Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

Task 5.1: DH Frame assignment (15 points)
Using standard DH convention, assign DH frames to the robot arm in Figure 5.1 . Make sure to clearly indicate the z and x axes, and the origin of each frame; drawing the y axis is optional. Place the origin of the end-effector frame at the centre of the gripper motor horn, for convenience of measurements in upcoming tasks. Draw and paste each frame's z and x-axis on the motor or link bodies of the robot. This will help your visualisation in later tasks.

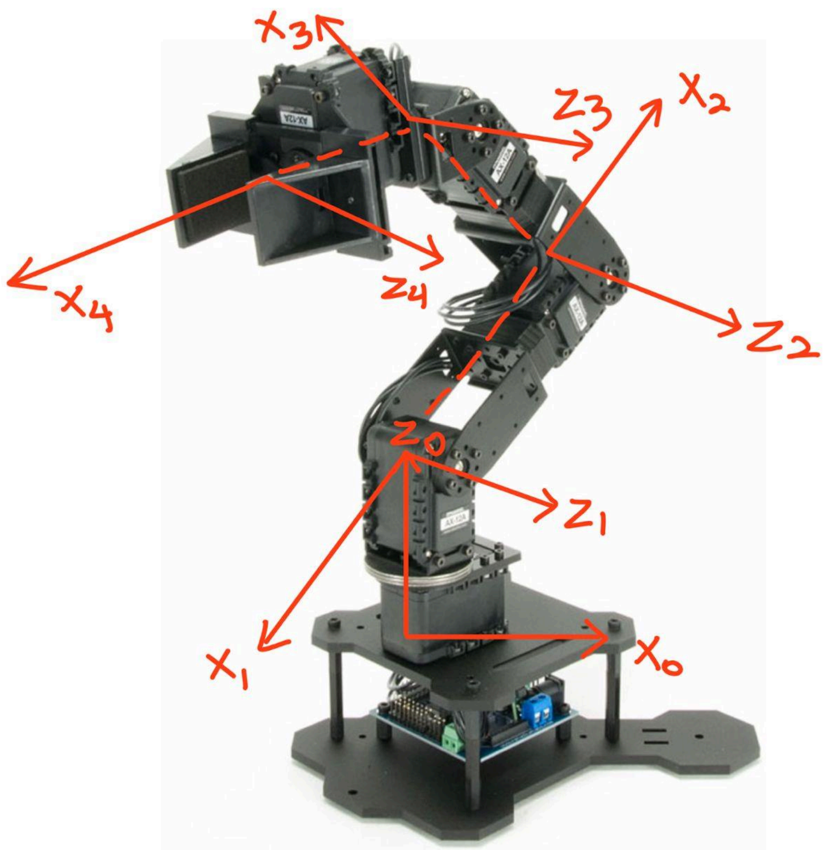


Figure 5.1: DH Frame assignment

Task 5.2 DH Parameters (15 points)
Annotate Table 5.1 with DH parameters based on your frame assignments, complete , and explain your process for determining the parameters where needed. You'll have to physically measure the values of some parameters.

Link	a_i	α_i	d_i	θ_i

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

1	0	0	L1 (0.04495m)	θ_0
2	L2 (0.1035m)	0	0	θ_1
3	L3 (0.10375m)	0	0	θ_2
4	L4 (0.111m)	0	0	θ_3

Task 5.3 Homogeneous Transformations (5+5+3 points)

Use MATLAB's symbolic math toolbox to determine the intermediate homogeneous transformations ${}^0T_1$, ${}^1T_2$, ${}^2T_3$, ${}^3T_4$, and the resultant transformation ${}^0T_4$.

(a) Write a MATLAB script to create symbolic matrices for all the homogeneous transformations listed above. Note that one of the parameters will be a joint variable.

- You can create a symbolic variable in MATLAB using `syms` function, e.g. `syms('theta_1')` will create a symbolic variable θ_1 in MATLAB. In case of a live script, the variable will also be displayed in Greek alphabet.
- The MATLAB functions `cos` and `sin` expect arguments in radians, while `cosd` and `sind` in degrees.
- Using the standard naming convention for the variables storing homogeneous transformations may result in convenience later, e.g. `T01` or `T_01`.

(b) Obtain ${}^0T_4$ by multiplying the previously determined homogeneous transformations in the appropriate order. The MATLAB functions `simplify` and `expand` may be of help in simplifying the final expressions.

(c) Provide expressions for the position and orientation of the end-effector frame with respect to the base frame.

a) This MATLAB function dhTransform computes the Denavit-Hartenberg (DH) transformation matrix, which encapsulates both translation and rotation between consecutive links in a serial robotic manipulator, given the DH parameters: link length a, link twist α, link offset d, and joint angle θ. The resulting 4x4 matrix comprises cosine and sine terms of the joint angle and link twist, multiplied by appropriate lengths and offsets, arranged to represent the transformation from one coordinate frame to another. This transformation matrix facilitates forward kinematics computations in robotics, aiding in determining the end-effector position and orientation based on the robot's joint configurations.

Matlab Code:

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

```
function T = dhTransform(a, alpha, d, theta)
  T = [cos(theta)   -sin(theta)*cos(alpha)  sin(theta)*sin(alpha)   a*cos(theta);
       sin(theta)   cos(theta)*cos(alpha)   -cos(theta)*sin(alpha)  a*sin(theta);
       0          sin(alpha)              cos(alpha)            d;
       0                  0                    0           1];
end
```

b) This MATLAB code calculates the homogeneous transformation matrices T_{01}, T_{12}, T_{23}, T_{34} \(using the DH parameters for a robotic manipulator, representing the transformations from one link to the next. T_{01} represents the transformation from the base to the first joint, T_{12} from the first joint to the second joint, and so on. Finally, the resultant transformation matrix T_{04} is computed by multiplying these individual transformation matrices together. This resultant matrix encapsulates the overall transformation from the base to the end-effector of the manipulator, considering the joint angles θ_1 through θ_4. The code effectively leverages the DH transformation approach to determine the end-effector's position and orientation based on the manipulator's joint configurations.

Code:

```
% Homogeneous Transformation matrices using DH parameters
T01 = dhTransform(0, pi/2, 0.04495, theta1);
T12 = dhTransform(0.1035, 0, 0, theta2);
T23 = dhTransform(0.10375, 0, 0, theta3);
T34 = dhTransform(0.111, 0, 0, theta4);
% Resultant transformation matrix
T04 = T01 * T12 * T23 * T34;
```

c)

```
syms theta1 theta2 theta3 theta4
% Homogeneous Transformation matrices using DH parameters
T01 = dhTransform(0, pi/2, 0.04495, theta1);
T12 = dhTransform(0.1035, 0, 0, theta2);
T23 = dhTransform(0.10375, 0, 0, theta3);
T34 = dhTransform(0.111, 0, 0, theta4);
% Resultant transformation matrix
simplify (T01 * T12 * T23 * T34);
disp(T04);
% T04 = T01 * T12 * T23 * T34;
%
% simplify(T04);
% disp(T04);
```

$$\begin{pmatrix} \cos(\theta_4)\,\sigma_3 - \sin(\theta_4)\,\sigma_5 & -\cos(\theta_4)\,\sigma_5 - \sin(\theta_4)\,\sigma_3 & \sin(\theta_1) & \frac{83\cos(\theta_3)\,\sigma_7}{800} + \frac{207\cos(\theta_1)\cos(\theta_2)}{2000} - \frac{83\sin(\theta_3)\,\sigma_8}{800} - \frac{1028325823204452777\sin(\theta_1)\sin(\theta_2)}{16225927682921336339157801028812800\,0} + \frac{111\cos(\theta_4)\,\sigma_3}{1000} - \frac{111\sin(\theta_4)\,\sigma_5}{1000} \\ \cos(\theta_4)\,\sigma_6 + \sin(\theta_4)\,\sigma_4 & \cos(\theta_4)\,\sigma_4 - \sin(\theta_4)\,\sigma_6 & -\cos(\theta_1) & \frac{83\cos(\theta_3)\,\sigma_{10}}{800} + \frac{83\sin(\theta_3)\,\sigma_9}{800} + \frac{1028325823204452777\cos(\theta_1)\sin(\theta_2)}{16225927682921336339157801028812800\,0} + \frac{207\cos(\theta_2)\sin(\theta_1)}{2000} + \frac{111\cos(\theta_4)\,\sigma_6}{1000} + \frac{111\sin(\theta_4)\,\sigma_4}{1000} \\ \cos(\theta_4)\,\sigma_2 + \sin(\theta_4)\,\sigma_1 & \cos(\theta_4)\,\sigma_1 - \sin(\theta_4)\,\sigma_2 & \frac{4967757600021511}{81129638414606681695789005144064} & \frac{207\sin(\theta_2)}{2000} + \frac{83\cos(\theta_2)\sin(\theta_3)}{800} + \frac{83\cos(\theta_3)\sin(\theta_2)}{800} + \frac{111\cos(\theta_4)\,\sigma_2}{1000} + \frac{111\sin(\theta_4)\,\sigma_1}{1000} + \frac{899}{20000} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$$\sigma_1 = \cos(\theta_2)\cos(\theta_3) - \sin(\theta_2)\sin(\theta_3)$$

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

Task 5.4 FK Function (10 points)

Provide a MATLAB `function [x,y,z,R] = pincherFK(jointAngles)` or `function [x,y,z,R,theta,phi] = pincherFK(jointAngles)` that accepts joint angles of Phantom X Pincher and returns the end-effector position and orientation in the specified order. Make sure to add comments describing the arguments and corresponding units.

The choice between the provided function definitions depends on which strategy you want to adopt for describing orientation, as outlined in the previous remark. You can also decide whether your function will accept arguments in degrees or radians. You can find help on how to create MATLAB functions at [2].

This MATLAB function "pincherFK" computes the end-effector position and orientation for a Phantom X Pincher robotic manipulator based on the input joint angles. It begins by extracting the joint angles θ_1 through θ_4 from the input "jointAngles". Then, it computes the homogeneous transformation matrices $T_{01}, T_{12}, T_{23}, T_{34}$ using the DH parameters for each joint configuration. Next, it multiplies these matrices to obtain the overall transformation matrix T_{04} representing the transformation from the base to the end-effector. The function then extracts the end-effector position (x, y, z) and orientation (R) from T_{04}. Additionally, it calculates the Euler angles θ and Φ from the rotation matrix R, representing the end-effector's orientation in terms of roll and pitch angles, respectively. Finally, it returns the computed position and orientation values. This function effectively leverages DH transformation techniques to determine the end-effector's pose based on the manipulator's joint angles.

```
function [x, y, z, R, theta, phi] = pincherFK(jointAngles)
 % Function to calculate end-effector position and orientation for Phantom X Pincher
  % Extract joint angles
 theta1 = jointAngles(1);
 theta2 = jointAngles(2);
 theta3 = jointAngles(3);
 theta4 = jointAngles(4);
 % Homogeneous Transformation matrices using DH parameters
 T01 = dhTransform(0, pi/2, 0.04495, theta1);
 T12 = dhTransform(0.1035, 0, 0, theta2);
 T23 = dhTransform(0.10375, 0, 0, theta3);
 T34 = dhTransform(0.111, 0, 0, theta4);
 % Resultant transformation matrix
 T04 = T01 * T12 * T23 * T34;
 % Extract position and orientation from the transformation matrix
 x = T04(1, 4);
 y = T04(2, 4);
```

Huzaifah, Asghar, Daniyal


```
 z = T04(3, 4);
 R = T04(1:3, 1:3);
 % Calculate theta and phi from the rotation matrix
 phi = atan2(sqrt(R(1,3)^2 + R(2,3)^2), R(3,3));
 theta = atan2(R(2,3)/sin(phi), R(1,3)/sin(phi));
end
```

Task 5.5 Verification of Forward Kinematic Mapping (5 points)

Enter your DH parameters from the previous task in `pincherModel.m`. The file should display a skeleton of the robot with frames. If you set your desired configuration, i.e. joint angles as the value of the `configNow` variable at the bottom of the file, the script returns the end-effector position and orientation with respect to the base frame, and displays the configuration graphically.

Select 4-5 random configurations for the manipulator and share the end-effector position and orientation, as determined by the provided `pincherModel` and your own `pincherFK` function. Make sure that they match. MATLAB command `randomConfiguration(robot)` can also generate a random configuration for `robot` in MATLAB workspace.

PINCHER MODEL:

```
% jA1 = [0 pi/2 pi pi/3];
% jA2 = [pi/2 0 pi/4 pi];
% jA3 = [pi/3 pi 0 3*pi/4];
function [] = pinchermodel2(jA)
configNow = jA;
dhparams = [0 pi/2 0.04495 jA(1);
        0.1035 0 0 jA(2);
        0.10375 0 0 jA(3);
        0.111 0 0 jA(4)];
numJoints = size(dhparams,1);
% Create a rigid body tree object.
robot = rigidBodyTree;
% Create a model of the robot using DH parameters.
% Create a cell array for the rigid body object, and another for the joint
% objects. Iterate through the DH parameters performing this process:
% 1. Create a rigidBody object with a unique name.
% 2. Create and name a revolute rigidBodyJoint object.
% 3. Use setFixedTransform to specify the body-to-body transformation of the
% joint using DH parameters.
% 4. Use addBody to attach the body to the rigid body tree.
```

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

```
bodies = cell(numJoints,1);
joints = cell(numJoints,1);
for i = 1:numJoints
  bodies{i} = rigidBody(['body' num2str(i)]);
  joints{i} = rigidBodyJoint(['jnt' num2str(i)],"revolute");
  setFixedTransform(joints{i},dhparams(i,:),"dh");
  bodies{i}.Joint = joints{i};
  if i == 1 % Add first body to base
      addBody(robot,bodies{i},"base")
  else % Add current body to previous body by name
      addBody(robot,bodies{i},bodies{i-1}.Name)
  end
end
% Verify that your robot has been built properly by using the showdetails or
% show function. The showdetails function lists all the bodies of the robot
% in the MATLAB® command window. The show function displays the robot with
% a specified configuration (home by default).
showdetails(robot)
figure(Name="Phantom X Pincher")
show(robot);
%% Forward Kinematics for different configurations
% Enter joint angles in the matrix below in radians
% configNow = [pi/3,pi/3,pi/3,pi/3]; assigned above already
% Display robot in provided configuration
config = homeConfiguration(robot);
for i = 1:numJoints
  config(i).JointPosition = configNow(i);
end
show(robot,config);
% Determine the pose of end-effector in provided configuration
poseNow = getTransform(robot,config,"body4");
% Display position and orientation of end-effector
clc;
disp('The position of end-effector is:');
disp('');
disp(['X: ', num2str(poseNow(1,4))]);
disp('');
disp(['Y: ', num2str(poseNow(2,4))]);
disp('');
disp(['Z: ', num2str(poseNow(3,4))]);
disp(' ');
disp(['R: ']);
poseNow(1:3,1:3)
disp(' ');
disp('The orientation angle is given with respect to the x-axis of joint 2:');
disp('');
poseNow01 = getTransform(robot,config,"body1");
```

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

```
R14 = poseNow01(1:3,1:3)'*poseNow(1:3,1:3);
angle = rad2deg(atan2(R14(2,1),R14(1,1)));
disp(['Angle: ',num2str(angle), ' degrees.']);
```

Verifying that it is indeed working!

```
[x, y, z, R, theta, phi] = pincherFK(jointAngles)

x = 0.0962
y
  -0.1457
 R = 3x3
    0.9794         0   -0.2021
   -0.2021    0.0000   -0.9794
   -0.0000    1.0000    0.0000

theta = -1.
phi = 1.5708
```

```
pinchermodel2(jointAngles)

--------------------
Robot: (4 bodies)

 Idx    Body Name   Joint Name   Joint Type    Parent Name(Idx)   Children Name(s)
 ---    ---------   ----------   ----------    ----------------   ----------------
   1        body1         jnt1     revolute             base(0)   body2(2)
   2        body2         jnt2     revolute            body1(1)   body3(3)
   3        body3         jnt3     revolute            body2(2)   body4(4)
   4        body4         jnt4     revolute            body3(3)
--------------------
The position of end-effector is:
X: 0.096203
Y: -0.019857

R:
 ans = 3x3
    0.9794         0   -0.2021
   -0.2021    0.0000   -0.9794
   -0.0000    1.0000    0.0000

The orientation angle is g      respect to the x-axis of joint 2:
Angle: -6.3611e-15 degrees.
```

```
[x, y, z, R, theta, phi] = pincherFK(jointAngles)

x = 0.0962
y = -0.0199
z = -0.1457
 R = 3x3
    0.9794         0   -0.2021
   -0.2021    0.0000   -0.9794
   -0.0000    1.0000    0.0000

theta = -1.7743
phi = 1.5708
```

```
pinchermodel2(jointAngles)

--------------------
Robot: (4 bodies)

 Idx    Body Name   Joint Name   Joint Type    Parent Name(Idx)   Children Name(s)
 ---    ---------   ----------   ----------    ----------------   ----------------
   1        body1         jnt1     revolute             base(0)   body2(2)
   2        body2         jnt2     revolute            body1(1)   body3(3)
   3        body3         jnt3     revolute            body2(2)   body4(4)
   4        body4         jnt4     revolute            body3(3)
--------------------
The position of end-effector is:
X: 0.096203
Y: -0.019857
Z: -0.14566

R:
 ans = 3x3
    0.9794         0   -0.2021
   -0.2021    0.0000   -0.9794
   -0.0000    1.0000    0.0000


The orientation angle is given with respect to the x-axis of joint 2:
Angle: -6.3611e-15 degrees.
```

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

```
[x, y, z, R, theta, phi] = pincherFK(jointAngles)
```

```
x = 0.0962
y = -0.0199
z = -0.1457
 R = 3×3
      0.9794         0   -0.2021
     -0.2021    0.0000   -0.9794
     -0.0000    1.0000    0.0000

theta = -1.7743
phi = 1.5708
```

```
pinchermodel2(jointAngles)
```

```
--------------------
Robot: (4 bodies)

 Idx     Body Name   Joint Name   Joint Type    Parent Name(Idx)   Children Name(s)
 ---     ---------   ----------   ----------    ----------------   ----------------
   1         body1         jnt1     revolute             base(0)   body2(2)
   2         body2         jnt2     revolute            body1(1)   body3(3)
   3         body3         jnt3     revolute            body2(2)   body4(4)
   4         body4         jnt4     revolute            body3(3)
--------------------
The position of end-effector is:
X: 0.096203
Y: -0.019857
Z: -0.14566

R:
 ans = 3×3
      0.9794         0   -0.2021
     -0.2021    0.0000   -0.9794
     -0.0000    1.0000    0.0000


The orientation angle is given with respect to the x-axis of joint 2:
Angle: -6.3611e-15 degrees.
```

```
[x, y, z, R, theta, phi] = pincherFK(jointAngles)
```

```
x = 0.0962
y = -0.0199
z = -0.1457
 R = 3×3
      0.9794         0   -0.2021
     -0.2021    0.0000   -0.9794
     -0.0000    1.0000    0.0000

theta = -1.7743
phi = 1.5708
```

```
pinchermodel2(jointAngles)
```

```
--------------------
Robot: (4 bodies)

 Idx     Body Name   Joint Name   Joint Type    Parent Name(Idx)   Children Name(s)
 ---     ---------   ----------   ----------    ----------------   ----------------
   1         body1         jnt1     revolute             base(0)   body2(2)
   2         body2         jnt2     revolute            body1(1)   body3(3)
   3         body3         jnt3     revolute            body2(2)   body4(4)
   4         body4         jnt4     revolute            body3(3)
--------------------
The position of end-effector is:
X: 0.096203
Y: -0.019857
Z: -0.14566

R:
 ans = 3×3
      0.9794         0   -0.2021
     -0.2021    0.0000   -0.9794
     -0.0000    1.0000    0.0000


The orientation angle is given with respect to the x-axis of joint 2:
Angle: -6.3611e-15 degrees.
```

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

```
[x, y, z, R, theta, phi] = pincherFK(jointAngles)
```

```
x = 0.0962
y = -0.0199
z = -0.1457
 R = 3x3
      0.9794         0   -0.2021
     -0.2021    0.0000   -0.9794
     -0.0000    1.0000    0.0000

theta = -1.7743
phi = 1.5708
```

```
pinchermodel2(jointAngles)
```

```
--------------------
Robot: (4 bodies)

 Idx    Body Name   Joint Name   Joint Type   Parent Name(Idx)   Children Name(s)
 ---    ---------   ----------   ----------   ----------------   ----------------
   1        body1         jnt1     revolute            base(0)   body2(2)
   2        body2         jnt2     revolute           body1(1)   body3(3)
   3        body3         jnt3     revolute           body2(2)   body4(4)
   4        body4         jnt4     revolute           body3(3)
--------------------
The position of end-effector is:
X: 0.096203
Y: -0.019857
Z: -0.14566

R:
 ans = 3x3
      0.9794         0   -0.2021
     -0.2021    0.0000   -0.9794
     -0.0000    1.0000    0.0000


The orientation angle is given with respect to the x-axis of joint 2:
Angle: -6.3611e-15 degrees.
```

This MATLAB script pincherl2 models a Phantom X Pincher robotic manipulator using DH parameters and performs forward kinematics to determine the end-effector position and orientation for a given set of joint angles jA.

First, it defines the DH parameters based on the provided joint angles jA. Then, it creates a rigid body tree object representing the robotic manipulator and builds the robot model by iteratively adding rigid bodies and revolute joints according to the DH parameters.

After building the robot model, it displays the details of the robot and visualizes it in its home configuration. Next, it displays the robot in the provided configuration specified by configNow.

Finally, it computes the pose of the end-effector in the provided configuration and displays the position and orientation information. The orientation angle is given with respect to the x-axis of joint 2.

This script serves as a comprehensive tool for modeling and analyzing the Phantom X Pincher robotic manipulator's kinematics.

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

Task 5.6 DH and Servo Joint Angles alignment (10 points)

Map the DH joint angles to the respective servomotor angles in Table 5.2 and Table 5.3. You'll have to determine (i) the possible angular shift between $0°$ of each DH joint angle (see the definition of joint angle in DH parameters) and the joint position when $0°$ command is sent to the corresponding servomotor, and (ii) whether the positive directions of rotation in the two cases are aligned. The determined shifts can be used to determine transform motor joint limits to DH specifications in Table 5.3.

Joint ID	DH Joint Angle (Θ_i)	Servo Angle ψ_i	Aligned Direction Of rotation(Yes/No)
#1	0°	0	Yes
#2	0°	-90	Yes
#3	0°	0	Yes
#4	0°	0	Yes

Table 5.2: Linear mapping between servo angles and DH angles

Joint ID	Minimum Joint Angle		Maximum Joint Angle	
	Servo angle	DH joint angle	Servo Angle	DH Joint Angle
#1	-150°	-150	150°	150
#2	-150°	-240	150°	60
#3	-150°	-150	150°	150
#4	-150°	-150	150°	150

Table 5.3: Joint Limits

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

Task 5.7 Mapping servo angles to DH angles (5 points)

Provide a MATLAB `function dhJointAngles = servo2dh(jointAngles)` that accepts joint angles of Phantom X Pincher, as understood by the servomotors, and convert them to your corresponding DH-assignment based joint angles. The function should be properly commented.

- The `jointAngles` vector contains joint angles, received from motor encoders, in order from the base to the wrist. The angles should either be in radians or angles.
- You need to find out the appropriate mapping function based on Table 5.2.

<table>
<tr><th rowspan="2">Joint ID</th><th colspan="2">Minimum Joint Angle</th><th colspan="2">Maximum Joint Angle</th></tr>
<tr><th>Servo angle</th><th>DH joint angle</th><th>Servo Angle</th><th>DH Joint Angle</th></tr>
<tr><td>#1</td><td>-150°</td><td>-150</td><td>150°</td><td>149</td></tr>
<tr><td rowspan="2">#2</td><td rowspan="2">-150°</td><td>-110 (reaches physical workspace limit given by servo)</td><td rowspan="2">150°</td><td>110 (same reason given by servo)</td></tr>
<tr><td>-200 (by dh)</td><td>30 (by dh)</td></tr>
<tr><td>#3</td><td>-150°</td><td>-149</td><td>150°</td><td>148 (doesn’t accept 150)</td></tr>
<tr><td>#4</td><td>-150°</td><td>-105.5</td><td>150°</td><td>103</td></tr>
</table>

Table 5.4: Physical Joint Limits

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

Task 5.8 Identifying reachable workspace (10 points)

Use the outlined idea of determining end-effector positions for selected joint configurations (uniform or random) to plot the reachable workspace of our Phantom X Pincher robot arm. Provide an isometric view of the workspace as well as a top-view, i.e. a projection of your workspace onto $X-Y$ plane of your base frame. Remember to mark axes in your plots. What is the maximum horizontal reach according to your identified workspace?

- The MATLAB function `rand` generates uniform pseudorandom numbers in $[0, 1]$. We can generate N samples for a joint angle θ_i, with lower bound θ_i^{min} and upper bound θ_i^{max} using the expression:

$$\theta_i = \theta_i^{\text{min}} + \left(\theta_i^{\text{max}} - \theta_i^{\text{min}}\right) \times \texttt{rand(N,1)}.$$

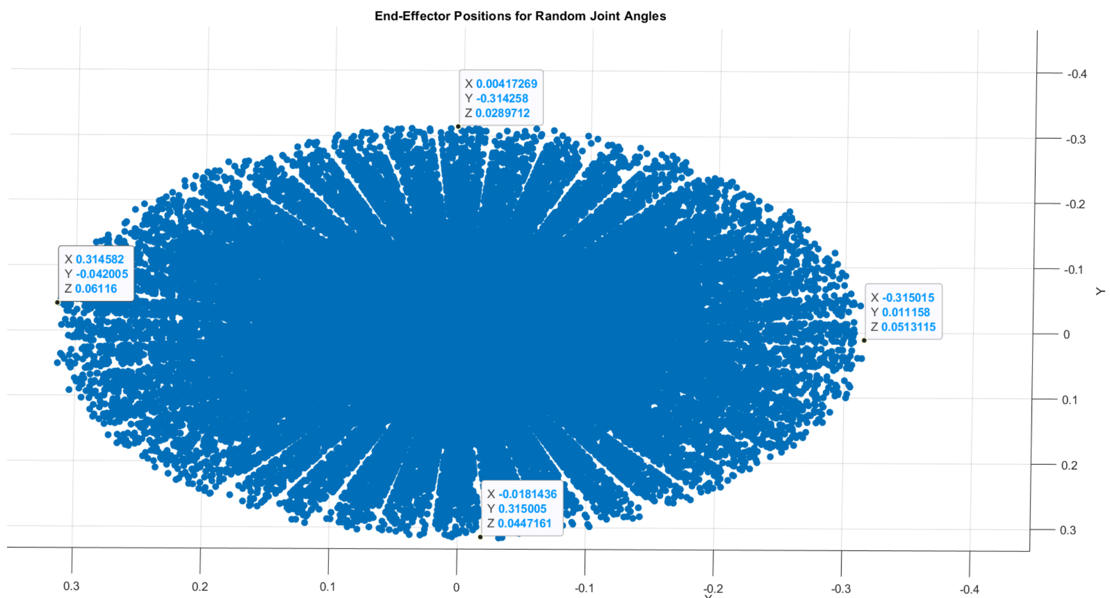
- The MATLAB functions `linspace`, `ndgrid`, and `scatter3` may be of help for this task.

```
joint_angles_random = zeros(1, 4);

% Initialize arrays to store x, y, z coordinates
x_points = zeros(1, num_points);
y_points = zeros(1, num_points);
z_points = zeros(1, num_points);

num_points = 100000;

for i = 1:num_points
    for j = 1:4
        joint_angles_random(i, j) = randi([-150, 150]);
    end
    joint_angles_random(i, 2) = joint_angles_random(i, 2) + 90;
end
joint_angles_random(2) = joint_angles_random(2) + 90;


%disp(joint_angles_random); % Display the random array
[x_points(i), y_points(i), z_points(i), R, theta, phi] = pincherFK(joint_angles_random);
% Compute end-effector positions for all points
for i = 1:num_points
    [x_points(i), y_points(i), z_points(i), ~, ~, ~] = pincherFK(joint_angles_random(i, :));
end


% Plot the points in 3D space
scatter3(x_points, y_points, z_points, 'filled');
xlabel('X');
ylabel('Y');
zlabel('Z');
title('End-Effector Positions for Random Joint Angles');
```

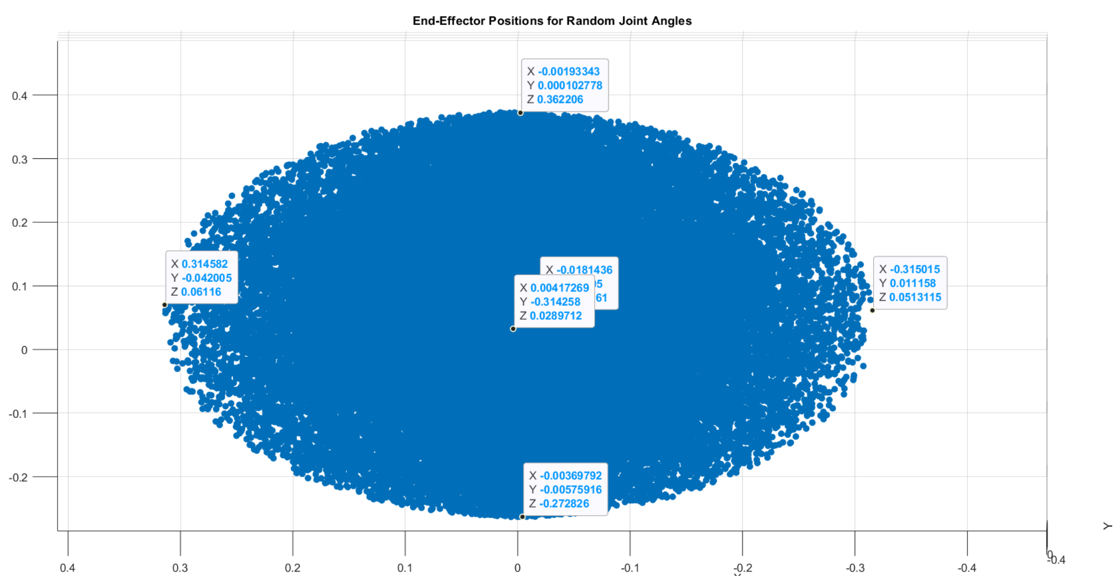
X-Y axis

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal



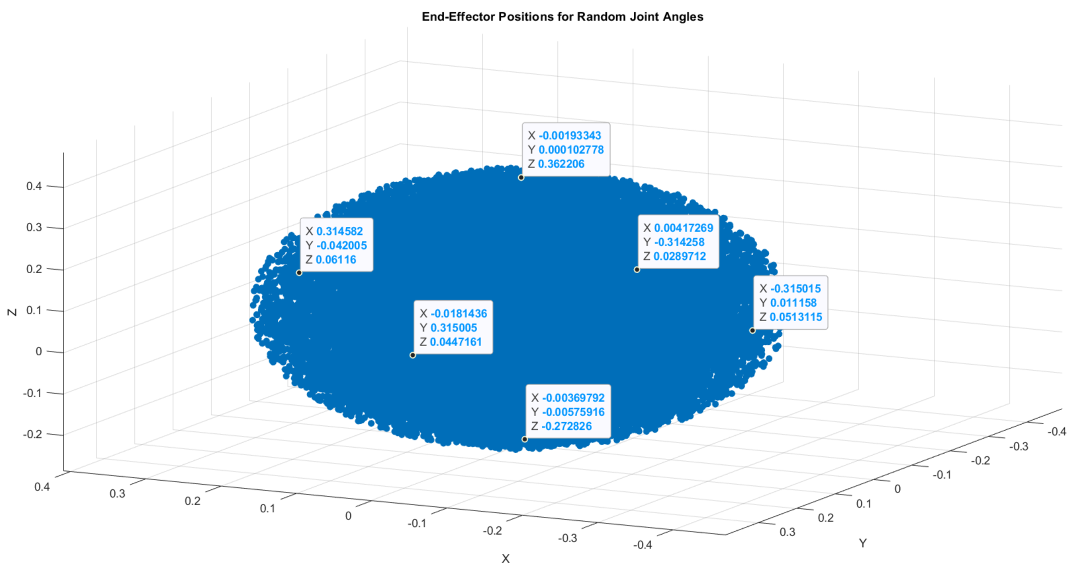
X-Z axis (Vertical axis below is Z-axis)



Another general view of the workspace in three-axis

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal



Maximum Horizontal Reach in X-Y plane can be observed as

P1 = (0.315, -0.042, 0.06116);
P2 = (-0.315, 0.011, 0.05);
P3 = (-0.0181436, 0.315005, 0.044716);
P4 = (0.00417, -0.3143, 0.0289712);
P5 = (-0.00193343, 0.000102778, 0.362206);
P6 = (-0.00369792, -0.00575916, -0.272826);

Along X-axis P1 to P2, **0.315** - (-0.315) = 0.630m (answer)
Along Y-axis P3 to P4, 0.315 - (-0.3143) = 0.629m
Along Z-axis P5 to P6, 0.362206 -(-0.272826) = 0.635032 ≈ 0.635m

Maximum reach is 0.315m

Task 5.9 Communicating with motors (7 points)

Provide a MATLAB `function [x,y,z,R] = findPincher()` or `function [x,y,z,R,theta,phi] = findPincher()` that queries the current servo angles from Phantom X Pincher motor encoders and returns the current end-effector position and orientation in the specified order. The function should be properly commented.

- Physically measure and note the end-effector position and orientation. How does it compare to the pose returned by your function?

- Do you think the pose returned by this process will ever be accurate? If not, what do you think are the sources of error?

```
function [x, y, z, R, theta, phi] = pincherFK(jointAngles)
 % Function to calculate end-effector position and orientation for Phantom X Pincher
  % Extract joint angles
 theta1 = jointAngles(1);
 theta2 = jointAngles(2);
 theta3 = jointAngles(3);
 theta4 = jointAngles(4);
 % Homogeneous Transformation matrices using DH parameters
 T01 = dhTransform(0, pi/2, 0.04495, theta1);
 T12 = dhTransform(0.1035, 0, 0, theta2);
 T23 = dhTransform(0.10375, 0, 0, theta3);
 T34 = dhTransform(0.111, 0, 0, theta4);
 % Resultant transformation matrix
 T04 = T01 * T12 * T23 * T34;
 % Extract position and orientation from the transformation matrix
 x = T04(1, 4);
 y = T04(2, 4);
 z = T04(3, 4);
 R = T04(1:3, 1:3);
 % Calculate theta and phi from the rotation matrix
 phi = atan2(sqrt(R(1,3)^2 + R(2,3)^2), R(3,3));
 theta = atan2(R(2,3)/sin(phi), R(1,3)/sin(phi));
end
```

We positioned the arm such that $\theta_1 = \pi/2$ and the other angles were set to zero. Subsequently, we determined the end-effector position experimentally, and our results were verified using MATLAB.

Physical Measurement:

The experimental observation of the end-effector position yielded coordinates (x,y,z) in metres which were as follows.

x = 0

y = 0.314

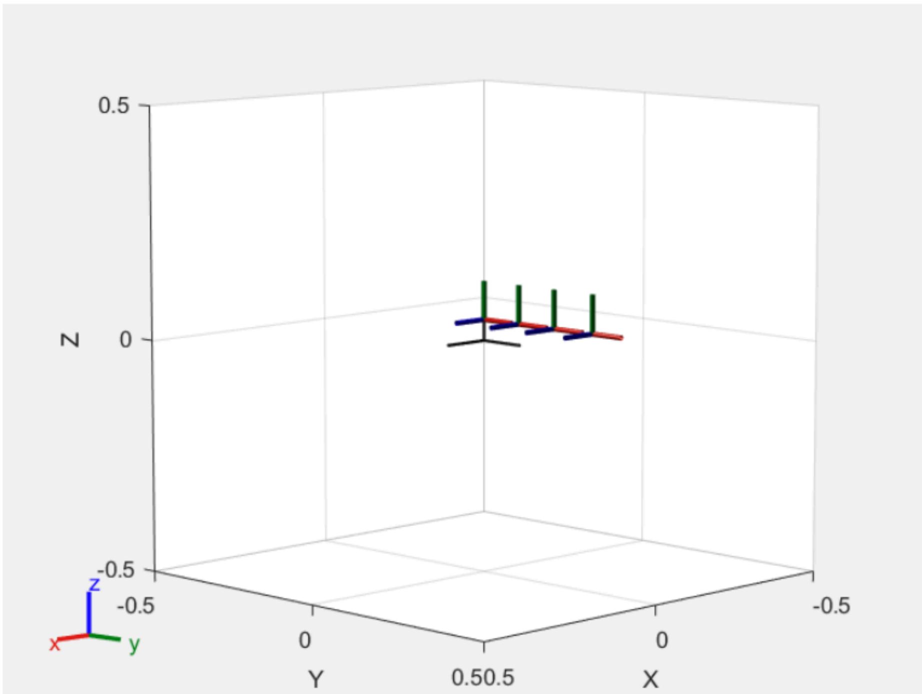
Z = 0.104 - 0.140 (as the base frame is abit on the top rather than on the ground level) = -0.036

Ideally, it should have been

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

```
--------------------
```




```
The position of end-effector is:
X: 1.9487e-17
Y: 0.31825
Z: 0.04495
```

Observation:

The accuracy of the pose returned by this process vary. Factors such as mechanical play in the joints, inaccuracies in motor encoders, errors in the DH parameters, and environmental conditions (e.g., temperature variations affecting the robot's components) can contribute to inaccuracies in the calculated pose. Regular calibration and error correction techniques can help improve accuracy to some extent, but achieving perfect accuracy is always challenging due to these inherent sources of error

Although we expected a perfect 0 on X-axis, in practical axis, it is very negligible. Y also varies only slightly.

The most interesting observation is on Z-axis, while we expected the end-effector position to be “positive” even if it was only by 0.04m, we instead got -0.036m. This can easily be explained by the fact that we have considered the **role of gravity** in our model. As it’s a straight-hand pointing forward, due to its’ nature having mechanical parts, it bents slightly downwards naturally in the physical world.

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

In our model, we expected a completely ideal rigid body! If we don't model for that, we won't ever get a completely accurate reading in regards to this.

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

Task 5.10 Mapping DH angles to servo angles (10 points)

Write a helper MATLAB `function [servoJointAngles,errorCode] = dh2servo(jointAngles)` that accepts DH joint angles of Phantom X Pincher and convert them to corresponding servomotor angles. The function should be properly commented.

- The `jointAngles` vector contains joint angles, according to DH frame assignment, in order from the base to the wrist. The angles should either be in radians or angles.
- The angle limits for the motors are $[-150°, 150°]$, and your function should return an empty array for `servoJointAngles` and an appropriate error code if the provided joint angles map to angles outside this limit.
- You will have to map the computed servo angle θ_i to its corresponding value in $[-\pi, \pi]$ to compare against the allowed joint limits. One way to find the equivalent value of θ in the interval $[-\pi, \pi]$ is `angle = mod(theta+π,2π)-π`.
- You need to find out the appropriate mapping function, based on Table 5.2.
- You have freedom to choose complexity of the error reporting system. It could be as simple as `errorCode=0` if all angles are within limits, and `errorCode=1`, if they're not.

Task 5.10 (Creating the dh2servo function to be used)

```
%Creating jointAngles array consisting of our DH angles.
DH_JointAngle1 = 90;
DH_JointAngle2 = -240;
DH_JointAngle3 = 150;
DH_JointAngle4 = 90;

% Prompt the user for joint angles
% DH_JointAngle1 = input('Enter DH Joint Angle 1 in degrees: ');
% DH_JointAngle2 = input('Enter DH Joint Angle 2 in degrees: ');
% DH_JointAngle3 = input('Enter DH Joint Angle 3 in degrees: ');
% DH_JointAngle4 = input('Enter DH Joint Angle 4 in degrees: ');

jointAngles = [DH_JointAngle1, DH_JointAngle2, DH_JointAngle3, DH_JointAngle4];
[servoJointAngles,errorCode] = dh2servo(jointAngles);
```

```
DH Joint angles are: 90 -240  150   90
Servo Joint angles are: 1.5708      2.0944      2.618      1.5708
In degrees without pi to pit transform, Servo Joint angles are: 90 -150  150   90
```

```
if (errorCode == 1)
    servoJointAngles = []; % Empty array for servo joint angles
    error('Joint angles are outside the allowable range of [-150, 150] degrees')
end
```

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

When we give out-of-range values:

Task 5.10 (Creating the dh2servo function to be used)

```
%Creating jointAngles array consisting of our DH angles.
DH_JointAngle1 = -151;
DH_JointAngle2 = -240;
DH_JointAngle3 = 150;
DH_JointAngle4 = 90;

% Prompt the user for joint angles
% DH_JointAngle1 = input('Enter DH Joint Angle 1 in degrees: ');
% DH_JointAngle2 = input('Enter DH Joint Angle 2 in degrees: ');
% DH_JointAngle3 = input('Enter DH Joint Angle 3 in degrees: ');
% DH_JointAngle4 = input('Enter DH Joint Angle 4 in degrees: ');

jointAngles = [DH_JointAngle1, DH_JointAngle2, DH_JointAngle3, DH_JointAngle4];
[servoJointAngles,errorCode] = dh2servo(jointAngles);
```

```
DH Joint angles are: -151 -240  150   90
Servo Joint angles are: -2.6354      2.0944       2.618      1.5708
In degrees without pi to pit transform, Servo Joint angles are: -151 -150  150   90
```

```
if (errorCode == 1)
    servoJointAngles = []; % Empty array for servo joint angles
    error('Joint angles are outside the allowable range of [-150, 150] degrees')
end
```

```
Joint angles are outside the allowable range of [-150, 150] degrees
```

As can be observed, error is given *"Joint angles are outside of the allowable range of [-150 150] degrees"* , and the code exits.

FUNCTION OF DH2SERVO

```
function [servoJointAngles,errorCode]= dh2servo(jointAngles);

%display
str_jointAngles = ['DH Joint angles are: ' num2str(jointAngles)];
disp(str_jointAngles);
%Assignment of servojointAngles
servojointAngles = [jointAngles(1), jointAngles(2) + 90, jointAngles(3), jointAngles(4)];


%Taking everything to Radians and setting it to -pi to pi limit!
% Converting DH jointangles from degrees to radians

jointAngles_rad = deg2rad(jointAngles);
servojointAngles_rad = mod(jointAngles_rad + pi, 2*pi) - pi; % Apply transformation to the range [-pi, pi], eval converts mathematical to scalar

%%%%%%%%%%%%%%%%%% If compairsion was done in Radians, which we aren't
%%%%%%%%%%%%%%%%%% doing for the sake of simplicity
% precision_level = 32; %Increasing precision will give us mathematical expression!
% jointAngles_rad_precise = vpa(deg2rad(jointAngles), precision_level); %Converts JointAngles to radians while increasing the precision from default 8 to precision_level!
% servojointAngles_rad_precise = mod(jointAngles_rad_precise + pi, 2*pi) - pi;% Apply transformation to the range [-pi, pi]
%
% % Checking if any joint angle is outside the limits
% angleLimits = [-150, 150];
% angleLimits_rad = vpa(deg2rad(angleLimits), precision_level); % Converting to rad, -2.618 to 2.618
% %we will use any() for normal case, but isAlways for mathematical
% %Increasing precision will give us mathematical expression!
% isAlways(servojointAngles_rad > angleLimits_rad(2));
% array_result = isAlways(servojointAngles_rad < angleLimits_rad(1)) | isAlways(servojointAngles_rad > angleLimits_rad(2));
% errorCode= any(array_result); %Violation of limits
%%%%%%%%%
```

```
%%%%%%%%%

%%degrees decimal comparision
angleLimits = [-150, 150];
errorCode = any(servojointAngles < angleLimits(1)) || any(servojointAngles > angleLimits(2));


str_servojointAngles_rad = ['Servo Joint angles are: ' num2str(servojointAngles_rad)];
disp(str_servojointAngles_rad);
%EXTRA
str_servojointAngles = ['In degrees without pi to pit transform, Servo Joint angles are: ' num2str(servojointAngles)];
disp(str_servojointAngles);

servoJointAngles = servojointAngles;
end
```

Intro to Robotics Lab 5

Huzaifah, Asghar, Daniyal

Appendix:

The drive folder link where all the functions, livescript as well as pdf uploaded can be found here:

Robotics Lab 5