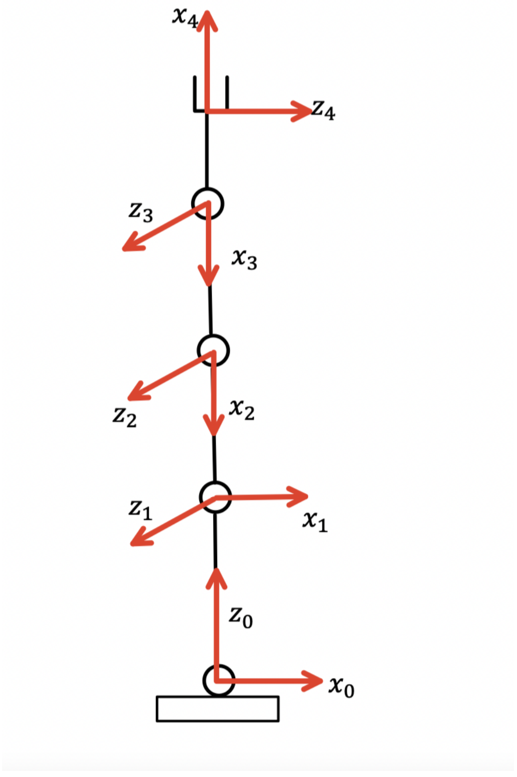
Robotics Lab 04

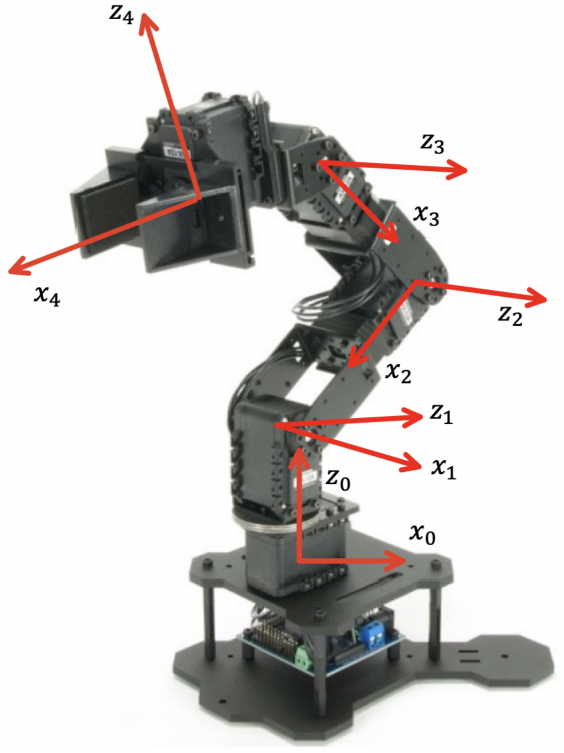
Syeda Manahil Wasti, Syed Mujtaba Hassan, Aatiqa Khalid

February 19, 2023

1. Using standard DH convention, assign DH frames to the robot arm in Figure 4.1. 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 center of gripper motor horn, for convenience of measurements in upcoming tasks.

Solution:





2. Annotate Figure 4.1 with DH parameters based on your frame assignments, complete Table 4.1, and explain your process for determining the parameters where needed. You'll have to physically measure the values of some parameters.

Solution:

	Link Length (a_i)	Link Twist (α_i)	Link Offset (d_i)	Joint Angle (θ_i)
1	0 cm	90°	13 cm	θ_1
2	10 cm	0°	0 cm	θ_2
3	10 cm	0°	0 cm	θ_3
4	10 cm	0°	0 cm	θ_4

Link lengths were physically measured. First one is zero because z_0 and z_1 intersect.
As the axis of rotation of joint 2, 3, and 4 are parallel, their link twist is zero and as axis of rotation of joint 1 and 2

are perpendicular, their link twist is 90°.
As axis of rotation of joint 1 and 2 are perpendicular, their link offset was measured along z_0. The axis of rotation of joint 2, 3, and 4 are parallel so their link offsets are zero.
As all the joints are revolute their joint angles are variable.

3. Use MATLAB's symbolic math toolbox to determine the intermediate homogeneous trans- formations ${}^0T_1$, ${}^1T_2$, ${}^2T_3$, ${}^3T_4$, 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.

Solution:

```
syms('theta_1')
syms('theta_2')
syms('theta_3')
syms('theta_4')

T01 = [cos(theta_1), 0, sin(theta_1), 0;
        sin(theta_1), 0, -cos(theta_1), 0;
        0, 1, 0, 130;
        0, 0, 0, 1];

T12 = [cos(theta_2), -sin(theta_2), 0, 110*cos(theta_2);
        sin(theta_2), cos(theta_2), 0, 110*sin(theta_2);
        0, 0, 1, 0;
        0, 0, 0, 1];

T23 = [cos(theta_3), -sin(theta_3), 0, 110*cos(theta_3);
        sin(theta_3), cos(theta_3), 0, 110*sin(theta_3);
        0, 0, 1, 0;
        0, 0, 0, 1];

T34 = [cos(theta_4), -sin(theta_4), 0, 110*cos(theta_4);
        sin(theta_4), cos(theta_4), 0, 110*sin(theta_4);
        0, 0, 1, 0;
        0, 0, 0,1];

T04 = T01*T12*T23*T34
```

   (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.

Solution:

$$\begin{bmatrix} -\cos\theta_4\sigma_3 - \sin\theta_4\sigma_4 & \sin\theta_4\sigma_3 - \cos\theta_4\sigma_4 & \sin\theta_1 & 110\cos\theta_1\cos\theta_2 - 110\cos\theta_4\sigma_3 - 110\sin\theta_4\sigma_4 - 110\cos\theta_1\sin\theta_2\sin\theta_3 + 110\cos\theta_1\cos\theta_2\cos\theta_3 \\ -\cos\theta_4\sigma_1 - \sin\theta_4\sigma_2 & \sin\theta_4\sigma_1 - \cos\theta_4\sigma_2 & -\cos\theta_1 & 110\cos\theta_2\sin\theta_1 - 110\cos\theta_4\sigma_1 - 110\sin\theta_4\sigma_2 - 110\sin\theta_1\sin\theta_2\sin\theta_3 + 110\cos\theta_2\cos\theta_3\sin\theta_1 \\ \cos\theta_4\sigma_6 + \sin\theta_4\sigma_5 & \cos\theta_4\sigma_5 - \sin\theta_4\sigma_6 & 0 & 110\sin\theta_2 + 110\cos\theta_2\sin\theta_3 + 110\cos\theta_3\sin\theta_2 + 110\cos\theta_4\sigma_6 + 110\sin\theta_4\sigma_5 + 130 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Solution:
Orientation:

$$R_{3x3} = \begin{bmatrix} -cos(\theta_4)\sigma_3 - sin(\theta_4)\sigma_4 & sin(\theta_4)\sigma_3 - cos(\theta_4)\sigma_4 & sin(\theta_1) \\ -cos(\theta_4)\sigma_1 - sin(\theta_4)\sigma_2 & sin(\theta_4)\sigma_1 - cos(\theta_4)\sigma_2 & -cos(\theta_1) \\ cos(\theta_4)\sigma_6 + sin(\theta_4)\sigma_5 & cos(\theta_4)\sigma_5 - sin(\theta_4)\sigma_6 & 0 \end{bmatrix}$$

Position:

$$P_{3x1} = \begin{bmatrix} 110cos(\theta_1)cos(\theta_2) - 110cos(\theta_4)\sigma_3 - 110sin(\theta_4)\sigma_4 - 110cos(\theta_1)sin(\theta_2)sin(\theta_3) + 110cos(\theta_1)cos(\theta_2)cos(\theta_3) \\ 110cos(\theta_2)sin(\theta_1) - 110cos(\theta_4)\sigma_1 - 110sin(\theta_4)\sigma_2 - 110sin(\theta_1)sin(\theta_2)sin(\theta_3) + 110cos(\theta_2)cos(\theta_2)sin(\theta_1) \\ 110sin(\theta_2) + 110cos(\theta_2)sin(\theta_3) + 110cos(\theta_3)sin(\theta_2) + 110cos(\theta_4)\sigma_6 + 110sin(\theta_4)\sigma_5 + 130 \end{bmatrix}$$

where

$$\sigma_1 = \sin(\theta_1)\,\sin(\theta_2)\,\sin(\theta_3) - \cos(\theta_2)\,\cos(\theta_3)\,\sin(\theta_1)$$

$$\sigma_2 = \cos(\theta_2)\,\sin(\theta_1)\,\sin(\theta_3) + \cos(\theta_3)\,\sin(\theta_1)\,\sin(\theta_2)$$

$$\sigma_3 = \cos(\theta_1)\,\sin(\theta_2)\,\sin(\theta_3) - \cos(\theta_1)\,\cos(\theta_2)\,\cos(\theta_3)$$

$$\sigma_4 = \cos(\theta_1)\,\cos(\theta_2)\,\sin(\theta_3) + \cos(\theta_1)\,\cos(\theta_3)\,\sin(\theta_2)$$

$$\sigma_5 = \cos(\theta_2)\,\cos(\theta_3) - \sin(\theta_2)\,\sin(\theta_3)$$

$$\sigma_6 = \cos(\theta_2)\,\sin(\theta_3) + \cos(\theta_3)\,\sin(\theta_2)$$

4. Provide a MATLAB function [x,y,z,R] = findPincher(jointAngles) or function [x,y,z,R,theta,phi] = findPincher(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 function definitions depends on whether you want to adopt the strategy for describing orientation outlined in the previous remark. You can also decide whether your function will accept arguments in degrees or radians, and whether you want to . You can find help on how to create MATLAB functions at [2] and [3].

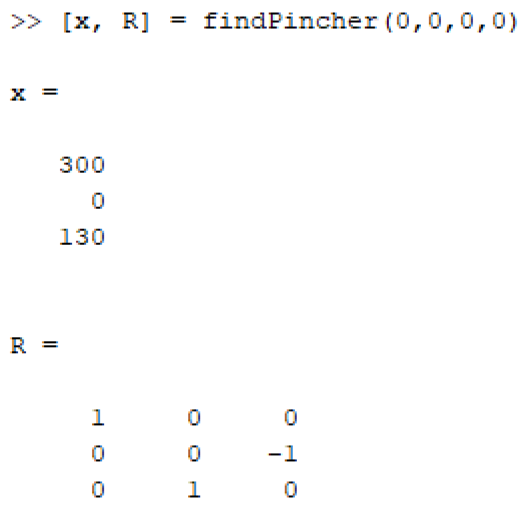
Solution:
```
function [x, R] = findPincher(theta_1, theta_2, theta_3, theta_4) % takes joint
   angles (in radians) as argument and returns the end-effector position and
   orientation

% transformation from base frame ( frame 0) to frame 1:
T01 = [cos(theta_1), 0, sin(theta_1), 0;
        sin(theta_1), 0, -cos(theta_1), 0;
        0, 1, 0, 130;
        0, 0, 0, 1];

% transformation from frame 1 to frame 2:
T12 = [cos(theta_2), -sin(theta_2), 0, 100*cos(theta_2);
        sin(theta_2), cos(theta_2), 0, 100*sin(theta_2);
        0, 0, 1, 0;
        0, 0, 0, 1];

% transformation from frame 2 to frame 3:
T23 = [cos(theta_3), -sin(theta_3), 0, 100*cos(theta_3);
```

```
        sin(theta_3), cos(theta_3), 0, 100*sin(theta_3);
        0, 0, 1, 0;
        0, 0, 0, 1];

% transformation from frame 3 to frame 4:
T34 = [cos(theta_4), -sin(theta_4), 0, 100*cos(theta_4);
        sin(theta_4), cos(theta_4), 0, 100*sin(theta_4);
        0, 0, 1, 0;
        0, 0, 0,1];

% total transformation from base frame (frame 0) to frame 4:
T04 = T01*T12*T23*T34;

x = T04(1:3, 4); % position vector obtained from T04 through indexing is stored
   in variable x

R = T04(1:3, 1:3); % rotation matrix obtained from T04 through indexing is stored
    in variable R
end
```

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

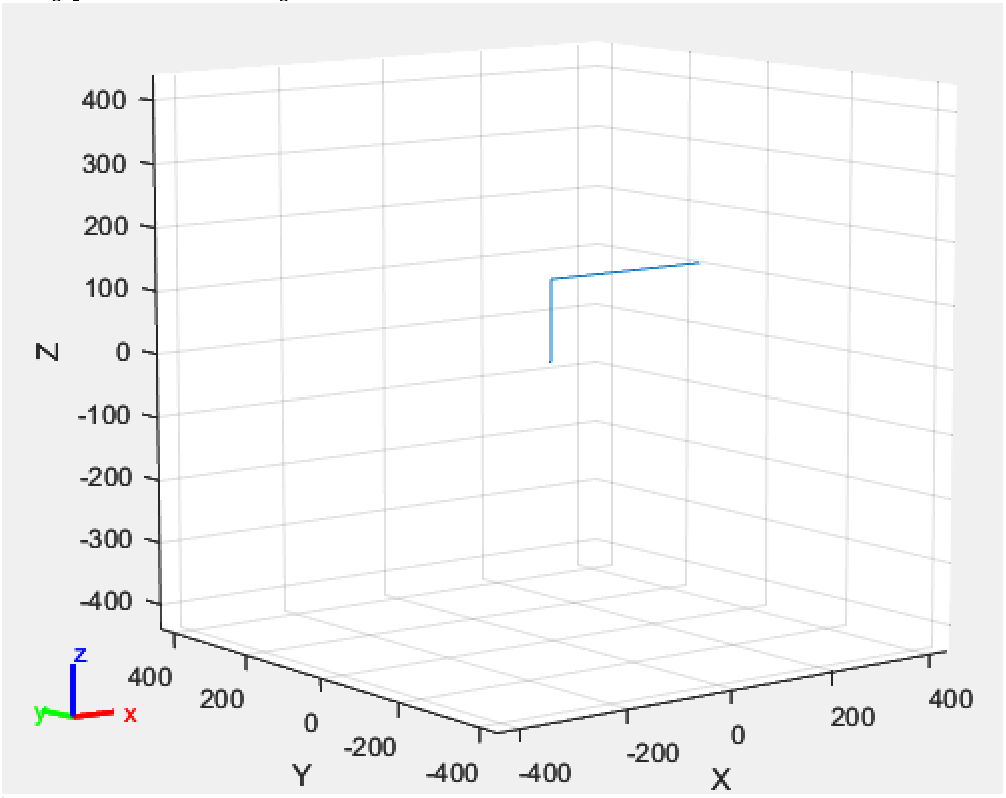
   Select 4-5 random configurations for the manipulator and share the end-effector position and orientation, as determined by the provided pincherModel and your own findPincher function. Make sure that they match. MATLAB command randomConfiguration(robot) can also generate a random configuration for robot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

Solution:
First configuration (0, 0, 0, 0):
using findPincher we get:



```
>> [x, R] = findPincher(0,0,0,0)

x =

   300
     0
   130


R =

     1     0     0
     0     0    -1
     0     1     0
```

using pincherModel we get:



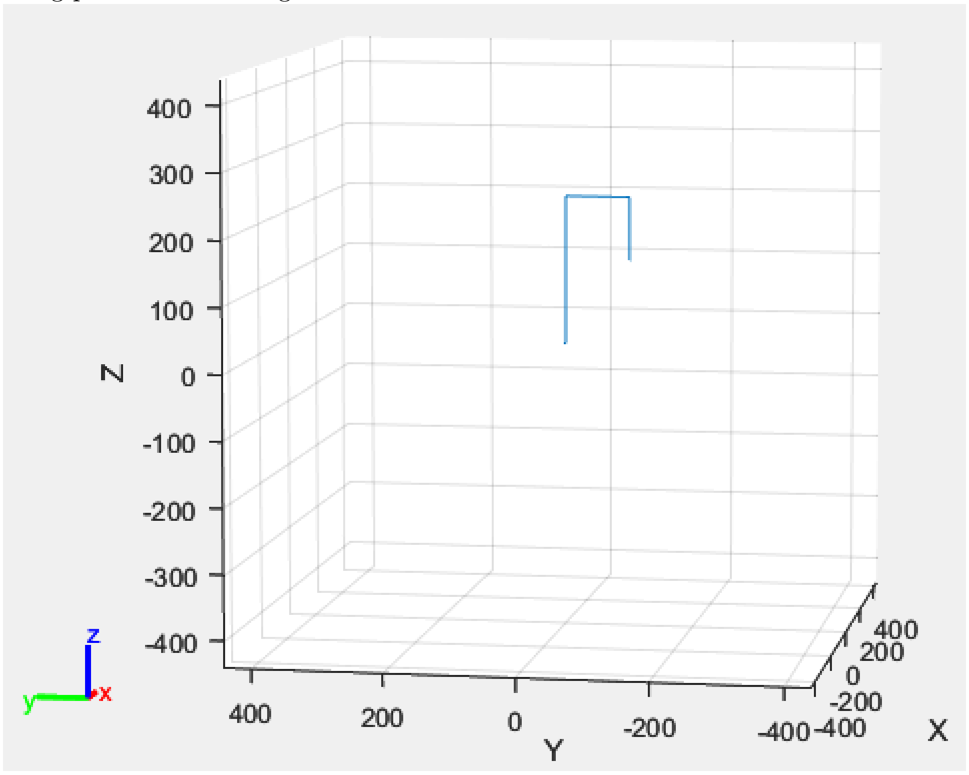
```
The position of end-effector is:
X: 300
Y: 0
Z: 130

R:

ans =

    1.0000         0         0
         0    0.0000   -1.0000
         0    1.0000    0.0000


The orientation angle is given with respect to the x-axis of joint 2:
Angle: 0 degrees.
```

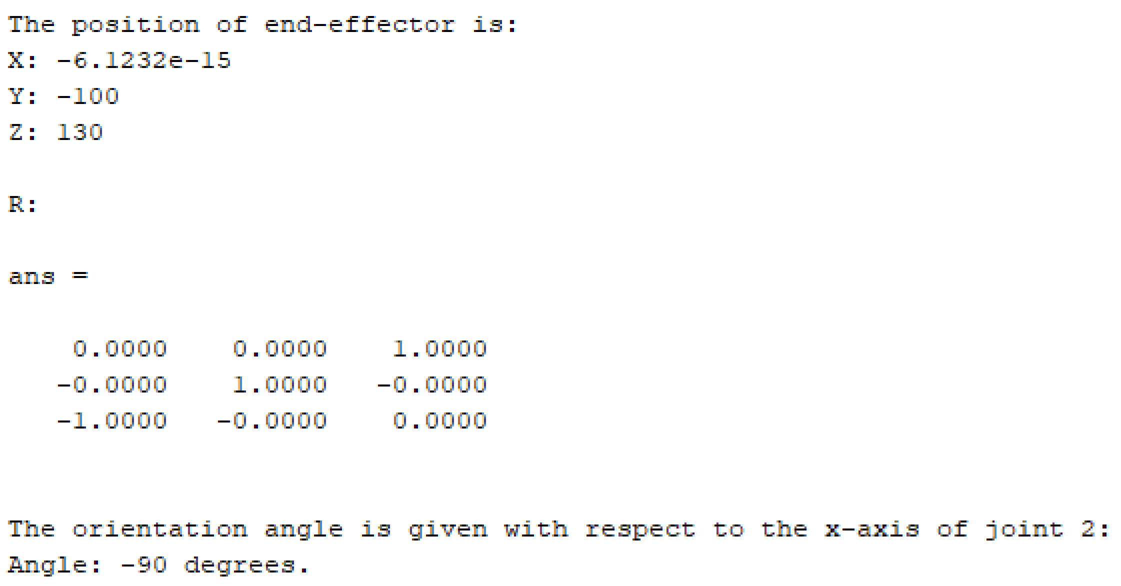
Second configuration (pi/2, pi/2, pi/2, pi/2):

using findPincher we get:

```
>> [x, R] = findPincher(pi/2,pi/2,pi/2,pi/2)

x =

   -0.0000
 -100.0000
  130.0000


R =

   -0.0000    0.0000    1.0000
   -0.0000    1.0000   -0.0000
   -1.0000   -0.0000         0
```

using pincherModel we get:

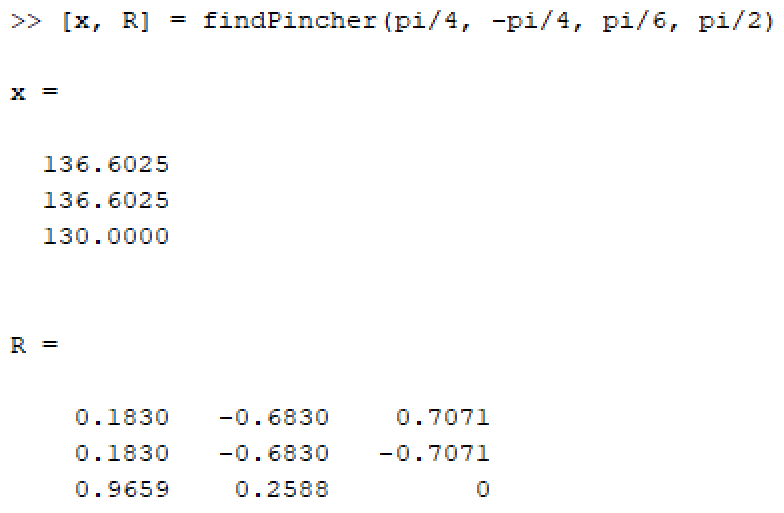




```
The position of end-effector is:
X: -6.1232e-15
Y: -100
Z: 130

R:

ans =

    0.0000    0.0000    1.0000
   -0.0000    1.0000   -0.0000
   -1.0000   -0.0000    0.0000


The orientation angle is given with respect to the x-axis of joint 2:
Angle: -90 degrees.
```

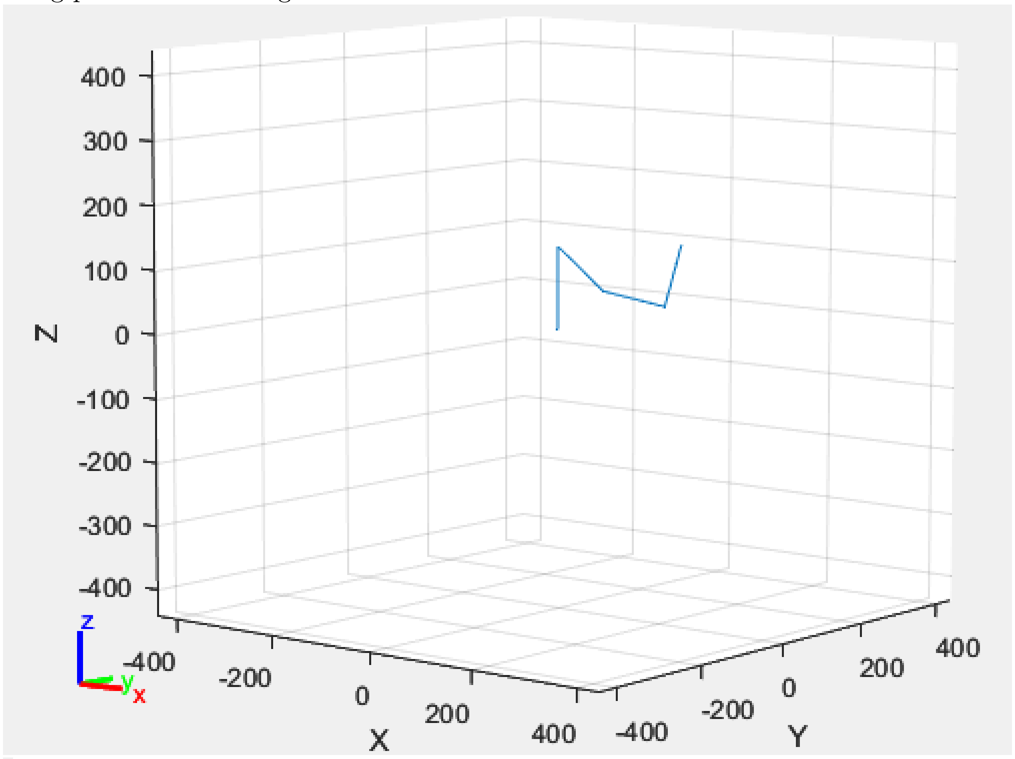
Third configuration (pi/4, -pi/4, pi/6, pi/2):

using findPincher we get:



```
>> [x, R] = findPincher(pi/4, -pi/4, pi/6, pi/2)

x =

  136.6025
  136.6025
  130.0000


R =

    0.1830   -0.6830    0.7071
    0.1830   -0.6830   -0.7071
    0.9659    0.2588         0
```

using pincherModel we get:

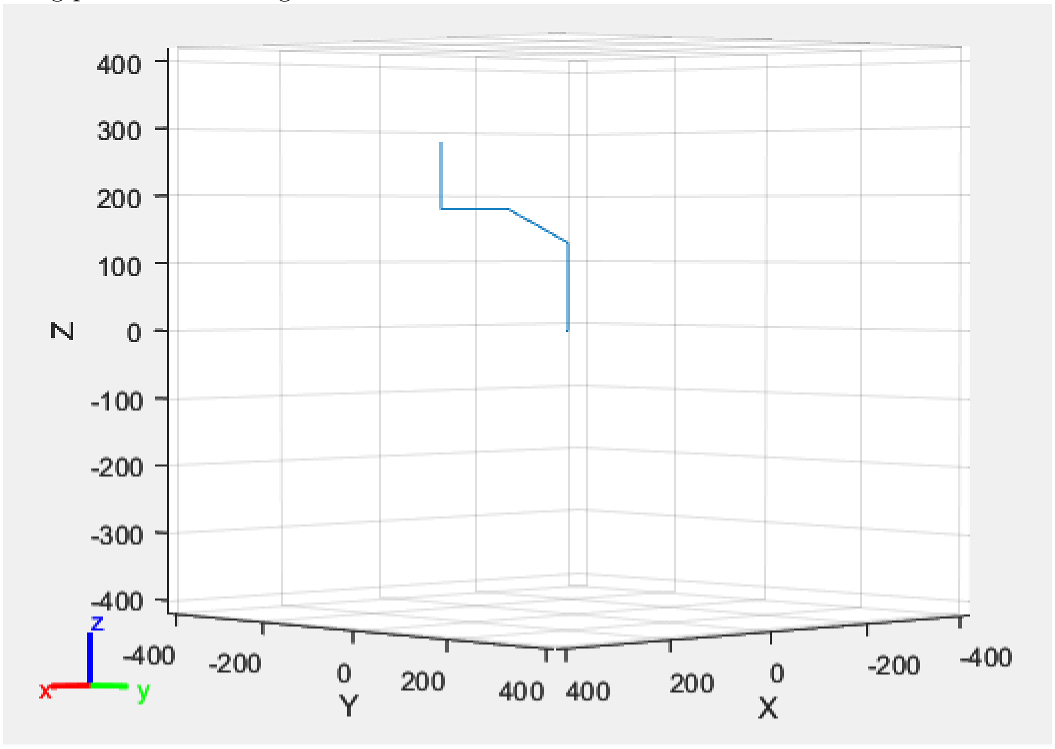


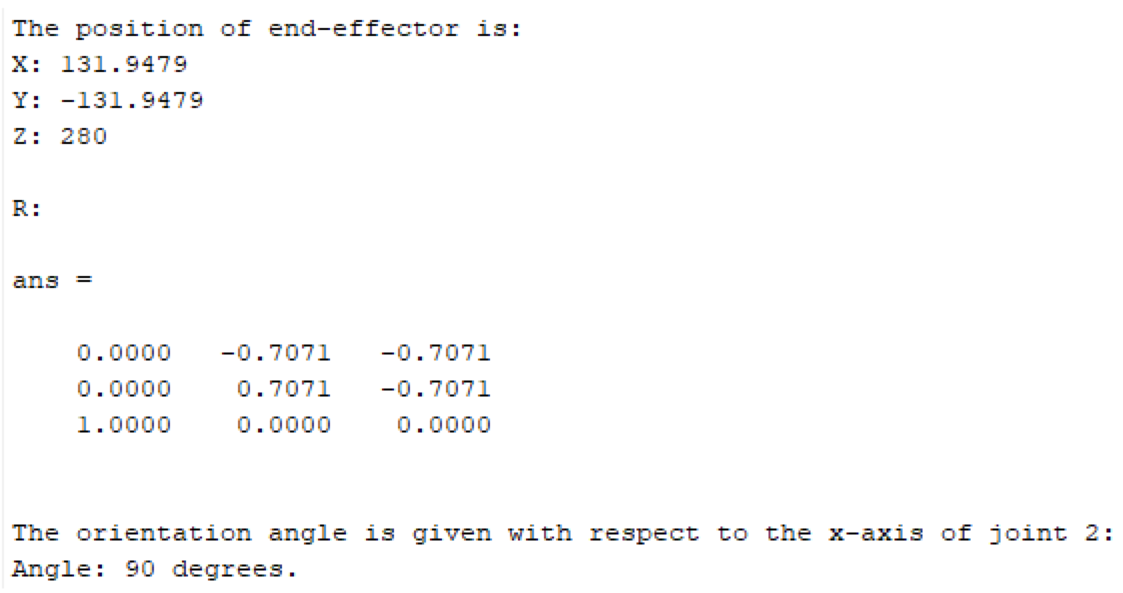
```
The position of end-effector is:
X: 136.6025
Y: 136.6025
Z: 130

R:

ans =

    0.1830   -0.6830    0.7071
    0.1830   -0.6830   -0.7071
    0.9659    0.2588    0.0000


The orientation angle is given with respect to the x-axis of joint 2:
Angle: 75 degrees.
```

Fourth configuration (-pi/4, pi/6, -pi/6, pi/2):
using findPincher we get:

```
>> [x, R] = findPincher(-pi/4, pi/6, -pi/6, pi/2)

x =

  131.9479
 -131.9479
  280.0000



R =

    0.0000   -0.7071   -0.7071
    0.0000    0.7071   -0.7071
    1.0000    0.0000         0
```

using pincherModel we get:





```
The position of end-effector is:
X: 131.9479
Y: -131.9479
Z: 280

R:

ans =

    0.0000   -0.7071   -0.7071
    0.0000    0.7071   -0.7071
    1.0000    0.0000    0.0000


The orientation angle is given with respect to the x-axis of joint 2:
Angle: 90 degrees.
```

Hence, we can see that both functions return the exact same position and orientation for different configurations i.e., if we call the functions for any four joint angle values, we will get the same transformation matrix from both of them.

6. Map the DH joint angles to the respective servomotor angles in Table 4.2 and Table 4.3. You'll have to determine (i) the possible angular shift between $\theta°$ 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 4.3.

Solution:

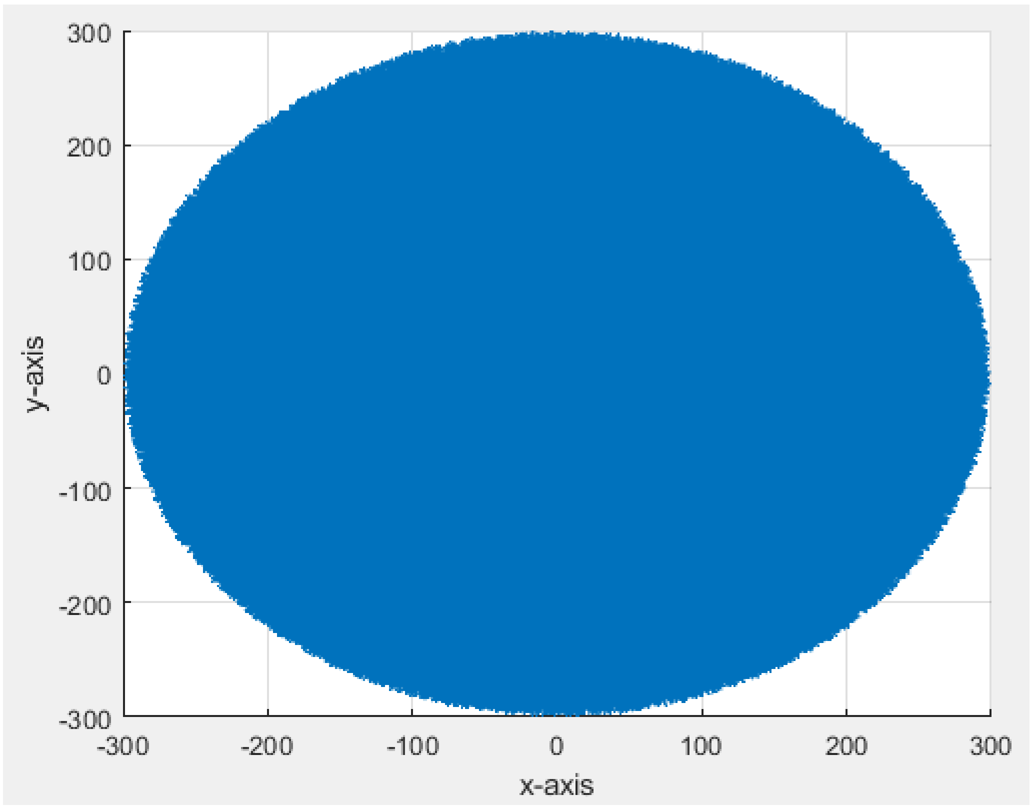
Joint ID	DH Joint Angle (θ_i)	Servo Angle ψ_i	Aligned directions of rotation (Yes/No)
1	0°	-90°	yes
2	0°	-90°	yes
3	0°	0°	yes
4	0°	0°	yes

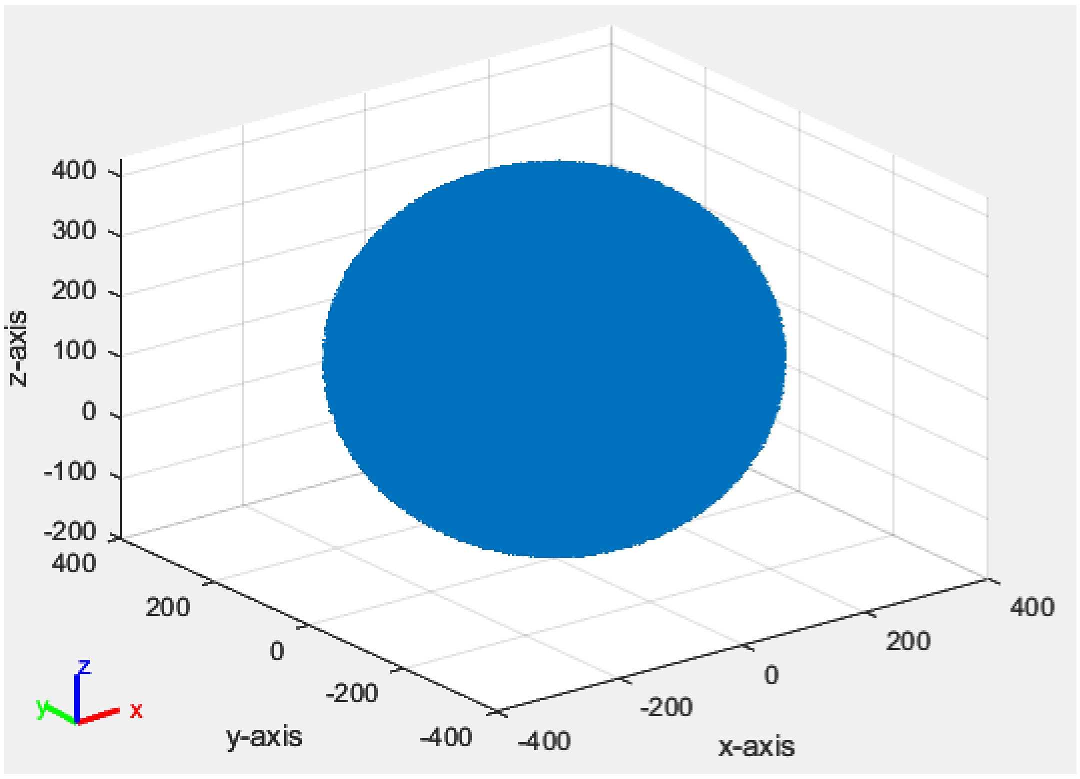
Joint ID	Minimum Joint Angle		Maximum Joint Angle	
	Servo angle	DH Joint Angle	Servo Angle	DH Joint Angle
1	$-150°$	-60°	$-150°$	240°
2	$-150°$	-60°	150°	240°
3	$-150°$	-150°	150°	150°
4	$-150°$	-150°	150°	150°

7. 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?

Solution:

```
N = 3000000;
theta_1 = -60+(240+60) * rand(N,1);
theta_2 = -60+(240+60) * rand(N,1);
theta_3 = -150+(150+150) * rand(N,1);
theta_4 = -150+(150+150) * rand(N,1);
x = zeros(N,1);
y = zeros(N,1);
z = zeros(N,1);
for i=1:N
    var = findPincher(theta_1(i), theta_2(i), theta_3(i), theta_4(i));
    x(i) = var(1);
    y(i) = var(2);
    z(i) = var(3);
end
plot3(x, y, z)
xlabel("x-axis")
ylabel("y-axis")
zlabel("z-axis")
grid on
```





As seen from the figures attached above, the maximum horizontal reach is -300mm to 300mm.

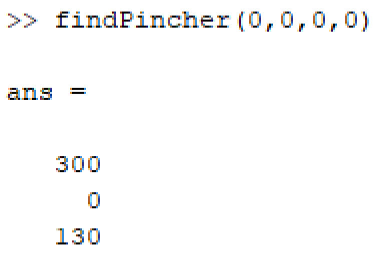
8. Provide a MATLAB function errorCode = setPosition(jointAngles) that accepts joint angles of Phantom X Pincher as argument, and sets them as goal positions for the respective motors in the arm. The function should be properly commented, especially the error codes should be explained in detail.

Solution:

```
function errorCode = setPosition(jointAngles) % the function takes vector of
    joint angle values in radians

% using the given equation to map the angles recieved to motor angles in
% range [-150, 150]
angle_1 = mod(jointAngles(1)+pi,2*pi)-pi;
angle_2 = mod(jointAngles(2)+pi,2*pi)-pi;
angle_3 = mod(jointAngles(3)+pi,2*pi)-pi;
angle_4 = mod(jointAngles(4)+pi,2*pi)-pi;

% getting the mapping function for all 4 angles according to table 4.2
a1 = angle_1 - pi/2;
a2 = angle_2 - pi/2;
a3 = angle_3;
a4 = angle_4;

% setting condition that arm only moves in the given range of [-150,150]
if (((a1 > (-5*pi)/6) && (a2 > (-5*pi)/6) && (a3 > (-5*pi)/6) && (a4 > (-5*pi)/6)
    ) && ((a1 < (5*pi)/6) && (a2 < (5*pi)/6) && (a3 < (5*pi)/6) && (a4 < (5*pi)/6)
    ))
    b = Arbotix('port', 'COM7', 'nservos', 5);
    b.setpos([a1, a2, a3, a4, 0], [55, 55, 55, 55, 55]); % getting the position
       of arm when angles are given at speed of 55 (in the range [0, 1023])
end
```

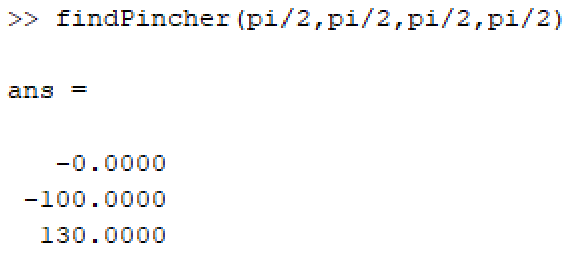
9.
- Select 5 random joint configurations for the manipulator;
- Determine the end-effector position and orientation from findPincher function;
- If it appears that the arm will self-collide in the resulting image for your configuration, obtained from findPincher, or the obtained end-effector position is too close to the baseboard (less than 95mm), then change chosen joint angles. This is a safety measure to prevent any collisions due to errors in your mappings.
- Use your setPosition function to move the end-effector to each of the five chosen configurations.
- For each configuration, physically measure and note the actual end-effector position.
- Tabulate the errors in Euclidean distance between the position computed by your software and the achieved position by the arm for all the samples. Compute the mean error.
- If there is any error, what are the possible source(s) of error?

Solution:
First configuration (0, 0, 0, 0):
using findPincher we get:



```
>> findPincher(0,0,0,0)

ans =

   300
     0
   130
```

using setPosition and measuring physically we get:
x = 310 mm
y = 0 mm
z = 135 mm
error$_1$ = $\sqrt{(310-300)^2+(0-0)^2+(135-130)^2} = 11.18$
Second configuration (pi/2, pi/2, pi/2, pi/2):
using findPincher we get:

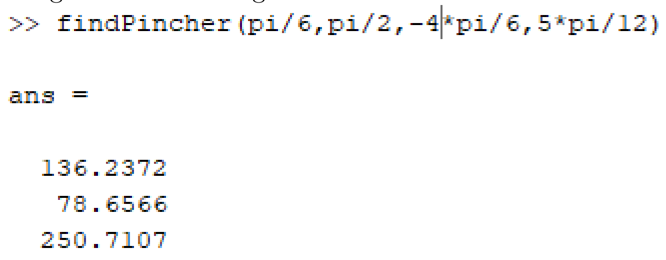


```
>> findPincher(pi/2,pi/2,pi/2,pi/2)

ans =

   -0.0000
 -100.0000
  130.0000
```

using setPosition and measuring physically we get:
x = 0 mm
y = -96 mm
z = 130 mm
error$_2$ = $\sqrt{(0-0)^2+(-96-(-100))^2+(130-130)^2} = 4$

Third configuration (pi/6, pi/2, -4*pi/6, 5*pi/12):

using findPincher we get:

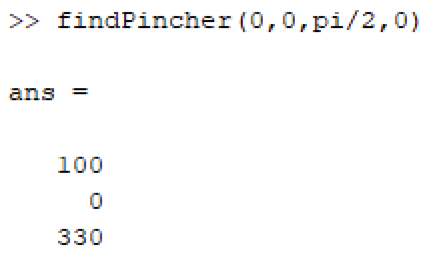


```
>> findPincher(pi/6,pi/2,-4*pi/6,5*pi/12)

ans =

  136.2372
   78.6566
  250.7107
```

using setPosition and measuring physically we get:
x = 130 mm
y = 82 mm
z = 240 mm
error$_3$ = $\sqrt{(130 - 136.2372)^2 + (82 - 78.6566)^2 + (240 - 250.7107)^2} = 12.84$

Fourth configuration (0, 0, pi/2, 0):

using findPincher we get:

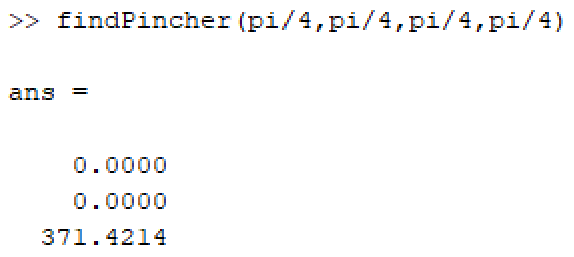


```
>> findPincher(0,0,pi/2,0)

ans =

   100
     0
   330
```

using setPosition and measuring physically we get:
x = 110 mm
y = 0 mm
z = 330 mm
error$_4$ = $\sqrt{(110 - 100)^2 + (0 - 0)^2 + (330 - 330)^2} = 10$

Fifth configuration (pi/4, pi/4, pi/4, pi/4):

using findPincher we get:



```
>> findPincher(pi/4,pi/4,pi/4,pi/4)

ans =

    0.0000
    0.0000
  371.4214
```

using setPosition and measuring physically we get:
x = 0 mm
y = 0 mm
z = 379 mm
error$_5$ = $\sqrt{(0 - 0)^2 + (0 - 0)^2 + (379 - 371.4241)^2} = 7.58$

mean error (e$_m$) = $\frac{error_1 + error_2 + error_3 + error_4 + error_5}{5}$

e$_m$ = $\frac{11.18 + 4 + 12.84 + 10 + 7.58}{5}$

e$_m$ = 9.12

There is an error which may have been due to human error while physically measuring the distances.