety it is taken to mean;

> *If P_1 is a point at $(1, 1, 1)$ and point P_2 is found 45° **clockwise** about the origin y axis from P_1 find the coordinates for P_2.*

The general form described above for counter clockwise (CCW) rotations can be adapted to the clockwise (CW) form simply by swapping the signs of the sin terms.

$$\text{HTM for Simple Rotation of Angle } \theta \text{ about } y$$

Counter Clockwise	Clowckwise
$\begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Thus the transformation matrix could be applied to find P_2 as follows:

$$P_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{bmatrix} \cos(45) & 0 & \sin(45) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(45) & 0 & \cos(45) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{pmatrix} \sqrt{2} \\ 1 \\ 0 \\ 1 \end{pmatrix} \approx \begin{pmatrix} 1.41 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

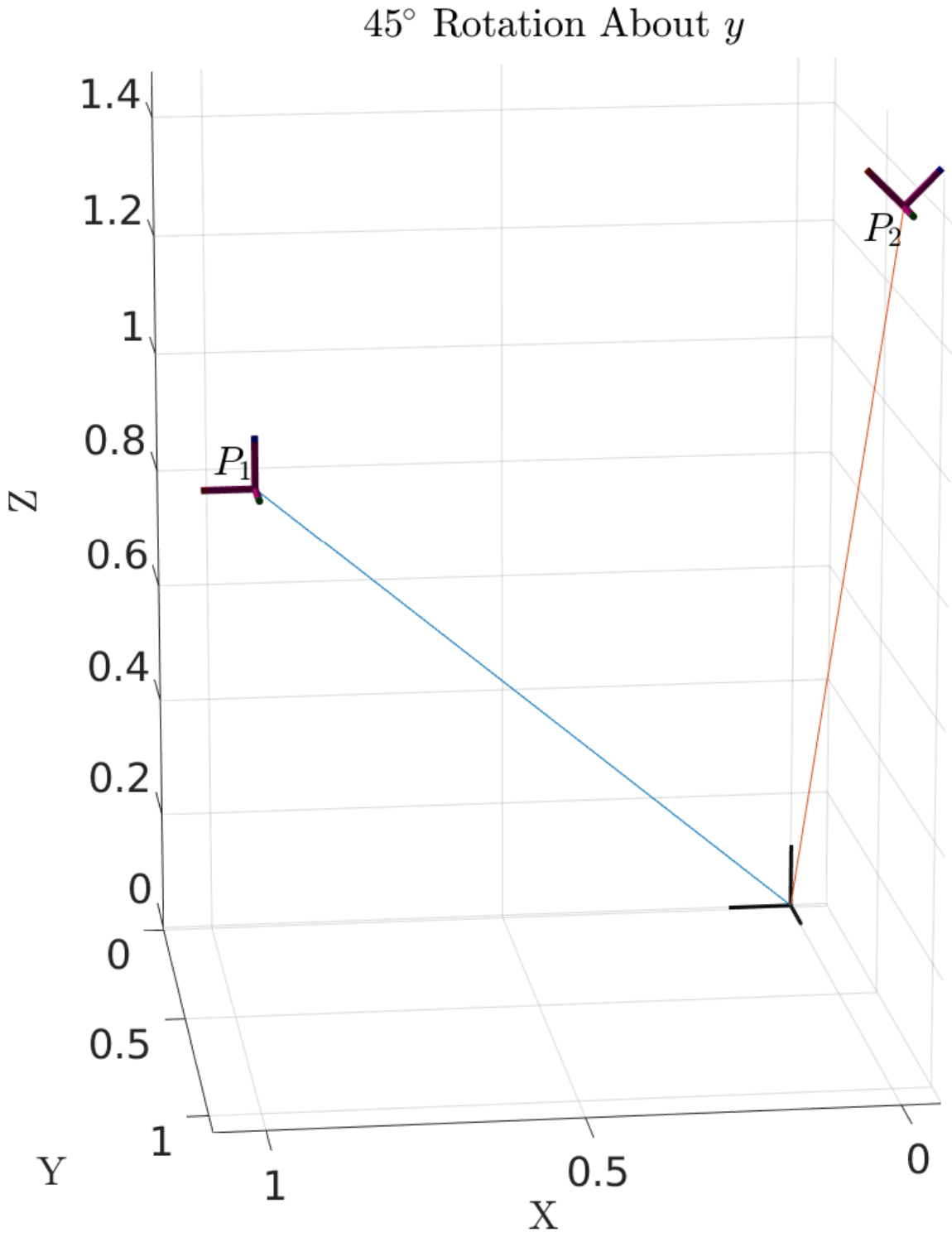
4c. MATLAB Implementation

This simple rotation can be used to arrange the joints of of a robot in MATLABs robotics toolbox as demonstrated by the code below.

```
% Define P1 & theta
P1 = [1, 1, 1];
theta_x = 0;
theta_y = 45;
theta_z = 0;

% Generate transformation matricies
T_P1 = trvec2tform(P1);
eul = deg2rad([theta_z,theta_y,theta_x]*-1);
P2 = (eul2rotm(eul)*P1')';
T_P2 = se3(eul,"eul",P2);

% Create rigidBodyTree
TransformDiagram = rigidBodyTree;
TransformDiagram.BaseName = 'O';

% Add P1
P_1 = rigidBody('$P_{1}$');
D_P1 = rigidBodyJoint('$\vec{P_{1}}');
setFixedTransform(D_P1,T_P1);
P_1.Joint = D_P1;
addBody(TransformDiagram,P_1,'O')

% Add P2
P_2 = rigidBody('$P_{2}$');
D_P2 = rigidBodyJoint('$\vec{P_{2}}');
setFixedTransform(D_P2,T_P2);
P_2.Joint = D_P2;
addBody(TransformDiagram,P_2,'O')

% Visualize using rigidBody
show(TransformDiagram);
view([0.25, 2, 0.5]); % side on view
axis equal;

% lables
P2 = tform2trvec(P_2.Joint.JointToParentTransform);
text(P1(1), P1(2), P1(3), '$P_1$', 'VerticalAlignment', 'bottom', 'HorizontalAlignment',
'right');
text(P2(1), P2(2), P2(3), '$P_2$', 'VerticalAlignment', 'top', 'HorizontalAlignment', 'right');
title('$45^{\circ}$ CW Rotation About $y$');
set(groot,'defaultTextInterpreter','latex'); % nicer lables
fontsize(24,"points"); % bigger lables
```

Simple Clockwise rotation about y figure

This script generates the following figure.

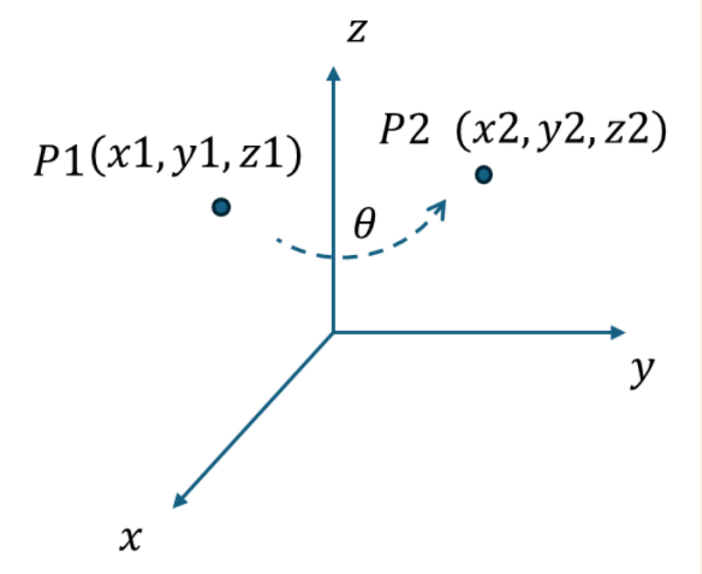


⊗ 1 Error in region

Question >

5. Consider the rotation about z of P1 to P2

[15 marks]



5a. Determine the homogeneous transformation matrix

Determine the homogeneous transformation matrix (4×4) to translate $P1$ to $P2$ around axis y for angle θ as shown in the figure above.
[5 marks]

5b. Specific Case

$$\text{If} \quad \theta = 60^\circ \quad \text{and} \quad P1 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{then calculate} \quad P2 \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

[5 marks]

5c. Do the calculations in MATLAB

Do the calculations in MATLAB and use rigidBody to visualise (both $P1$ and $P2$). Show your code (copy and paste here) and result (save as figure the insert here, not screen capture).
[5 marks]

5a. Transformation Matrix for Simple Rotation about z

The figure below (left) shows a simple rotation about z. As the case above is about the z axis the transformation can be applied as shown in the equation below (right). Where as before θ is the angle of CCW rotation and P_1 & P_2 are points represented by column vectors of homogeneous coordinates in the form.

5b. Specific Case

The case stated in the brief once again is ambiguous about whether the rotation is to be take about the origin O or point $P1$ and as to the direction of the rotation, however it is taken to mean;

> *If P_1 is a point at $(1, 1, 1)$ and point P_2 is found 60° counter clockwise about the origin x axis from P_1, find the coordinates for P_2.*

Such a transformation can be applied with the HTM equation mentioned above with the 60° angle substituted in for θ to find P_2 as shown below:

$$P_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{bmatrix} \cos(60) & -\sin(60) & 0 & 0 \\ \sin(60) & \cos(60) & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \approx \begin{pmatrix} -0.33 \\ 1 \\ 1.38 \\ 1 \end{pmatrix}$$

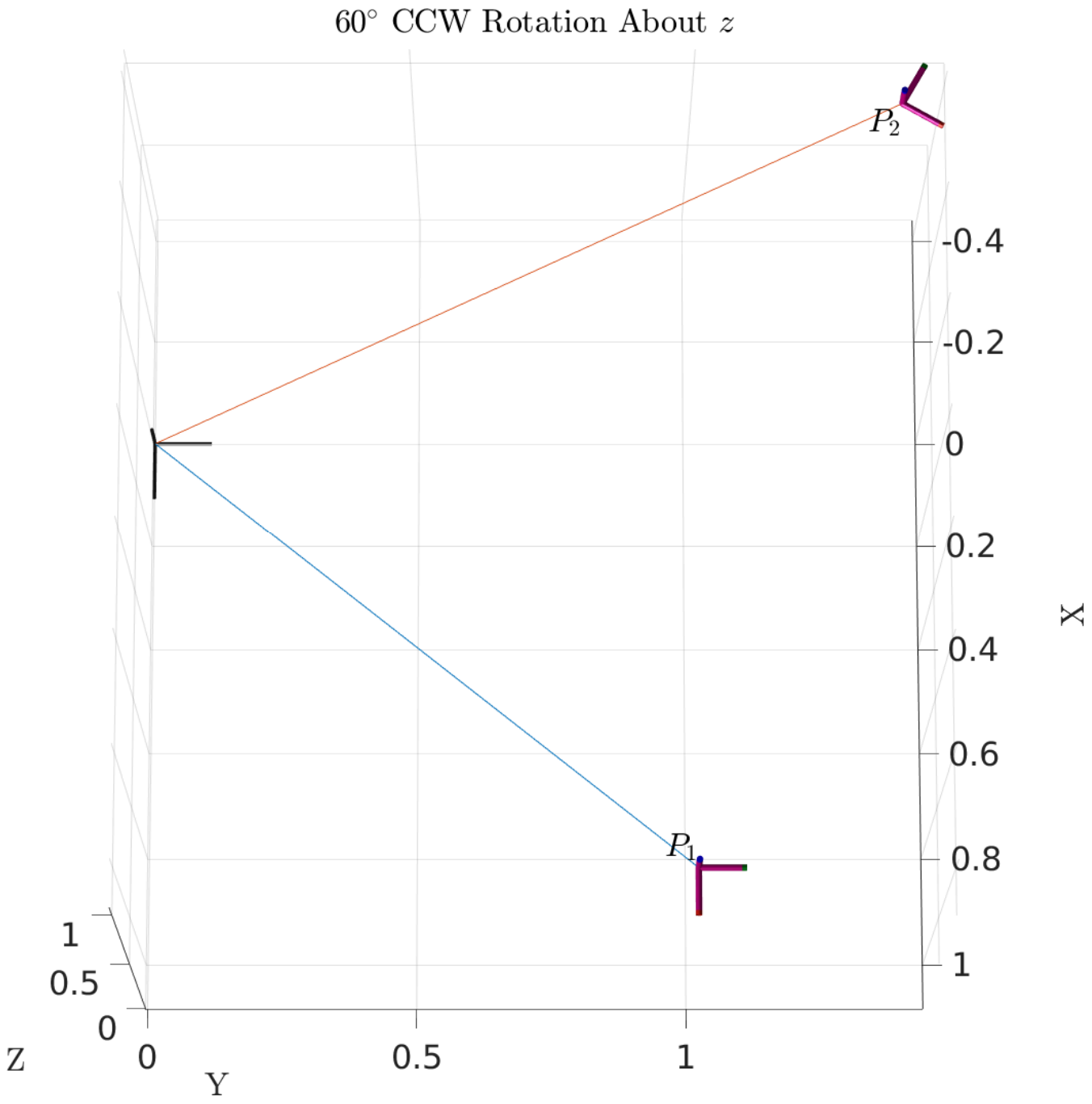
5c. MATLAB Implementation

This simple rotation can be used to arrange the joints of of a robot in MATLABs robotics toolbox as demonstrated by the code below.

```
% Define P1 & theta
P1 = [1, 1, 1];
theta_x = 0;
theta_y = 0;
theta_z = 60;

% Generate transformation matricies
T_P1 = trvec2tform(P1);
eul = deg2rad([theta_z,theta_y,theta_x]);
P2 = (eul2rotm(eul)*P1')';
T_P2 = se3(eul,"eul",P2);

% Create rigidBodyTree
TransformDiagram = rigidBodyTree;
TransformDiagram.BaseName = 'O';

% Add P1
P_1 = rigidBody('$P_{1}$');
D_P1 = rigidBodyJoint('$\vec{P_{1}}');
setFixedTransform(D_P1,T_P1);
P_1.Joint = D_P1;
addBody(TransformDiagram,P_1,'O')

% Add P2
P_2 = rigidBody('$P_{2}$');
D_P2 = rigidBodyJoint('$\vec{P_{2}}');
setFixedTransform(D_P2,T_P2);
P_2.Joint = D_P2;
addBody(TransformDiagram,P_2,'O')

% Visualize using rigidBody
show(TransformDiagram);
view([2,0,8.5]); % side on view
axis tight;

% lables
P2 = tform2trvec(P_2.Joint.JointToParentTransform);
text(P1(1), P1(2), P1(3), '$P_1$', 'VerticalAlignment', 'bottom', 'HorizontalAlignment',
'right');
text(P2(1), P2(2), P2(3), '$P_2$', 'VerticalAlignment', 'top', 'HorizontalAlignment', 'right');
title('$60^{\circ}$ CCW Rotation About $z$');
set(groot,'defaultTextInterpreter','latex'); % nicer lables
fontsize(24,"points"); % bigger lables
```

Simple z rotation figure

This script generates the following figure.



⊗ 1 Error in region

6. Inverse Kinematics

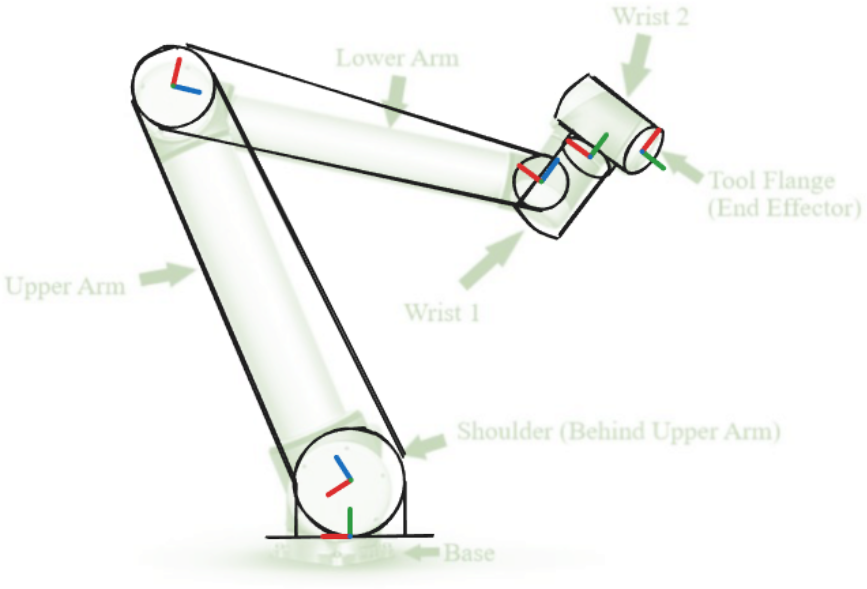
6a. Diagram

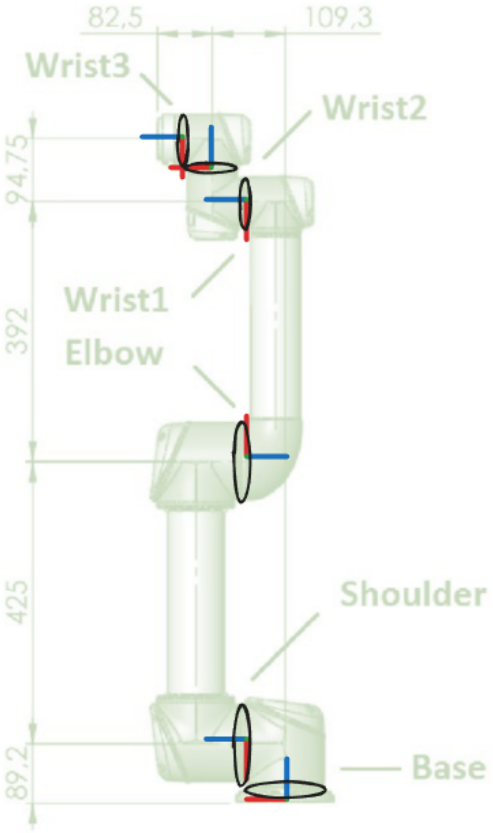
Question

6a. Assign appropriate frames to the given robot (on the given figure or your own sketch).

[5 marks]

It is sufficient to show only 2 axes for each frame since the third axis can be deduced. Briefly comment on how you assigned the frames.





6b. MatLab Model

Question

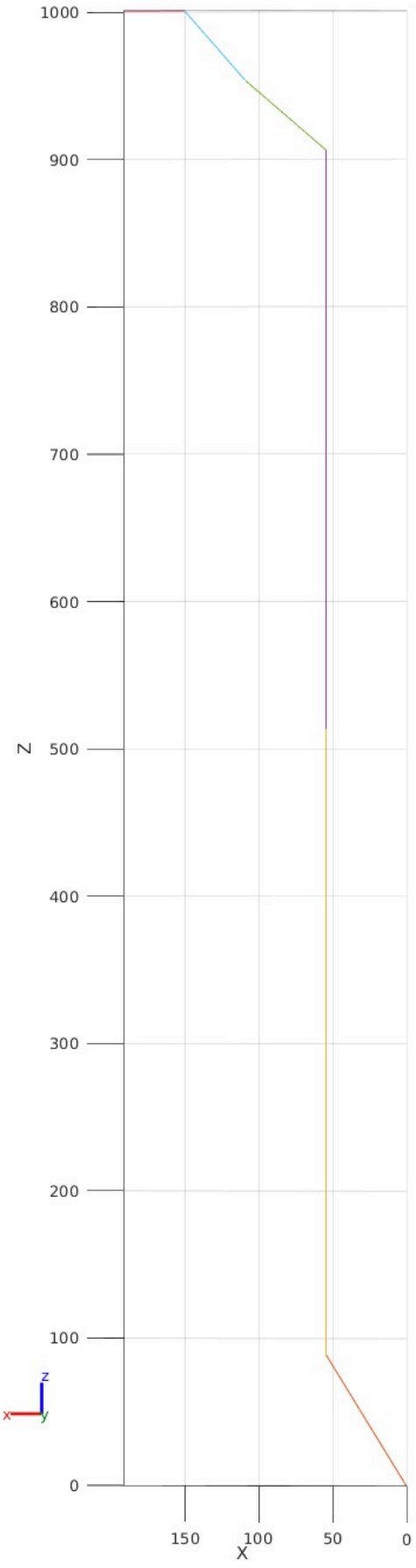
6b. Now that you have assigned appropriate frames, model this robot in MATLAB, show your code (copy and paste here) and result (save as figure the insert here, not screen capture).

[10 marks]

```
robotArm = rigidBodyTree;

% Actual Base to Base Joint
base = rigidBody('Base');
joint_base = rigidBodyJoint('joint_base', 'revolute');
setFixedTransform(joint_base, trvec2tform([0, 0, 0]));
joint_base.JointAxis = [0 0 1]; % Rotation about Z0
base.Joint = joint_base;
addBody(robotArm, base, robotArm.BaseName);

% Base to Shoulder
shoulder = rigidBody('Shoulder');
joint_shoulder = rigidBodyJoint('joint_shoulder', 'revolute');
setFixedTransform(joint_shoulder, trvec2tform([54.65, 0, 89.2]));
joint_shoulder.JointAxis = [0 1 0]; % Rotation about Y-axis
shoulder.Joint = joint_shoulder;
addBody(robotArm, shoulder, 'Base');

% Shoulder to Elbow
elbow = rigidBody('Elbow');
joint_elbow = rigidBodyJoint('joint_elbow', 'revolute');
setFixedTransform(joint_elbow, trvec2tform([0, 0, 425]));
joint_elbow.JointAxis = [0 1 0]; % Rotation about Y-axis
elbow.Joint = joint_elbow;
addBody(robotArm, elbow, 'Shoulder');

% Elbow to Wrist1
wrist1 = rigidBody('Wrist1');
joint_wrist1 = rigidBodyJoint('joint_wrist1', 'revolute');
setFixedTransform(joint_wrist1, trvec2tform([0, 0, 392]));
joint_wrist1.JointAxis = [0 1 0]; % Rotation about Y-axis
wrist1.Joint = joint_wrist1;
addBody(robotArm, wrist1, 'Elbow');

% Wrist1 to Wrist2
wrist2 = rigidBody('Wrist2');
joint_wrist2 = rigidBodyJoint('joint_wrist2', 'revolute');
setFixedTransform(joint_wrist2, trvec2tform([54.65, 0, 47.375]));
joint_wrist2.JointAxis = [0 0 1]; % Rotation about Z-axis
```

```
wrist2.Joint = joint_wrist2;
addBody(robotArm, wrist2, 'Wrist1');

% Wrist2 to Wrist3
wrist3 = rigidBody('Wrist3');
joint_wrist3 = rigidBodyJoint('joint_wrist3', 'revolute');
setFixedTransform(joint_wrist3, trvec2tform([41.25, 0, 47.375]));
joint_wrist3.JointAxis = [0 1 0]; % Rotation about Y-axis
wrist3.Joint = joint_wrist3;
addBody(robotArm, wrist3, 'Wrist2');

% Wrist3 to End-effector
end_effector = rigidBody('EndEffector');
joint_ee = rigidBodyJoint('joint_ee', 'fixed');
setFixedTransform(joint_ee, trvec2tform([41.25, 0, 0]));
end_effector.Joint = joint_ee;
addBody(robotArm, end_effector, 'Wrist3');
```

Home Position



6c. Pose

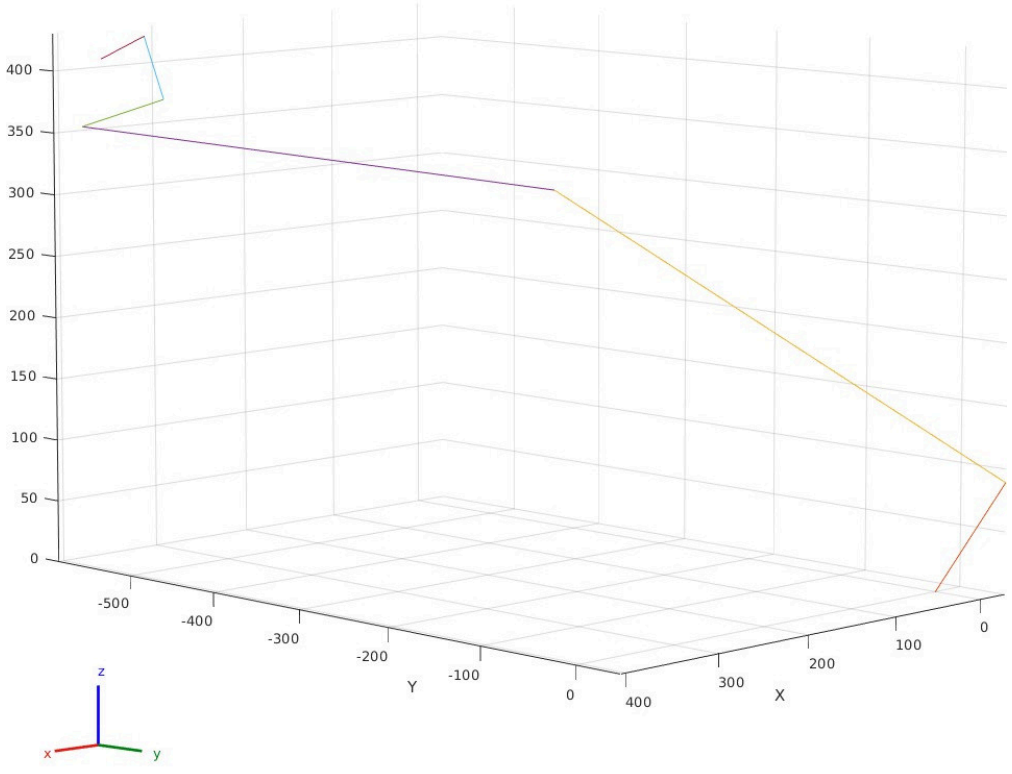
Question

6c. Assign a random configuration (angles) to this robot, using MATLAB, show your code (copy and paste here) and result (save as figure the insert here, not screen capture).

You can use other programming language such as Python/C/C++.
[5 marks]

```
randomPose = randomConfiguration(robotArm);
figure
show (robotArm, randomPose);
```

Random Pose



Bibliography

[1] S. H. Young, *Ultralearning: Accelerate Your Career, Master Hard Skills and Outsmart the Competition*. Thorsons, 2019.

[2] “Droid (Star Wars),” 27-Feb-2025. [Online]. Available: https://en.wikipedia.org/wiki/Droid_(Star_Wars)#List_of_droid_characters. [Accessed: 26-Mar-2025].

[3] “rigidBodyJoint.” [Online]. Available: https://www.mathworks.com/help/robotics/ref/rigidbodyjoint.html. [Accessed: 26-Mar-2025].

[4] D. Nguyen-Tuong and J. Peters, ‘Model learning for robot control: A survey’, *Cognitive processing*, vol. 12, pp. 319–40, Apr. 2011, doi: 10.1007/s10339-011-0404-1.

[5] X. Wang, Y. Li, and K.-W. Kwok, ‘A Survey for Machine Learning-Based Control of Continuum Robots’, *Front. Robot. AI*, vol. 8, Oct. 2021, doi: 10.3389/frobt.2021.730330.

[4] “rigidBody.” [Online]. Available: https://www.mathworks.com/help/robotics/ref/rigidbody.html. [Accessed: 26-Mar-2025].

[5] “se3.” [Online]. Available: https://www.mathworks.com/help/robotics/ref/se3.html. [Accessed: 26-Mar-2025].