Queen's University

Electrical and Computer Engineering

MREN 348

Assignment 2

Kay Burnham ID 20220414

Due Feb 17th, 2025

Question 1.

See Appendix I for MATLAB code.

Output:

i.

```
>> Sph_Direct(mu_1, mu_2, d_3);
    0.1422   -0.9703    0.1957    1.6424
    0.5703    0.2419    0.7850    6.5872
   -0.8090         0    0.5878    1.8809
         0         0         0    1.0000
```

ii.

```
>> SCARA_Direct(mu_1, mu_2, d_3, mu_4);
    0.9063   -0.4226         0    3.4345
    0.4226    0.9063         0    0.7576
         0         0    1.0000   14.0000
         0         0         0    1.0000
```

iii.

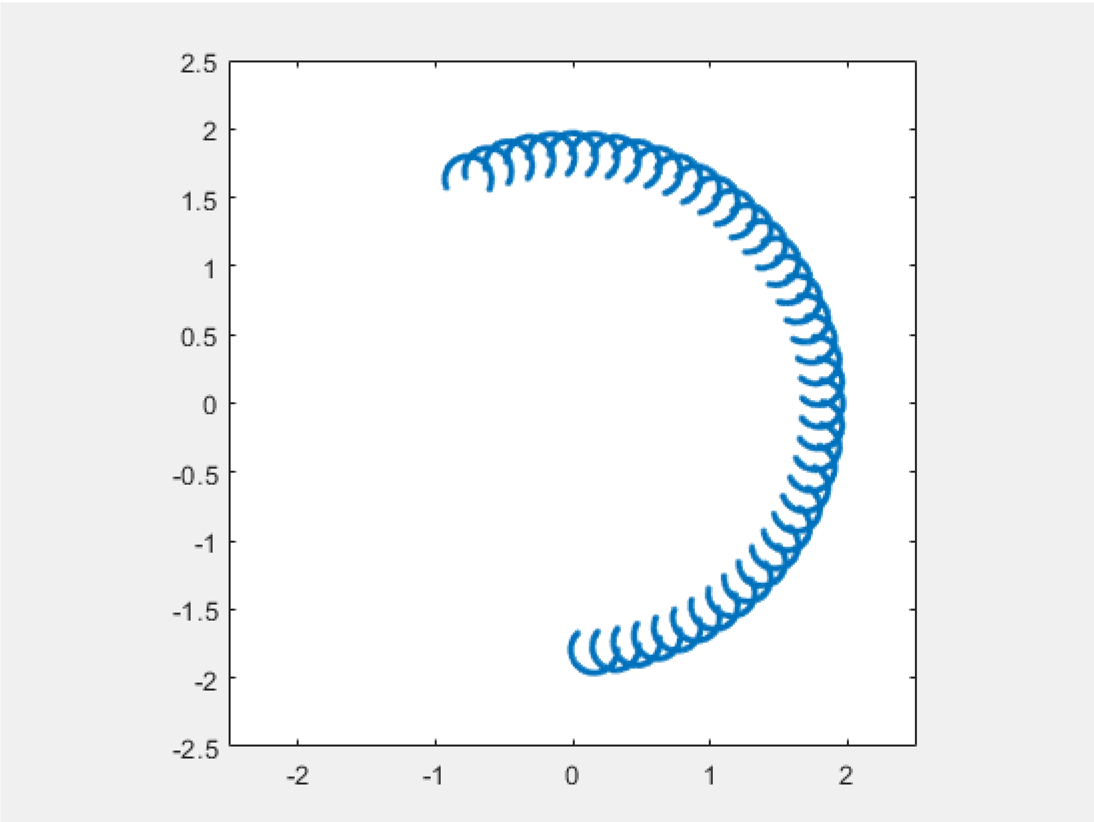
```
>> Cyl_Direct(mu_1, d_2, d_3);
    0.9854   -0.1702         0    3.0942
    0.1702    0.9854         0    0.5345
         0         0    1.0000    2.7200
         0         0         0    1.0000
```

Question 2.

See Appendix II for MATLAB code.

Printed output:

See below for a plot of all points reachable by the SCARA manipulator.

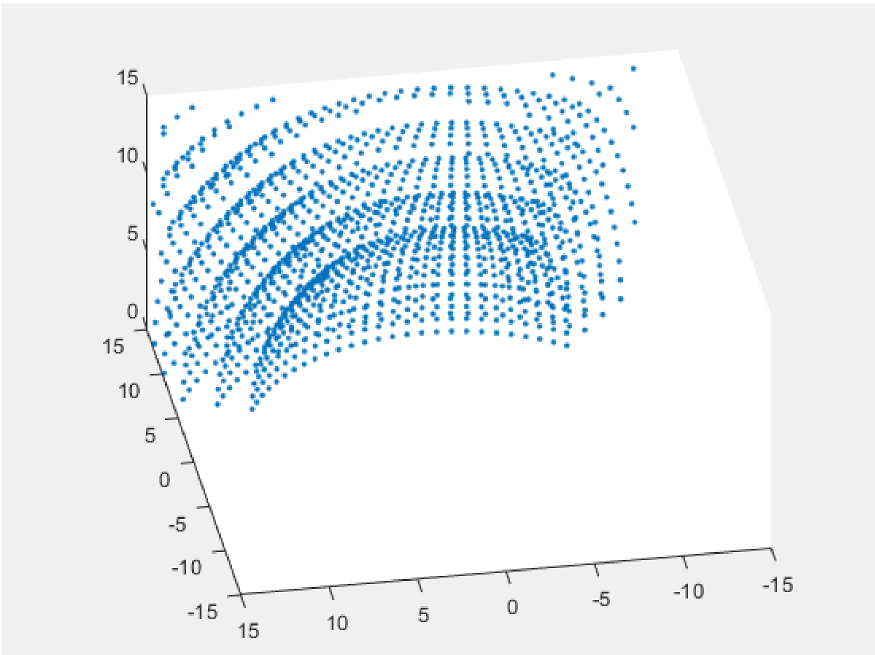


Question 3.

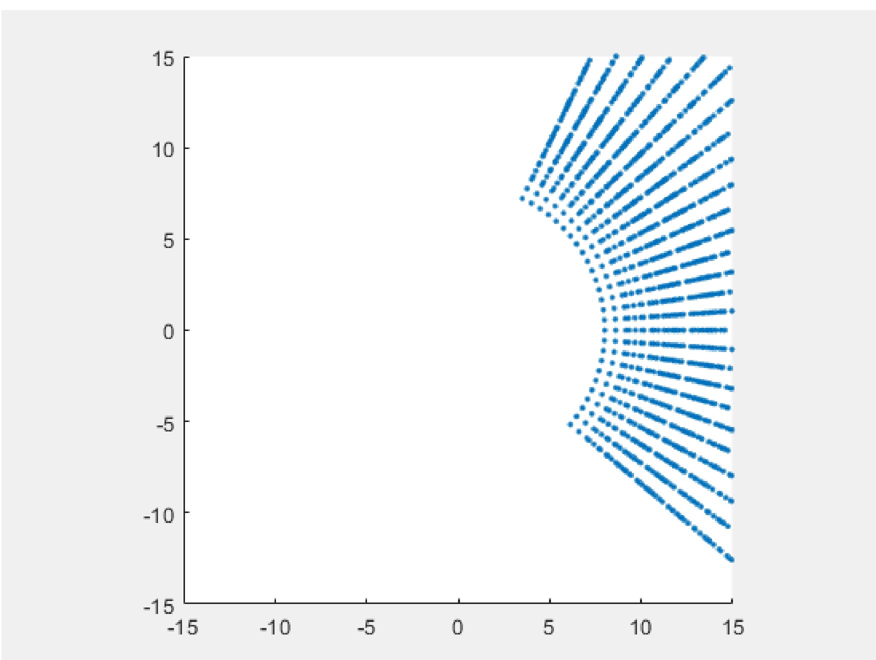
See Appendix II for MATLAB code.

Plot output:

i. View from Azimuth = -100deg, Elevation = 30



ii. View from Azimuth = 0deg, Elevation = 90



Appendix I.

Code for forward kinematic solutions of three robots (spherical arm, SCARA manipulator, cylindrical arm)

```
function Sph_Direct(mu_1, mu_2, d_3)

% assume inputs are in deg, deg, metres

% hardcoded link offset:

d_2 = 4.2;


%convert input deg to radians

mu_1 = deg2rad(mu_1);

mu_2 = deg2rad(mu_2);


% H. Transform matrices

A01 = [cos(mu_1), -sin(mu_1), 0, d_2*cos(mu_1);

       sin(mu_1),  cos(mu_1), 0, d_2*sin(mu_1);

       0,          0,          1, 0;

       0,          0,          0, 1];


A12 = [ cos(mu_2), 0, sin(mu_2), 0;

        0,         1,   0,        0;

       -sin(mu_2), 0, cos(mu_2), 0;

        0,         0, 0,         1];


A23 = [1, 0, 0, 0;

       0, 1, 0, 0;

       0, 0, 1, d_3;

       0, 0, 0, 1];


% Final transform

A03 = A01 * A12 * A23;

disp(A03);

end
```

```
function SCARA_Direct(mu_1, mu_2, d_3, mu_4)
% assume inputs are in deg, deg, metres, deg
% note: mu_4 doesn't influence location, only end effector orientation

% hardcoded link lengths:
a_1 = 3.4;
a_2 = 0.12;

%convert input deg to radians
mu_1 = deg2rad(mu_1);
mu_2 = deg2rad(mu_2);

% H. Transform matrices
A01 = [cos(mu_1), -sin(mu_1), 0, a_1*cos(mu_1);
       sin(mu_1),  cos(mu_1), 0, a_1*sin(mu_1);
       0,          0,         1, 0;
       0,          0,         0, 1];

A12 = [cos(mu_2), -sin(mu_2), 0, a_2*cos(mu_2);
       sin(mu_2),  cos(mu_2), 0, a_2*sin(mu_2);
       0,          0,         1, 0;
       0,          0,         0, 1];

A23 = [1, 0, 0, 0;
       0, 1, 0, 0;
       0, 0, 1, d_3;
       0, 0, 0, 1];

% Final transform
A03 = A01 * A12 * A23;
disp(A03);
end
```

```
function Cyl_Direct(mu_1, d_2, d_3)
% assume inputs are in deg, metres, metres

%convert input deg to radians
mu_1 = deg2rad(mu_1);

% H. Transform matrices
A01 = [cos(mu_1), -sin(mu_1), 0, 0;
       sin(mu_1),  cos(mu_1), 0, 0;
       0,          0,         1, 0;
       0,          0,         0, 1];

A12 = [1, 0, 0, d_2;
       0, 1, 0, 0;
       0, 0, 1, 0;
       0, 0, 0, 1];

A23 = [1, 0, 0, 0;
       0, 1, 0, 0;
       0, 0, 1, d_3;
       0, 0, 0, 1];

% Final transform
A03 = A01 * A12 * A23;
disp(A03);
end
```

Appendix II A.

Code for 2D graphing of SCARA manipulator.

```
function SCARA_2D
x_end = [];
y_end = [];
pause on;

for i = -85:5:115
    for j = -140:5:85
        % calculate new position
        [x1, y1, x2, y2, z_end] = SCARA_step(i, j);
        x_end(end+1) = x2;
        y_end(end+1) = y2;
    end
end

%draw figure
plot(x_end, y_end, '.');
axis equal
axis([-2.5 2.5 -2.5 2.5])
end
```

```
function [x1, y1, x2, y2, z_end] = SCARA_step(mu_1, mu_2)
% calculates positions of joints at given inputs
% hardcoded link lengths and set angles:
a_1 = 1.8;
a_2 = 0.17;
d_3 = 0;
mu_4 = 0;

% joint limits:
mu_1_min =  -85; mu_1_max = 115; % Degrees
mu_2_min = -140; mu_2_max = 85;  % Degrees

mu_1 = max(min(mu_1, mu_1_max), mu_1_min);
mu_2 = max(min(mu_2, mu_2_max), mu_2_min);

mu_1 = deg2rad(mu_1);
mu_2 = deg2rad(mu_2);

% joint positions:
x1 = a_1 * cos(mu_1);
y1 = a_1 * sin(mu_1);

x2 = x1 + a_2 * cos(mu_1 + mu_2);
y2 = y1 + a_2 * sin(mu_1 + mu_2);

z_end = d_3; % end effector position
end
```

Appendix II B.

Code for 2D graphing of cylindrical arm.

```
function Sph_2D

x_end = [];

y_end = [];

z_end = [];

pause on;


for i = -80:4:65

    for j = -40:4:50

        for k = 3:2:16

            % calculate new position

            [x1, y1, x2, y2, z] = Sph_step(i, j, k);

            x_end(end+1) = x2;

            y_end(end+1) = y2;

            z_end(end+1) = z;

        end

    end

end


% draw figure

plot3(x_end, y_end, z_end, '.');

axis equal

axis([-15 15 -15 15 -0 15])

% change variables below for different azimuth, elevation

view(-100, 30);

end
```

```
function [x1, y1, x2, y2, z] = Sph_step(mu_1, mu_2, d_3)
% calculates positions of joints at given inputs

% hardcoded link length: 2nd link fixed
d_2 = 9;

% joint limits:
mu_1_min = -40; mu_1_max = 65; % Degrees
mu_2_min = -40; mu_2_max = 50;  % Degrees
d_3_min =    3; d_3_max =  16; % cm

mu_1 = max(min(mu_1, mu_1_max), mu_1_min);
mu_2 = max(min(mu_2, mu_2_max), mu_2_min);
d_3 = max(min(d_3, d_3_max), d_3_min);

mu_1 = deg2rad(mu_1);
mu_2 = deg2rad(mu_2);
d_total = d_2 + d_3;

% joint positions: base position is fixed
x1 = 0;
y1 = 0;

x2 = d_total * cos(mu_1) * cos(mu_2);
y2 = d_total * sin(mu_1) * cos(mu_2);
z = d_total * sin(mu_2); % end effector position
end
```