Project Report for "*Medical Robotics*"

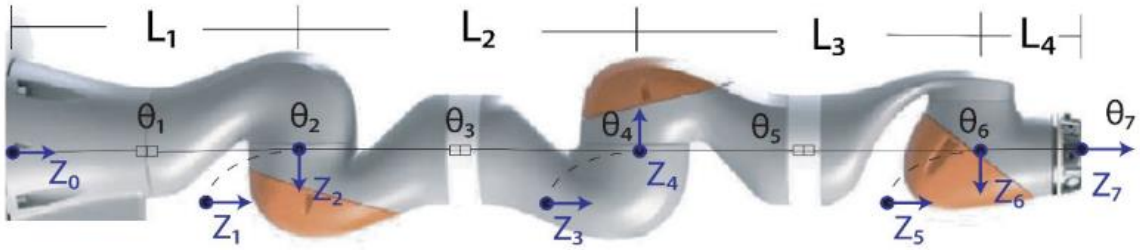
Biomedical Engineering MSc

By: *Agapi Konstantina Liontou 1057757*

The code for the project was created and executed in Matlab version 2022b.

[1]

For this project, we were assigned to deal with the robot given below:



For the above given robot-axis configuration it holds:

L1 = 360mm, L2= 420mm, L3= 400mm, L4=126mm.

$\theta 1_{min} = \theta 3_{min} = \theta 5_{min} = -\theta 1_{max} = -\theta 3_{max} = -\theta 5_{max} = 169$ degrees.

$\theta 2_{min} = \theta 4_{min} = \theta 6_{min} = -\theta 2_{max} = -\theta 4_{max} = -\theta 6_{max} = -119$ degrees.

$\theta 7_{min} = -\theta 7_{max} = -174$ degrees.

Axis X_0 points **inwards** to the screen. (A.k.a: origin of the axis is at origin of Z_0 and its direction is "away" from you).

First of all we were assigned to provide a 3-axis coordinate systems for each joint and derive the corresponding Denavit - Hartenberg parameters. In order to do that, we have to define the directions of the 3 axes (X, Y, Z) at first. We were given that Z0 axis points to the right and the X0 axis points inwards the screen, so by the right hand rule we can find the Y0 direction, which points upwards. As we can see the direction of x axis is the same for all joints, the direction of z axis is given by the provided image, so the y axis is upwards for joints 1, 3, 5, 7 and to the left for joint 4 and to the right for joint 2.

Note: we considered that z1, z2 are at the same origin, z3, z4 too and z5, z6.

So now, according to these rules:

a) d is the offset along previous z to common normal.
b) theta is the angle about previous z, from old x to new x.
c) a is the length of the common normal.
d) αlpha is the angle about the common normal, from old z-axis to the new z-axis.

We can extract the Denavit Hartenberg parameters.

Here is the code for that:

```
%% 1
% Define DH parameters for the robot
L1 = 360;
L2 = 420;
L3 = 400;
```

```
L4 = 126;

% Define joint limits in degrees
theta1_min = -169; theta1_max = 169;
theta2_min = -119; theta2_max = 119;
theta3_min = -169; theta3_max = 169;
theta4_min = -119; theta4_max = 119;
theta5_min = -169; theta5_max = 169;
theta6_min = -119; theta6_max = 119;
theta7_min = -174; theta7_max = 174;

% Define DH parameters for each joint

% Joint 1
d1=L1;
th1=0;
a1=0;
alpha1=0;
dh1 = [th1, d1, a1, alpha1];

% Joint 2
d2=0;
th2=0;
a2=0;
alpha2 = -90;
%alpha2= -pi/2;
dh2 = [th2, d2, a2, alpha2];

% Joint 3
d3=420;
th3=0;
a3=0;
alpha3 = 90;
%alpha3= pi/2;
dh3 = [th3, d3, a3, alpha3];

% Joint 4
d4=0;
th4=0;
a4=0;
alpha4 = 90;
%alpha4= pi/2;
dh4 = [th4, d4, a4, alpha4];

% Joint 5
d5=400;
th5=0;
a5=0;
alpha5 = -90;
%alpha5= -pi/2;
dh5 = [th5, d5, a5, alpha5];

% Joint 6
d6=0;
th6=0;
a6=0;
alpha6 = -90;
%alpha6= -pi/2;
dh6 = [th6, d6, a6, alpha6];

% Joint 7
d7=126;
th7=0;
a7=0;
alpha7 = 90;
%alpha7= pi/2;
dh7 = [th7, d7, a7, alpha7];

% Combine DH parameters for all joints
dh_params = [dh1; dh2; dh3; dh4; dh5; dh6; dh7];

% Display DH parameters in a table
joint_names = {'Joint', 'θ (deg)', 'd (mm)', 'a (mm)', 'α (deg)'};
joint_numbers = {'1', '2', '3', '4', '5', '6', '7'};
```

```
dh_table = table(joint_numbers', dh_params(:,1), dh_params(:,2), dh_params(:,3), dh_params(:,4),
'VariableNames', joint_names);
disp(dh_table)
```

Joint	θ (deg)	d (mm)	a (mm)	α (deg)
{'1'}	0	360	0	0
{'2'}	0	0	0	-90
{'3'}	0	420	0	90
{'4'}	0	0	0	90
{'5'}	0	400	0	-90
{'6'}	0	0	0	-90
{'7'}	0	126	0	90

The code defines the Denavit-Hartenberg (DH) parameters for a 7-DOF robotic arm. The DH parameters are used to describe the kinematics of robotic manipulators, and they define the relationship between adjacent links of the robot. The code defines the DH parameters for each joint of the robot arm, and also defines the joint limits in degrees. The DH parameters are stored in a matrix called **dh_params**, which contains the four DH parameters (θ, d, a, α) for each joint. The joint names and DH parameters are displayed in a table using the **table** function.

[2]

In this part, we were asked to compute analytically the transformation matrices for the forward kinematics problem. Here is the code for that:

```
% Compute transformation matrices
A = sym(zeros(4, 4, 7));
for i = 1:7
    theta_i = dh_params(i, 1);
    d_i = dh_params(i, 2);
    a_i = dh_params(i, 3);
    alpha_i = dh_params(i, 4);

    A(:,:,i) = [cosd(theta_i)   -sind(theta_i)*cosd(alpha_i)    sind(theta_i)*sind(alpha_i)    a_i*cosd(theta_i);
                sind(theta_i)    cosd(theta_i)*cosd(alpha_i)   -cosd(theta_i)*sind(alpha_i)    a_i*sind(theta_i);
                   0                 sind(alpha_i)                cosd(alpha_i)                 d_i;
                   0                      0                            0                          1];
end
```

```
%% 2

% Define joint angles (in radians)
theta1 = 0;
theta2 = 0;
theta3 = 0;
theta4 = 0;
theta5 = 0;
theta6 = 0;
theta7 = 0;
```

```
A0_1 = [cosd(theta1) -sind(theta1)*cosd(alpha1) sind(theta1)*sind(alpha1) a1*cosd(theta1);
        sind(theta1) cosd(theta1)*cosd(alpha1) -cosd(theta1)*sind(alpha1) a1*sind(theta1);
        0 sind(alpha1) cosd(alpha1) d1;
        0 0 0 1];

A1_2 = [cosd(theta2) -sind(theta2)*cosd(alpha2) sind(theta2)*sind(alpha2) a2*cosd(theta2);
        sind(theta2) cosd(theta2)*cosd(alpha2) -cosd(theta2)*sind(alpha2) a2*sind(theta2);
        0 sind(alpha2) cosd(alpha2) d2;
        0 0 0 1];

A2_3 = [cosd(theta3) -sind(theta3)*cosd(alpha3) sind(theta3)*sind(alpha3) a3*cosd(theta3);
        sind(theta3) cosd(theta3)*cosd(alpha3) -cosd(theta3)*sind(alpha3) a3*sind(theta3);
        0 sind(alpha3) cosd(alpha3) d3;
        0 0 0 1];

A3_4 = [cosd(theta4) -sind(theta4)*cosd(alpha4) sind(theta4)*sind(alpha4) a4*cosd(theta4);
        sind(theta4) cosd(theta4)*cosd(alpha4) -cosd(theta4)*sind(alpha4) a4*sind(theta4);
        0 sind(alpha4) cosd(alpha4) d4;
        0 0 0 1];

A4_5 = [cosd(theta5) -sind(theta5)*cosd(alpha5) sind(theta5)*sind(alpha5) a5*cosd(theta5);
        sind(theta5) cosd(theta5)*cosd(alpha5) -cosd(theta5)*sind(alpha5) a5*sind(theta5);
        0 sind(alpha5) cosd(alpha5) d5;
        0 0 0 1];

A5_6 = [cosd(theta6) -sind(theta6)*cosd(alpha6) sind(theta6)*sind(alpha6) a6*cosd(theta6);
        sind(theta6) cosd(theta6)*cosd(alpha6) -cosd(theta6)*sind(alpha6) a6*sind(theta6);
        0 sind(alpha6) cosd(alpha6) d6;
        0 0 0 1];

A6_7 = [cosd(theta7) -sind(theta7)*cosd(alpha7) sind(theta7)*sind(alpha7) a7*cosd(theta7);
        sind(theta7) cosd(theta7)*cosd(alpha7) -cosd(theta7)*sind(alpha7) a7*sind(theta7);
        0 sind(alpha7) cosd(alpha7) d7;
        0 0 0 1];
```

The transformation matrices describe the relationship between the coordinate frames of each joint in the arm.

In the code we first create an empty symbolic array **A** with dimensions 4 x 4 x 7, which will hold the transformation matrices for each joint. Then, for each joint in the arm, the code extracts the corresponding Denavit-Hartenberg (DH) parameters (θ, d, a, α) from the **dh_params** matrix, which

describes the physical dimensions and geometry of the arm. Then I use these DH parameters to construct a 4 x 4 transf matrix for each joint using trigonometric functions such as **sind()** and **cosd()**. The resulting transformation matrices are then stored in the **A** array.

In the second part of the code, the joint angles (θ1 to θ7) are defined, and separate transformation matrices are computed for each joint using the values of the joint angles and the DH parameters for that joint. These transformation matrices are named A0_1, A1_2, A2_3, and so on up to A6_7.

Note: actually here I do it twice, with a different way, but the result is exactly the same.

So, the transformation matrices are these:

```
val(:,:,1) =

[1, 0, 0,   0]
[0, 1, 0,   0]
[0, 0, 1, 360]
[0, 0, 0,   1]


val(:,:,2) =

[1,  0, 0, 0]
[0,  0, 1, 0]
[0, -1, 0, 0]
[0,  0, 0, 1]


val(:,:,3) =

[1, 0,  0,   0]
[0, 0, -1,   0]
[0, 1,  0, 420]
[0, 0,  0,   1]
```

```
val(:,:,4) =

[1, 0,  0, 0]
[0, 0, -1, 0]
[0, 1,  0, 0]
[0, 0,  0, 1]


val(:,:,5) =

[1,  0, 0,   0]
[0,  0, 1,   0]
[0, -1, 0, 400]
[0,  0, 0,   1]


val(:,:,6) =

[1,  0, 0, 0]
[0,  0, 1, 0]
[0, -1, 0, 0]
[0,  0, 0, 1]
```

```
val(:,:,7) =

[1, 0,  0,   0]
[0, 0, -1,   0]
[0, 1,  0, 126]
[0, 0,  0,   1]
```

[3]

In this part we were asked to compute analytically the transformation matrix T_0_7. To do that, we multiplied all the individual transformation matrices that we calculated before.

```
%% 3
% Overall transformation matrix from base to end-effector
T07 = A0_1 * A1_2 * A2_3 * A3_4 * A4_5 * A5_6 * A6_7;
```

And the result is this:

```
T07 =

     1     0     0     0
     0     1     0   146
     0     0     1   360
     0     0     0     1
```

In general, the resulting matrix T07 can be used to calculate the position and orientation of the end-effector of the robot arm in space, relative to the base of the arm. Here, I have to mention that T07 transformation matrix is orthonormal.

[4]

In this part we were asked to solve the inverse kinematics problem, which was a quite tough procedure. Here is the code:

```
%% 4

syms theta1 theta2 theta3 theta4 theta5 theta6 theta7
syms r1 r2 r3 r4 r5 r6 r7
syms v1 v2 v3 v4 v5 v6 v7
syms u11 u12 u13 u14 u21 u22 u23 u24 u31 u32 u33 u34
syms al1 al2 al3 al4 al5 al6 al7

theta1=0;
A0_1 = [cos(theta1) -sin(theta1)*cosd(al1) sin(theta1)*sind(al1) v1*cos(theta1);
        sin(theta1) cos(theta1)*cosd(al1) -cos(theta1)*sind(al1) v1*sin(theta1);
        0 sind(al1) cosd(al1) r1;
        0 0 0 1];

A1_2 = [cos(theta2) -sin(theta2)*cosd(al2) sin(theta2)*sind(al2) v2*cos(theta2);
        sin(theta2) cos(theta2)*cosd(al2) -cos(theta2)*sind(al2) v2*sin(theta2);
        0 sind(al2) cosd(al2) r2;
        0 0 0 1];

A2_3 = [cos(theta3) -sin(theta3)*cosd(al3) sin(theta3)*sind(al3) v3*cos(theta3);
        sin(theta3) cos(theta3)*cosd(al3) -cos(theta3)*sind(al3) v3*sin(theta3);
        0 sind(al3) cosd(al3) r3;
        0 0 0 1];

A3_4 = [cos(theta4) -sin(theta4)*cosd(al4) sin(theta4)*sind(al4) v4*cos(theta4);
        sin(theta4) cos(theta4)*cosd(al4) -cos(theta4)*sind(al4) v4*sin(theta4);
        0 sind(al4) cosd(al4) r4;
        0 0 0 1];

A4_5 = [cos(theta5) -sin(theta5)*cosd(al5) sin(theta5)*sind(al5) v5*cos(theta5);
        sin(theta5) cos(theta5)*cosd(al5) -cos(theta5)*sind(al5) v5*sin(theta5);
        0 sind(al5) cosd(al5) r5;
        0 0 0 1];
```

```
A5_6 = [cos(theta6) -sin(theta6)*cosd(al6) sin(theta6)*sind(al6) v6*cos(theta6);
        sin(theta6) cos(theta6)*cosd(al6) -cos(theta6)*sind(al6) v6*sin(theta6);
        0 sind(al6) cosd(al6) r6;
        0 0 0 1];

A6_7 = [cos(theta7) -sind(theta7)*cosd(al7) sin(theta7)*sind(al7) v7*cos(theta7);
        sin(theta7) cos(theta7)*cosd(al7) -cos(theta7)*sind(al7) v7*sin(theta7);
        0 sind(al7) cosd(al7) r7;
        0 0 0 1];

A0_1_new = subs(A0_1, [al1 v1 r1], [alpha1 a1 d1]);
A1_2_new = subs(A1_2, [al2 v2 r2], [alpha2 a2 d2]);
A2_3_new = subs(A2_3, [al3 v3 r3], [alpha3 a3 d3]);
A3_4_new = subs(A3_4, [al4 v4 r4], [alpha4 a4 d4]);
A4_5_new = subs(A4_5, [al5 v5 r5], [alpha5 a5 d5]);
A5_6_new = subs(A5_6, [al6 v6 r6], [alpha6 a6 d6]);
A6_7_new = subs(A6_7, [al7 v7 r7], [alpha7 a7 d7]);

A0_4i = A0_1_new * A1_2_new * A2_3_new * A3_4_new;
A4_7i = A4_5_new * A5_6_new * A6_7_new;

C = [u11 u12 u13 u14
     u21 u22 u23 u24
     u31 u32 u33 u34
     0 0 0 1];
```

In this code firstly, I define symbolic variables and create transformation matrices. Later, the symbolic variables are defined using the "syms" function in MATLAB, which allows them to be used in mathematical expressions without explicitly assigning values to them.

Again, I create seven homogeneous transformation matrices (A0_1 to A6_7) using the Denavit-Hartenberg (DH) convention. These matrices represent the transformation between adjacent coordinate systems along the manipulator, from the base to the end effector. Then I compute the overall transformation matrices from the base to joint 4 (A0_4i) and from joint 5 to the end effector (A4_7i) by multiplying the individual transformation matrices.

The main idea was to follow the examples given in the lectures. So, I had to divide my robot in 2 parts, a rotational one and a translational one. Moreover in order to simplify the problem I defined theta1 as a zero angle and I tried to solve the whole problem parametrically (that's why I used syms). So for the rotational part, I have to calculate the transf matrices A4_5_new, A5_6_new, A6_7_new by substituting the angle theta with a theta parameter like in the lectures. This was difficult to be done in maltab, so I did handwritten. For this, we start with -cos(theta6) = u32 and by Euler's equation we can find θ6. All angles can be found that way and it is very important to be very careful in our calculations to be sure that the angle limits are respected.

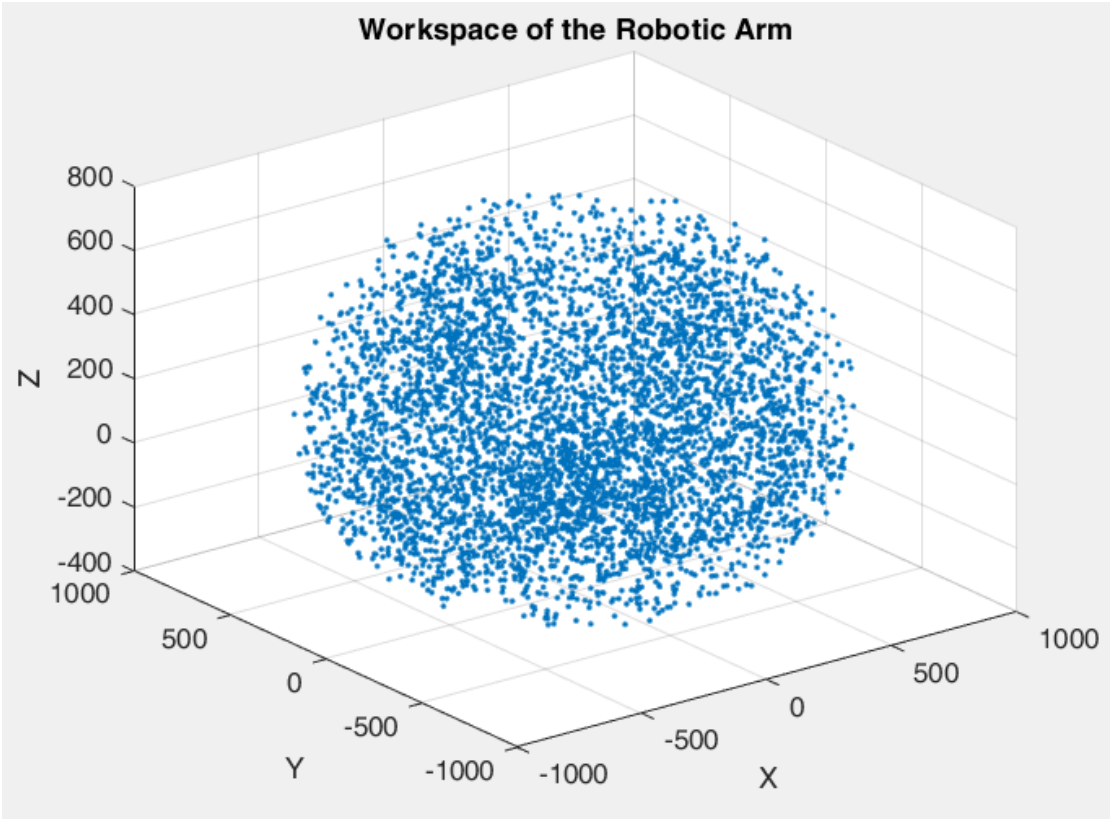
[5]

In this part we were asked to compute in a computational manner the workspace of the robot. Here is the code:

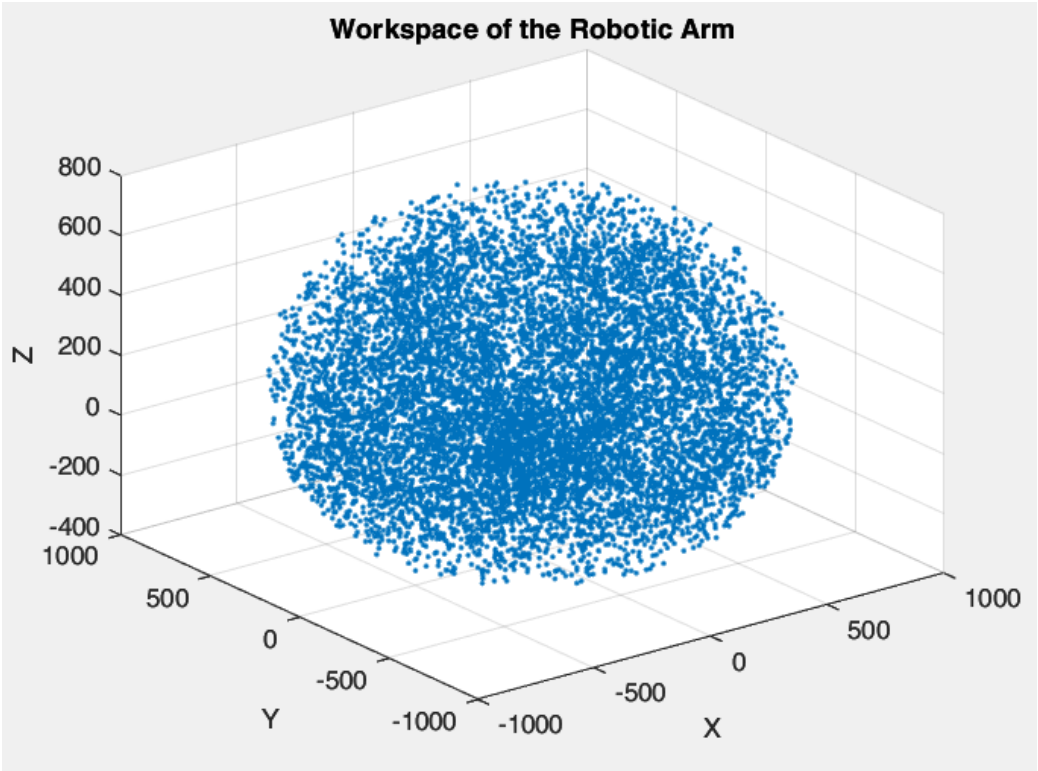
```
%% 5

N = 10000;
```

```
k1 = theta1_max + (theta1_max - theta1_min)*rand(N,1);
k2 = theta2_max + (theta2_max - theta2_min)*rand(N,1);
k3 = theta3_max + (theta3_max - theta3_min)*rand(N,1);
k4 = theta4_max + (theta4_max - theta4_min)*rand(N,1);
k5 = theta5_max + (theta5_max - theta5_min)*rand(N,1);
k6 = theta6_max + (theta6_max - theta6_min)*rand(N,1);
k7 = theta7_max + (theta7_max - theta7_min)*rand(N,1);

X = [];
Y = [];
Z = [];

for i = 1:N
    A1 = transf_matrix(k1(i),a1, d1, alpha1);
    A2 = transf_matrix(k2(i),a2, d2, alpha2);
    A3 = transf_matrix(k3(i),a3, d3, alpha3);
    A4 = transf_matrix(k4(i),a4, d4, alpha4);
    A5 = transf_matrix(k5(i),a5, d5, alpha5);
    A6 = transf_matrix(k6(i),a6, d6, alpha6);
    A7 = transf_matrix(k7(i),a7, d7, alpha7);

    T = A1 * A2 * A3 * A4 * A5 * A6 * A7;

    X = [X, T(1,4)];
    Y = [Y, T(2,4)];
    Z = [Z, T(3,4)];
end

scatter3(X,Y,Z,'.')
title('Workspace of the Robotic Arm');
xlabel('X');
ylabel('Y');
zlabel('Z');

% function for transformation matrix
function [ T ] = transf_matrix(theta, a, d, alpha)
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
```

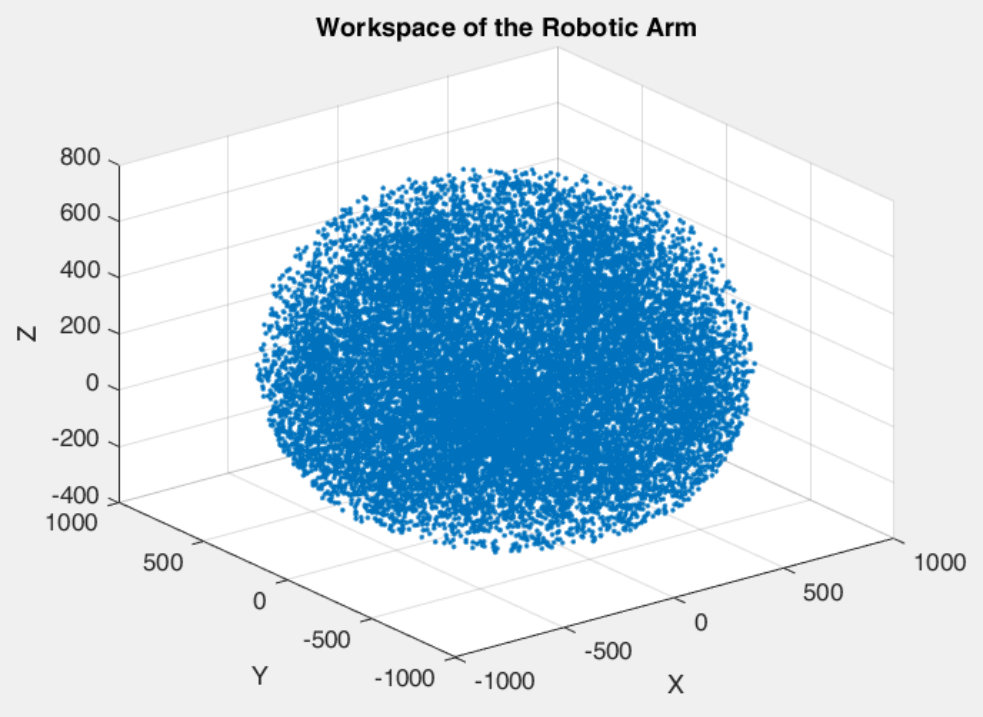
Here, we generate a workspace plot for the robot. The variables k1 to k7 are randomly generated joint angles for each degree of freedom within their specified maximum and minimum limits (theta1_min, theta1_max, ..., theta7_min, theta7_max). The for loop iterates through each of the N randomly generated joint angles, calculates the corresponding transformation matrix for each joint angle using the function transf_matrix, and then multiplies all transformation matrices together to obtain the end-effector transformation matrix T. The X, Y, and Z coordinates of the end-effector position are extracted from T and stored in the arrays X, Y, and Z respectively. Finally, the scatter3 function is used to plot the X, Y, and Z arrays as a 3D scatter plot, which represents the reachable workspace of the robotic arm. Here are the plots made by different values of N:



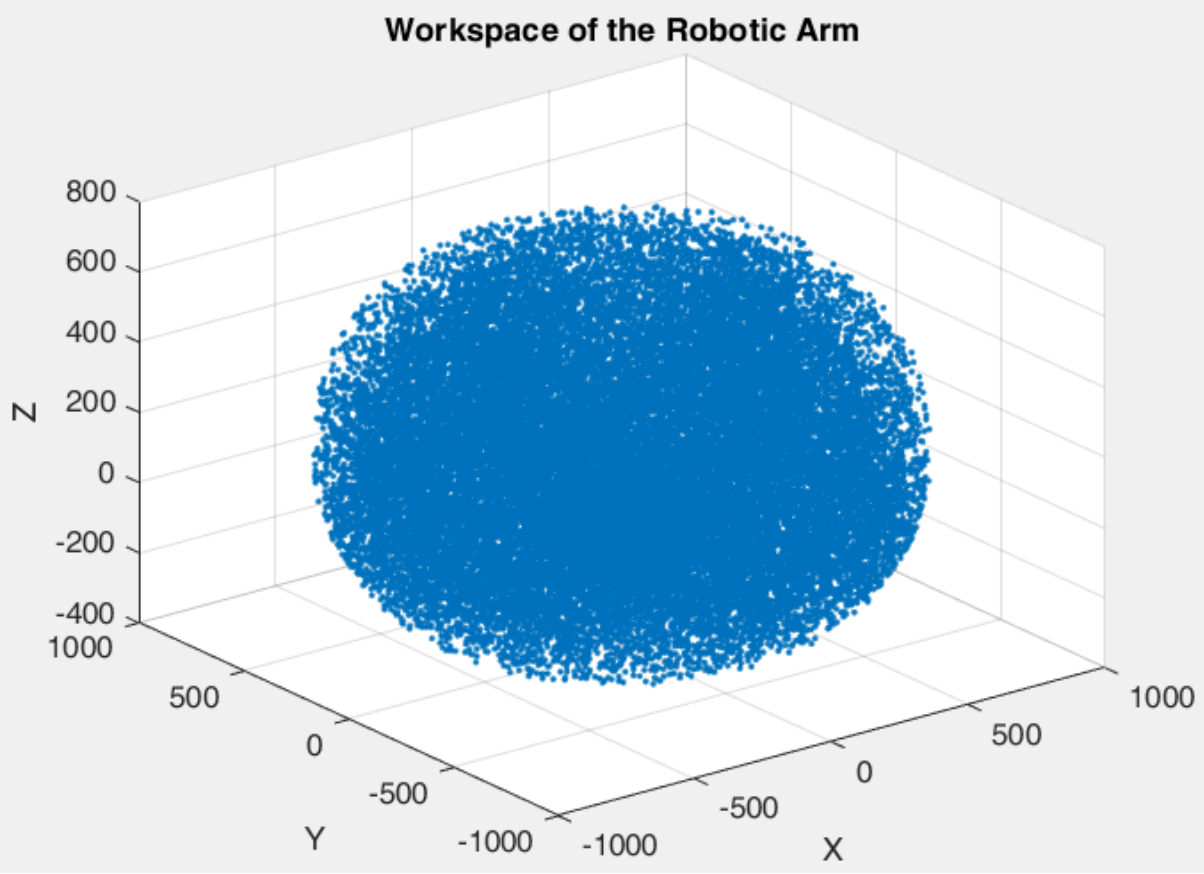
N=5.000



N=10.000



N=20.000



N=35.000