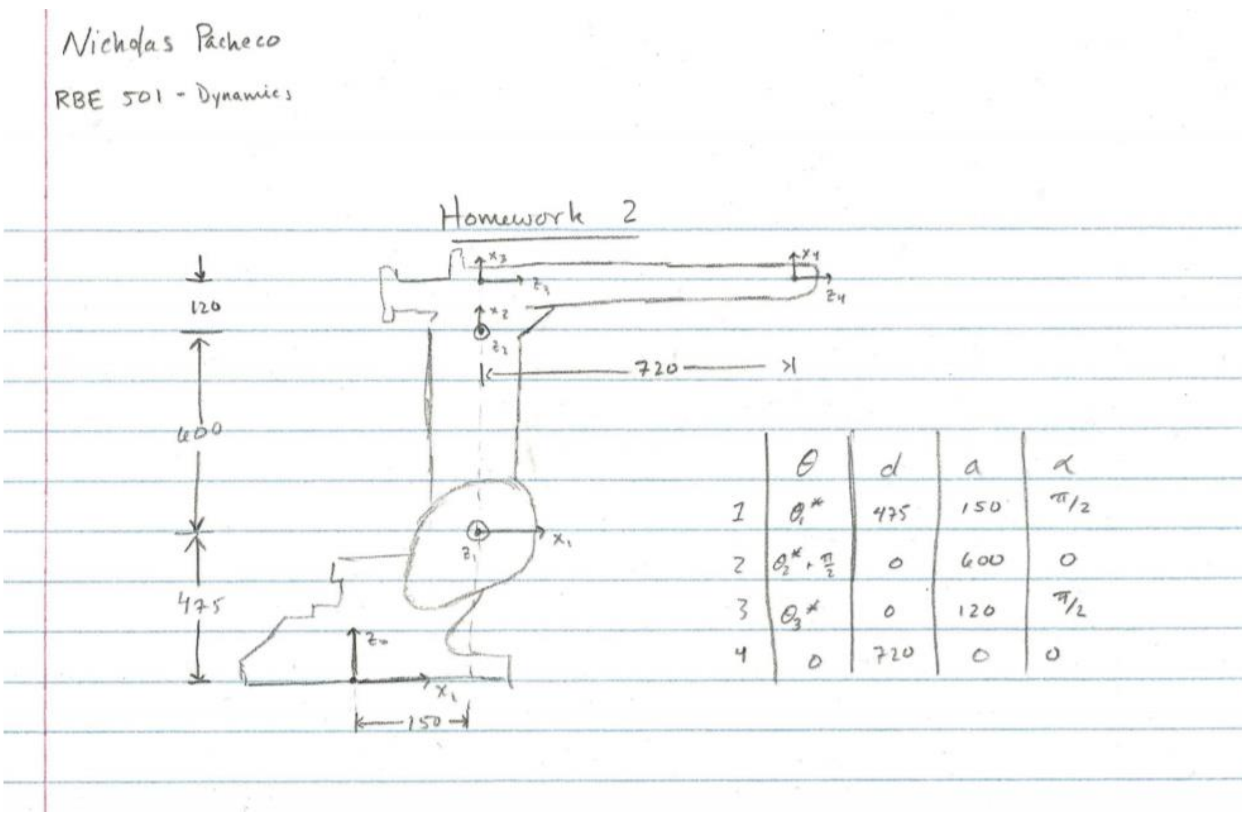
RBE 501 Homework 2

```
clc; clear; close all
```



Forward kinematics of robot without spherical wrist

```
syms t1 t2 t3
L1y = 475; L1x = 150; L2 = 600; L3 = 120; L4 = 720;
DH_table = [t1 L1y L1x sym(pi)/2;
    t2+sym(pi/2) 0 L2 0;
```

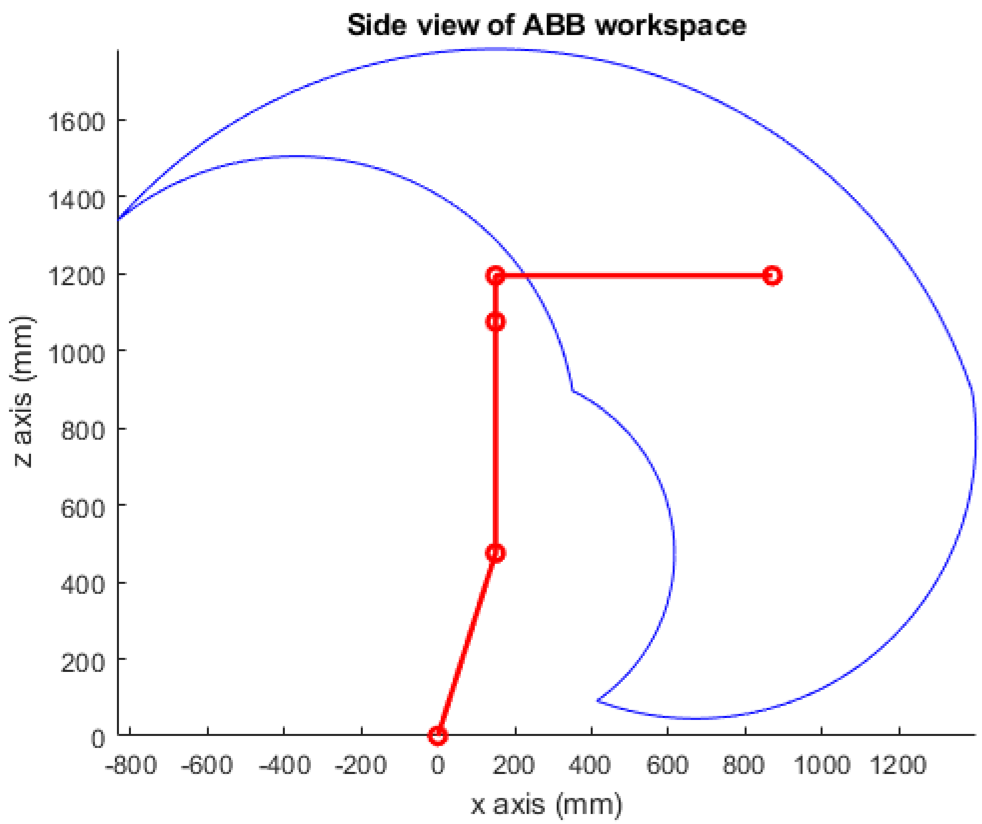
```
    t3 0 L3 sym(pi/2);
    0 L4 0 0];
T01 = tdh(DH_table(1,:));
T12 = tdh(DH_table(2,:));
T23 = tdh(DH_table(3,:)); % needs this dummy transform to get to tip
T3tip = tdh(DH_table(4,:));
T0tip = simplify(T01*T12*T23*T3tip,'Steps',10);

T02 = T01*T12;
```

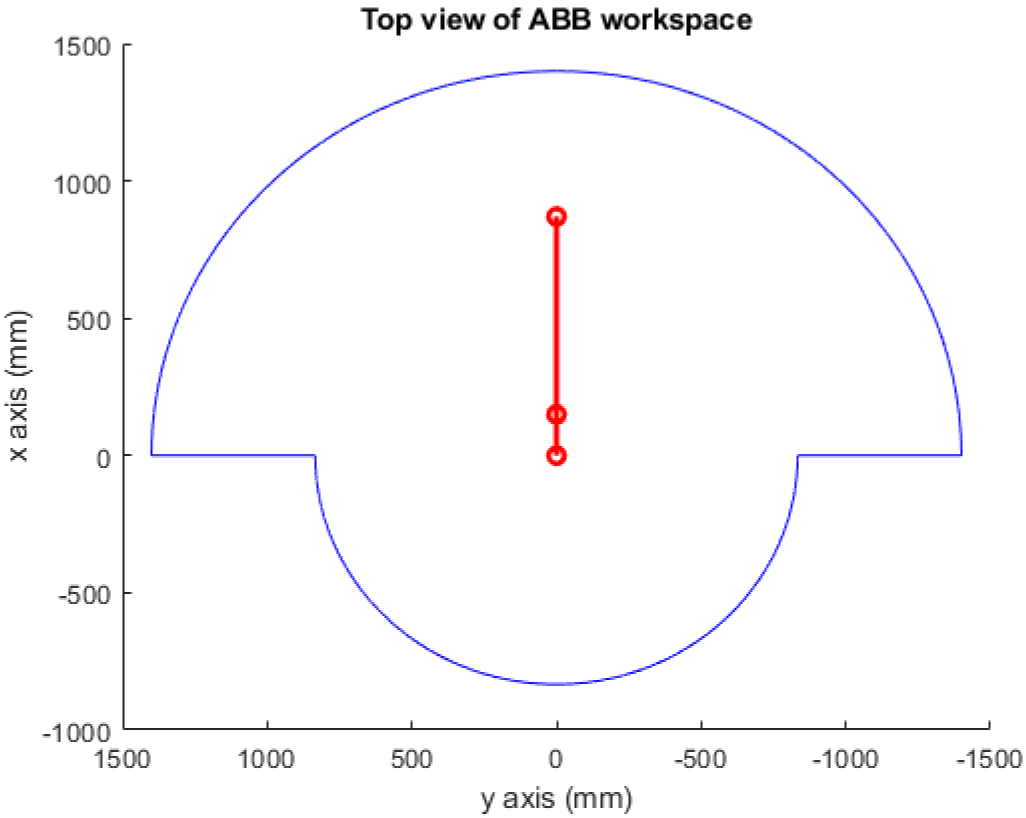
A. Generating workspace

```
%generated above
% this is the forward kinematics without a spherical wrist
T04 = @(t1,t2,t3)[ -sin(t2 + t3)*cos(t1),  sin(t1), cos(t2 + t3)*cos(t1), 30*cos(t1)*(24*cos(t2 +
t3) - 4*sin(t2 + t3) - 20*sin(t2) + 5);
    -sin(t2 + t3)*sin(t1), -cos(t1), cos(t2 + t3)*sin(t1), 30*sin(t1)*(24*cos(t2 + t3) - 4*sin(t2
+ t3) - 20*sin(t2) + 5);
    cos(t2 + t3),        0,         sin(t2 + t3),        600*cos(t2) + 120*37^(1/2)*cos(t2 + t3 -
atan(6)) + 475;
    0,        0,                    0,
1];

joint_min = -60*pi/180; % min for joint 2 and 3
joint_max = 60*pi/180; % max value for joint 2 and 3
% startig values for each joint
joint1 = 0;
joint2 = -60*pi/180;
joint3 = -60*pi/180;
x = [];
z = [];
% move joint 3 from min to max
for i = linspace(joint_min,joint_max,1000)
    joint3 = i;
    T_actual = T04(joint1,joint2,joint3);
    x = [x, T_actual(1,4)];
    z = [z, T_actual(3,4)];
end
% move joint 2 from min to max
for i = linspace(joint_min,joint_max,1000)
    joint2 = i;
    T_actual = T04(joint1,joint2,joint3);
    x = [x, T_actual(1,4)];
    z = [z, T_actual(3,4)];
end
% move joint 3 from max to min
for i = linspace(joint_max,joint_min,1000)
    joint3 = i;
    T_actual = T04(joint1,joint2,joint3);
    x = [x, T_actual(1,4)];
    z = [z, T_actual(3,4)];
end
```

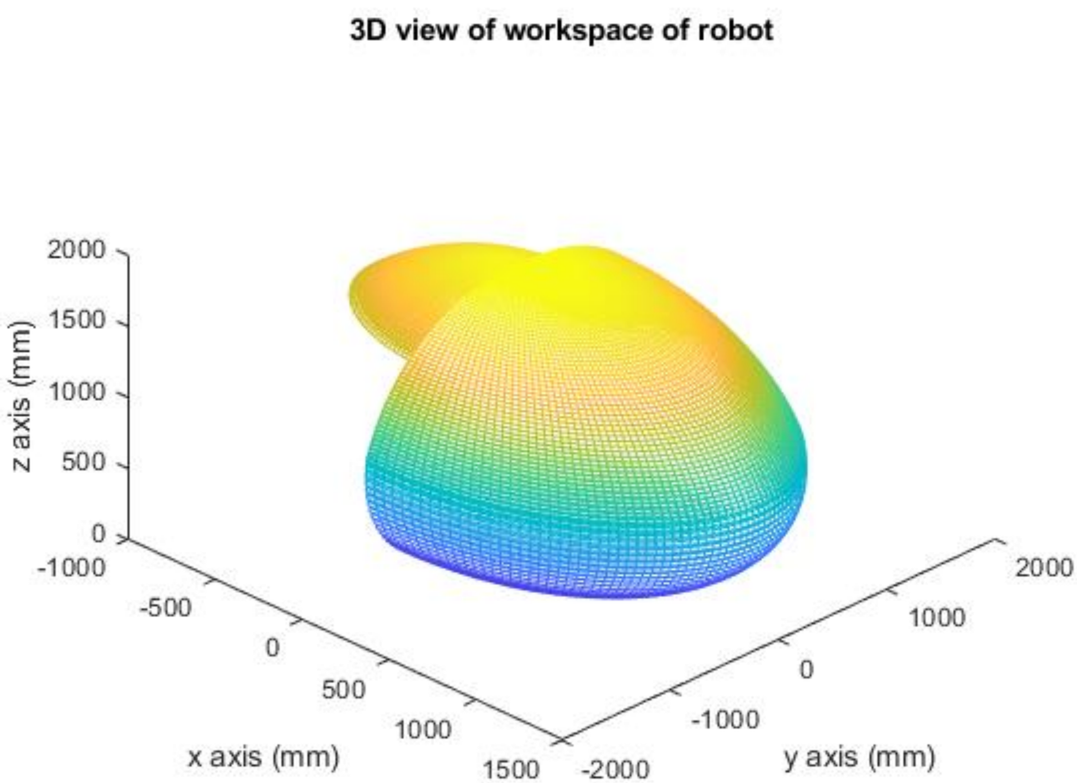
```
% move joint 2 from max to min
for i = linspace(joint_max,joint_min,1000)
    joint2 = i;
    T_actual = T04(joint1,joint2,joint3);
    x = [x, T_actual(1,4)];
    z = [z, T_actual(3,4)];
end

% plotting side view
fig_side_view = figure(1);
plot3(x,zeros(1000*4,1),z,'Color','b')
hold on
axis equal
title("Side view of ABB workspace")
xlabel("x axis (mm)");
zlabel("z axis (mm)");
plot_robot([0,0,0]);
view(0,0);
hold off

% Plotting top view
fig_top_view = figure(2);
hold on
plot_robot([0,0,0]);
x_min = min(x);
x_max = max(x);
theta = linspace(-pi/2,pi/2,4000);
title("Top view of ABB workspace")
xlabel("x axis (mm)");
ylabel("y axis (mm)");
% Arc created by the front of the robots workspace
plot3(cos(theta)*x_max,sin(theta)*x_max,zeros(4000,1),'Color','b');
% arc created from back of robots worksapces
plot3(cos(theta)*x_min,sin(theta)*x_min,zeros(4000,1),'Color','b');
% connectig the workspace on the right
plot3(zeros(2,1),linspace(x_max,-x_min,2),zeros(2,1),'Color','b');
% conneting the workspace on the left
plot3(zeros(2,1),linspace(-x_max,x_min,2),zeros(2,1),'Color','b');
view(-90,90);
hold off

% Plotting 3D view
fig_3d = figure(3);
title("3D view of workspace of robot");
t1=linspace(-90,90,90)*pi/180;
t2=linspace(-60,60,90)*pi/180;
t3=linspace(-60,60,90)*pi/180;
[T1,T2,T3]=meshgrid(t1,t2,t3); %
% x y and z equations from homogenous transform matrix
xM = 40.*cos(T1).*(18.*cos(T2 + T3) - 3.*sin(T2 + T3) - 15.*sin(T2) + 4);
yM = 40.*sin(T1).*(18.*cos(T2 + T3) - 3.*sin(T2 + T3) - 15.*sin(T2) + 4);
zM = 600.*cos(T2) + 120.*37.^(1/2).*cos(T2 + T3 - atan(6)) + 475;
hold on
```

```
for i = 1:30
    mesh(xM(:,:,i*3),yM(:,:,i*3),zM(:,:,i*3))
end
plot_robot([0,0,0]);
xlabel("x axis (mm)");
ylabel("y axis (mm)");
zlabel("z axis (mm)");
view(45,45);
snapnow
view(-45,45);
snapnow
hold off
```



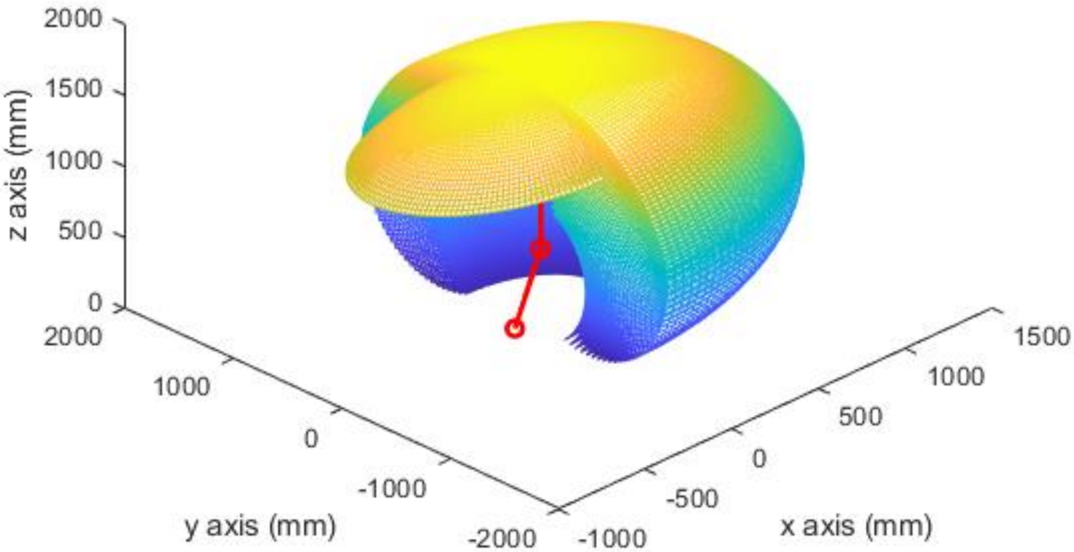


Top view of ABB workspace
1500
1000
500
0
-500
-1000
x axis (mm)
1500
1000
500
0
-500
-1000
-1500
y axis (mm)



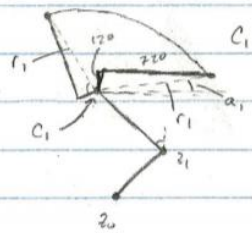
3D view of workspace of robot
2000
1500
1000
500
0
z axis (mm)
-1000
-500
0
500
1000
1500
x axis (mm)
-2000
-1000
0
1000
2000
y axis (mm)

3D view of workspace of robot



Boundary Equations: All numbers are in MATLAB

①



$r_1 = \|T_0^4(0,60^\circ,-60^\circ) - T_0^2(0^\circ,60^\circ,-60^\circ)\| = \sqrt{120^2+720^2}$

$C_1 = T_0^2(0^\circ,60^\circ,-60^\circ)$

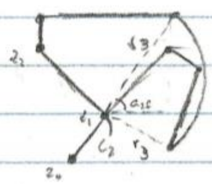
$a_1 = \frac{\pi}{2} - \mathrm{atan2}(720,120)$

$a_{1f} = 120\cdot\frac{\pi}{180} + a_1$

$r_1^2 = (x-C_{1x})^2 + (z-C_{1z})^2$

$\begin{cases} x = r\cos(\theta) + C_{1x} \\ z = r\sin(\theta) + C_{1z} \end{cases}$ for $a_1 \le \theta \le a_{1f}$

②



$r_3 = \|T_0^4(0,60^\circ,-60^\circ) - T_0^1(0,60^\circ,-60^\circ)\|$

$C_2 = T_0^1(0,60^\circ,-60^\circ)$

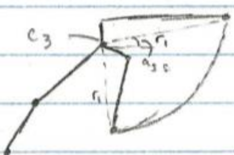
$a_{2f} = \frac{\pi}{2} - \mathrm{atan2}(r_{3x}, r_{3z})$

$a_2 = a_{2f} - 120\cdot\frac{\pi}{180}$

$r_3^2 = (x-C_{2x})^2 + (z-C_{2z})^2$

$\begin{cases} x = r_3\cos\theta + C_{2x} \\ z = r_3\sin\theta + C_{2z} \end{cases}$ for $a_2 \le \theta \le a_{2f}$

③



$r_1 = \sqrt{120^2+720^2}$

$C_3 = T_0^2(0,-60^\circ,-60^\circ)$

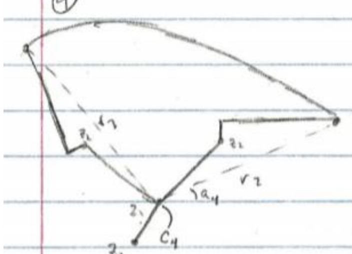
$a_{3f} = \frac{\pi}{2} - \mathrm{atan}(720,120)$

$a_3 = a_{3f} - \frac{120\cdot\pi}{180}$

$r_1^2 = (x-C_{3x})^2 + (z-C_{3z})^2$

$\begin{cases} x = r_1\cos\theta + C_{3x} \\ z = r_1\sin\theta + C_{3z} \end{cases}$ for $a_3 \le \theta \le a_{3f}$

④



$r_2 = \|T_0^4(0^\circ,-60^\circ,60^\circ) - T_0^1(0^\circ,-60^\circ,60^\circ)\|$

$C_4 = T_0^1(0^\circ,-60^\circ,60^\circ)$

$a_4 = \mathrm{atan}_2(r_z, r_x)$

$a_{4f} = a_4 + \frac{120\cdot\pi}{180}$

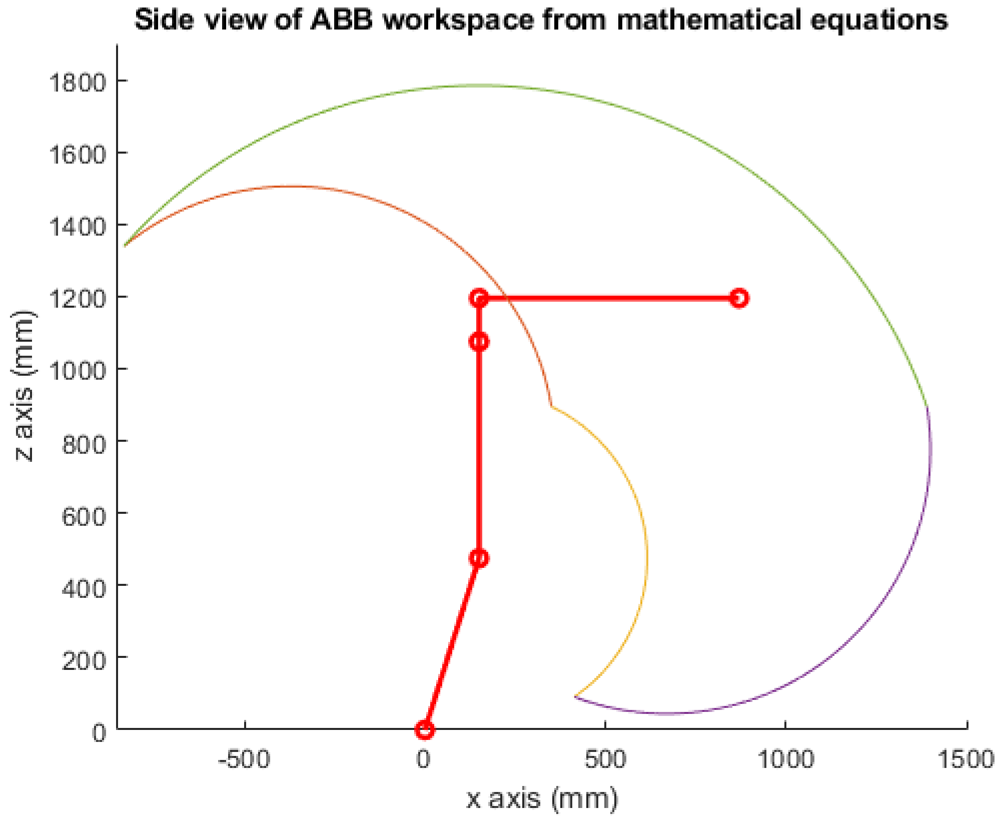
$r_2^2 = (x-C_{4x})^2 + (z-C_{4z})^2$

$\begin{cases} x = r\cos\theta + C_{4x} \\ z = r\sin\theta + C_{4z} \end{cases}$ for $a_4 \le \theta \le a_{4f}$

A. Mathematical representation of workspace

```
syms t1 t2 t3
% radius of first arc caused by end effector and joint 2
r1 = sqrt(120^2 + 720^2);
% position of end effector with joint 3 at 60 deg
T04_2 = T04(0,-60*pi/180,60*pi/180);
% position of end effector when joint 2 and joint 3 are - 60 deg
T04_3 = T04(0,-60*pi/180,-60*pi/180);
% position of joint 1 when t1 = 0 deg
```

```
T01_val = subs(T01,t1,0);
% radius of second arc caused by end effector and joint 1
r2 = norm(T04_2(1:3,4) - T01_val(1:3,4));
% radius of third arc caused by end effector and joint 1
r3 = norm(T04_3(1:3,4) - T01_val(1:3,4));

T02_val1 = subs(T02,[t1,t2],[0,60*pi/180]);
T02_val2 = subs(T02,[t1,t2],[0,-60*pi/180]);

C1  = T02_val1(1:3,4);
C2 = T01_val(1:3,4);
C3 = C2;
C4 = T02_val2(1:3,4);
% start angle for first circle
a1 = pi/2 - atan2(720,120);
% end angle for second circle
T04_a = T04(0, 60*pi/180, -60*pi/180);
a2f = pi/2 - atan2(T04_a(1,4)-T01_val(1,4),T04_a(3,4)-T01_val(3,4));
% angle for end of third circle
a3f = pi/2 - atan2(720,120);
% start angle for fourth circel
r2_vec = T04_2(1:3,4) - T01_val(1:3,4);
a4 = atan2(r2_vec(3), r2_vec(1));

equation_side_view = figure(4);
hold on
axis equal
title("Side view of ABB workspace from mathematical equations")
xlabel("x axis (mm)");
zlabel("z axis (mm)");
plot_robot([0,0,0]);
% circle 1
circle1 = circle(C1(1),C1(3),r1,[a1,a1+120*pi/180])
% circle 2
circle2 = circle(C2(1),C2(3),r3,[a2f-120*pi/180,a2f]);
% circle 3
circle3 = circle(C4(1),C4(3),r1,[a3f-120*pi/180,a3f]);
% circle 4
circle4 = circle(C3(1),C3(3),r2,[a4,a4+120*pi/180]);
hold off
view(0,0)
xlim([-850,1500]);
ylim([-1,1]);
zlim([0,1900]);
```

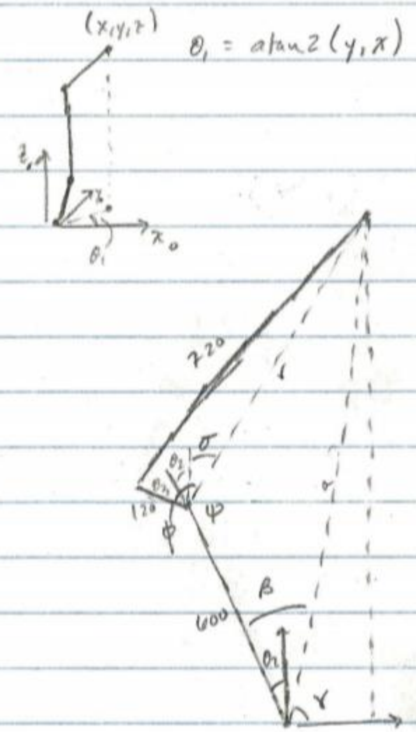

Side view of ABB workspace from mathematical equations
z axis (mm)
x axis (mm)
1800
1600
1400
1200
1000
800
600
400
200
0
-500
0
500
1000
1500

Ikin : Geometric Approach

$\theta_1 = atan2(y, x)$

$d = \sqrt{x^2+y^2} - 150$, $L_1 = 475$

$(d, z - L_1)$



$r = \sqrt{720^2 + 120^2}$

$a = \sqrt{d^2 + (z-L_1)^2}$

$\gamma = atan2(z-L_1, d)$

$F = \cos(\beta) = \frac{a^2+600^2-r^2}{2a600}$

$\beta = atan2(\sqrt{1-F^2}, F)$

$\boxed{\theta_2 = \beta + \gamma - \pi/2}$

$\phi = atan2(720, 120)$

$G = \cos(\psi) = \frac{r^2+600^2-a^2}{2r600}$

$\psi = atan2(\sqrt{1-G^2}, G)$

$\sigma = \pi - \theta_2 - \psi$

$\boxed{\theta_3 = \phi - \sigma - \theta_2}$

Ikin Orientation is in MATLAB

B. C. Inverse Kinematics Decoupling

Generate FWDKin for wrist assuming its a point

```
Twrist = @(t4,t5,t6) tdh([t4,0,0,-pi/2])*tdh([t5,0,0,pi/2])*tdh([t6,0,0,0]);
desired = [1 0 0 500; 0 1 0 100; 0 0 1 1500; 0 0 0 1];
% run inverse kinematics solved geometrically
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[th1, th2, th3, th4, th5, th6] = ikin(desired)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check to make sure output is correct by using forward kinematics
T = T04(th1,th2,th3)*Twrist(th4,th5,th6);
T(1:3,1:3)
T(1:3,4)
```

```
th1 =

    0.1974


th2 =

    0.3500


th3 =

    0.1691


th4 =

     0


th5 =

    1.0517


th6 =

    2.9442


ans =

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


ans =

   1.0e+03 *

    0.5000
    0.1000
    1.5000
```

D. Deriving the Jacobian Symbolically

```
syms t1 t2 t3
DH_table = [t1 475 150 sym(pi)/2;
    t2+sym(pi)/2 0 600 0;
    t3 0 120 sym(pi)/2;
    0 720 0 0];
T01 = tdh(DH_table(1,:));
```

```
T12 = tdh(DH_table(2,:));
T23 = tdh(DH_table(3,:));
T3tip = tdh(DH_table(4,:));
T0tip = simplify(T01*T12*T23*T3tip,'Steps',10);

T02 = T01*T12;
pe = T0tip(1:3,4);
% using the method of cross products
Jv = [cross([0;0;1],pe) cross(T01(1:3,3),pe-T01(1:3,4))...
    cross(T02(1:3,3),pe-T02(1:3,4))];
% geometric jacobian
Jw = [[0;0;1] T01(1:3,3) T02(1:3,3)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
J = [simplify(Jv,'Steps',10);Jw]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pretty(J)
```

```
J =

[ -30*sin(t1)*(24*cos(t2 + t3) - 4*sin(t2 + t3) - 20*sin(t2) + 5), -cos(t1)*(600*cos(t2) +
120*37^(1/2)*cos(t2 + t3 - atan(6))), -120*37^(1/2)*cos(t2 + t3 - atan(6))*cos(t1)]
[  30*cos(t1)*(24*cos(t2 + t3) - 4*sin(t2 + t3) - 20*sin(t2) + 5), -sin(t1)*(600*cos(t2) +
120*37^(1/2)*cos(t2 + t3 - atan(6))), -120*37^(1/2)*cos(t2 + t3 - atan(6))*sin(t1)]
[                                                              0,          120*37^(1/2)*cos(t2 +
t3 + atan(1/6)) - 600*sin(t2),        120*37^(1/2)*cos(t2 + t3 + atan(1/6))]
[                                                              0,
sin(t1),                                             sin(t1)]
[                                                              0,
-cos(t1),                                            -cos(t1)]
[                                                              1,
0,                                                   0]


/ -sin(t1) #1 30,    -cos(t1) #3,   -120 sqrt(37) #4 cos(t1) \
|                                                            |
|  cos(t1) #1 30,    -sin(t1) #3,   -120 sqrt(37) #4 sin(t1) |
|                                                            |
|        0,       #2 - 600 sin(t2),           #2             |
|                                                            |
|        0,            sin(t1),              sin(t1)         |
|                                                            |
|        0,           -cos(t1),             -cos(t1)         |
|                                                            |
\        1,                0,                    0           /
```

```
where

   #1 == 24 cos(t2 + t3) - 4 sin(t2 + t3) - 20 sin(t2) + 5

                      /               / 1 \ \
   #2 == 120 sqrt(37) cos| t2 + t3 + atan| - | |
                      \               \ 6 / /

   #3 == 600 cos(t2) + 120 sqrt(37) #4

   #4 == cos(t2 + t3 - atan(6))
```

E. Calculating Joint velocities given end effector velocities

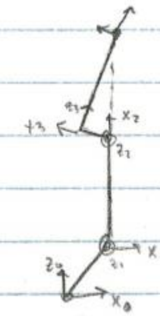
```
x_vel = [5 5 10 0 0 0]';
J_val = subs(J,[t1, t2, t3],[th1, th2, th3]);
% must take pseudo inverse because its not square
J_pinv = pinv(J_val);
joint_vel = J_pinv*x_vel;
% making the joint velocities easier to print
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
joint_vel = vpa(joint_vel,5)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
joint_vel =

 0.0076923
 -0.019196
  0.029893
```

Singularities:

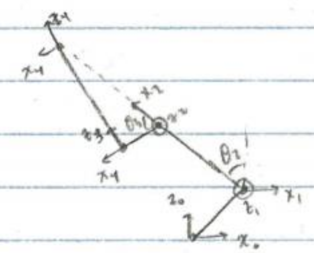
Type 1: End effector in line with joint 2 & 3

This will cause the robot tip position to lose a direction of movement



At this point $\theta_3 = atan2(720, 120) = 80.5377°$

this is out of the workspace of the robot so is not a singularity the robot can enter



Proof in MATLAB

Type 2: End Effector is over base link

When the tip of the robot is above joint 1, moving joint 1 will have no affect on the position of the end effector.

Proof in MATLAB

At Both singular configurations the rank of the Jacobian will be <3. Again though, the first singularity type is not achievable because of joint limitations. Additionally, the determinant will be ϕ at both types of singularities

F. Determining Singularities

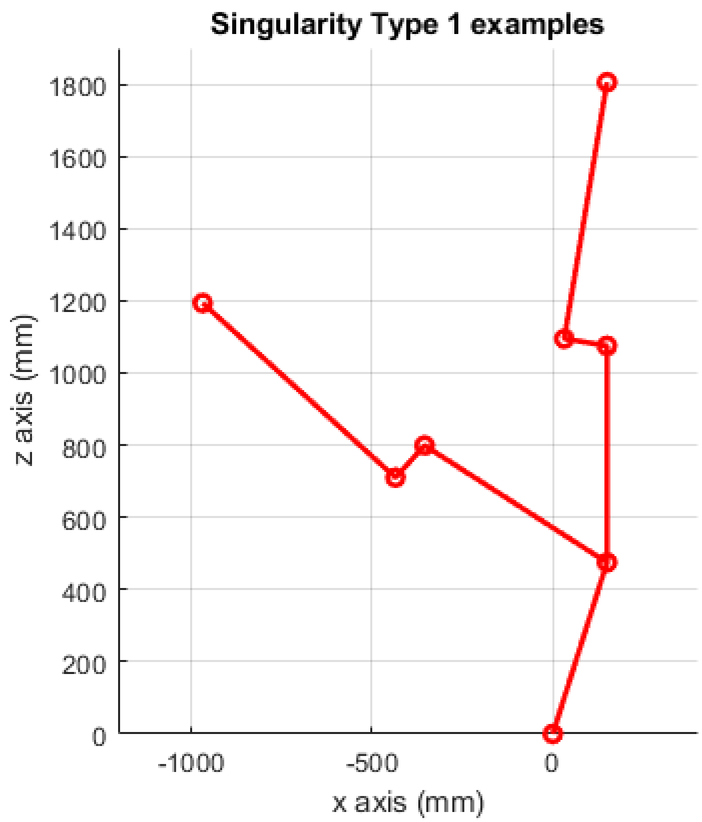
There are two main locations where a singularity will occur. The first location for a singularity is when the tip of the robot creates a line with the second and third joints. This would remove a degree of freedom for the robot since both joints will move the robot in the same direction. The second location for singularities is when the tip of the robot is directly above joint 1. Any rotation about the base frame would not cause the tip to change position.

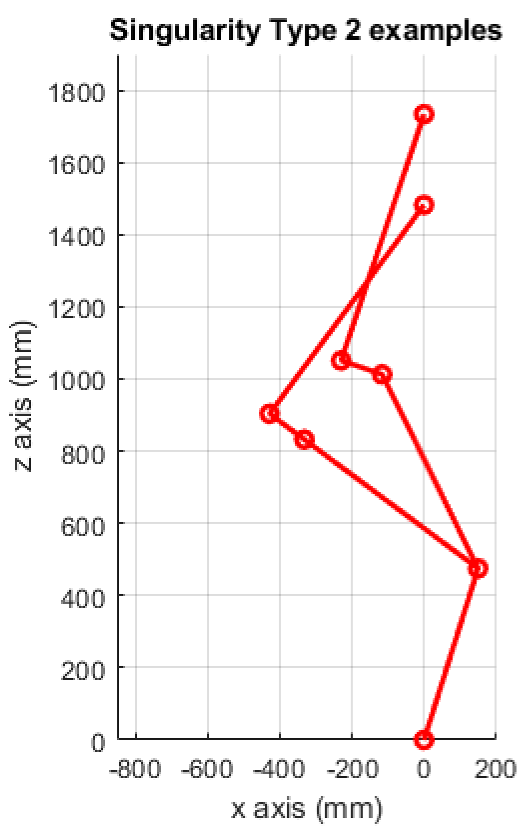
```
syms t1 t2 t3 real
min = -60*pi/180;
max = 60*pi/180;
assumeAlso(t2 > min & t2 < max);
assumeAlso(t3 > min & t3 < max);
Jv_val = @(t1,t2,t3)[ -30*sin(t1)*(24*cos(t2 + t3) - 4*sin(t2 + t3) - 20*sin(t2) + 5), -
cos(t1)*(600*cos(t2) + 120*37^(1/2)*cos(t2 + t3 - atan(6))), -120*37^(1/2)*cos(t2 + t3 -
atan(6))*cos(t1);
    30*cos(t1)*(24*cos(t2 + t3) - 4*sin(t2 + t3) - 20*sin(t2) + 5), -sin(t1)*(600*cos(t2) +
120*37^(1/2)*cos(t2 + t3 - atan(6))), -120*37^(1/2)*cos(t2 + t3 - atan(6))*sin(t1);
    0,          120*37^(1/2)*cos(t2 + t3 + atan(1/6)) - 600*sin(t2),        120*37^(1/2)*cos(t2 +
t3 + atan(1/6))];
Jv_det = simplify(det(Jv),'Steps',50);

% singularity when end effector is in line with joint 2 and joint 3
theta = atan2(720,120);
figure();
hold on
plot_robot([0,1,theta]);
plot_robot([0,0,theta]);
axis equal
xlim([-1200,400]);
ylim([-1,1]);
zlim([0,1900]);
hold off
grid on
view(0,0)
title("Singularity Type 1 examples")
xlabel('x axis (mm)');
zlabel('z axis (mm)');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sing1_ex1_rank = rank(Jv_val(0,1,theta))
sing1_ex2_rank = rank(Jv_val(0,0,theta))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% rank is less than 3 meaning there is a singularity

% singularity for when the end effector is over the base link
eqn = Jv_det == 0;
eqn = subs(eqn,t1,0);

T0tip = subs(T0tip,t1,0);
eqn1 = T0tip(1,4) == 0; % x axis is 0 when ee is above base link
eqn2 = T0tip(2,4) == 0; % y axis is 0 when ee is above base link

% solution for all possible singularities for arm above the base joint
[t2_sol,t3_sol,params,conditions] = solve([eqn,eqn1],[t2,t3],...
    'IgnoreAnalyticConstraints',false,...
    'ReturnConditions',true)

%displaying example
figure();
hold on
```

```
% both examples generated from solutions that MATLAB gave with a predefined
% joint 3 position
plot_robot([0,2*atan(1127^(1/2)/19 - 24/19),0]);
plot_robot([0,0.46008129995214253060709097553627,pi/4]);
grid on
view(0,0)
title("Singularity Type 2 examples")
xlabel('x axis (mm)');
zlabel('z axis (mm)');
xlim([-850,200]);
ylim([-1,1]);
zlim([0,1900]);
axis equal
hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sing2_ex1_rank = rank(Jv_val(0,2*atan(1127^(1/2)/19 - 24/19),0))
sing2_ex2_rank = rank(Jv_val(0,0.46008129995214253060709097553627,pi/4))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
sing1_ex1_rank =

     2


sing1_ex2_rank =

     2


t2_sol =

 2*atan((48*z + ((z^2 + 1)*(807*z^2 + 1920*z + 1127))^(1/2) + 16*z^2 + 24)/(29*z^2 + 8*z - 19)) + 2*pi*k
 2*atan((48*z - ((z^2 + 1)*(807*z^2 + 1920*z + 1127))^(1/2) + 16*z^2 + 24)/(29*z^2 + 8*z - 19)) + 2*pi*k


t3_sol =

 2*atan(z) + 2*pi*l
 2*atan(z) + 2*pi*l


params =

[ k, l, z]


conditions =

 0 < 2*atan((48*z + ((z^2 + 1)*(807*z^2 + 1920*z + 1127))^(1/2) + 16*z^2 + 24)/(29*z^2 + 8*z - 19)) + pi*(2*k + 1/3) & 2*atan((48*z + ((z^2 + 1)*(807*z^2 + 1920*z + 1127))^(1/2) + 16*z^2 +
```

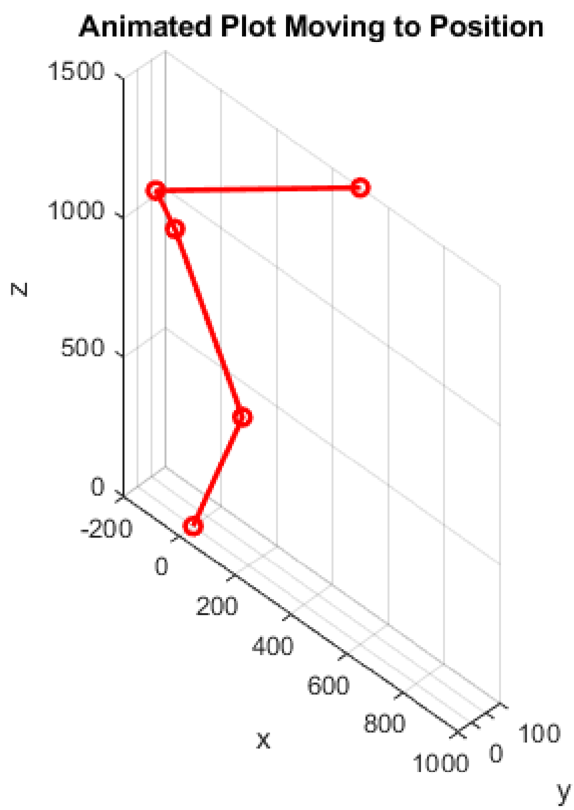
```
24)/(29*z^2 + 8*z - 19)) + pi*(2*k - 1/3) < 0 & 0 < 2*atan(z) + pi*(2*l + 1/3) & 2*atan(z) +
pi*(2*l - 1/3) < 0 & in(k, 'integer') & in(l, 'integer') & (z == - 3^(1/2)/2 - 3/2 | z ==
3^(1/2)/2 - 3/2 | (2*z^2 + 6*z + 3)*(48*z + ((z^2 + 1)*(807*z^2 + 1920*z + 1127))^(1/2) + 16*z^2
+ 24) + (3*z^2 + z - 3)*(29*z^2 + 8*z - 19) ~= 0 & (2*z^2 + 6*z + 3)*(48*z - ((z^2 + 1)*(807*z^2
+ 1920*z + 1127))^(1/2) + 16*z^2 + 24) + (3*z^2 + z - 3)*(29*z^2 + 8*z - 19) ~= 0 & 2*z^2 + 6*z
~= -3 & 29*z^2 + 8*z ~= 19) & (z < (9*7^(1/2))/29 - 4/29 & - (9*7^(1/2))/29 - 4/29 < z | z <= -
(4037^(1/2)*217083^(1/2))/217083 - 320/269 | (4037^(1/2)*217083^(1/2))/217083 - 320/269 <= z & z
< - (9*7^(1/2))/29 - 4/29 | (9*7^(1/2))/29 - 4/29 < z)
 0 < 2*atan((48*z - ((z^2 + 1)*(807*z^2 + 1920*z + 1127))^(1/2) + 16*z^2 + 24)/(29*z^2 + 8*z -
19)) + pi*(2*k + 1/3) & 2*atan((48*z - ((z^2 + 1)*(807*z^2 + 1920*z + 1127))^(1/2) + 16*z^2 +
24)/(29*z^2 + 8*z - 19)) + pi*(2*k - 1/3) < 0 & 0 < 2*atan(z) + pi*(2*l + 1/3) & 2*atan(z) +
pi*(2*l - 1/3) < 0 & in(k, 'integer') & in(l, 'integer') & (z == - 3^(1/2)/2 - 3/2 | z ==
3^(1/2)/2 - 3/2 | (2*z^2 + 6*z + 3)*(48*z + ((z^2 + 1)*(807*z^2 + 1920*z + 1127))^(1/2) + 16*z^2
+ 24) + (3*z^2 + z - 3)*(29*z^2 + 8*z - 19) ~= 0 & (2*z^2 + 6*z + 3)*(48*z - ((z^2 + 1)*(807*z^2
+ 1920*z + 1127))^(1/2) + 16*z^2 + 24) + (3*z^2 + z - 3)*(29*z^2 + 8*z - 19) ~= 0 & 2*z^2 + 6*z
~= -3 & 29*z^2 + 8*z ~= 19) & (z < (9*7^(1/2))/29 - 4/29 & - (9*7^(1/2))/29 - 4/29 < z | z <= -
(4037^(1/2)*217083^(1/2))/217083 - 320/269 | (4037^(1/2)*217083^(1/2))/217083 - 320/269 <= z & z
< - (9*7^(1/2))/29 - 4/29 | (9*7^(1/2))/29 - 4/29 < z)


sing2_ex1_rank =

     2


sing2_ex2_rank =

     2
```


Singularity Type 1 examples
1800
1600
1400
1200
1000
800
600
400
200
0
z axis (mm)
-1000
-500
0
x axis (mm)


Singularity Type 2 examples
1800
1600
1400
1200
1000
800
600
400
200
0
z axis (mm)
-800
-600
-400
-200
0
200
x axis (mm)

Extra credit: Animation

```
close all
desired_pos = [500;100;1500];
fanimator(@(t)animator(t,desired_pos));
title('Animated Plot Moving to Position')
xlabel('x');
ylabel('y');
zlabel('z');
view([45,45])
axis equal
grid on
playAnimation;
snapnow
```



Functions

```
function [t1,t2,t3,t4,t5,t6] = ikin(desired)
% Take the desired point from whats given
desired_pos = desired(1:3,4);
% same with orientation
desired_orient = desired(1:3,1:3);
L1y = 475; L1x = 150; L2 = 600; L3 = 120; L4 = 720;
x = desired_pos(1);
y = desired_pos(2);
```

```
z = desired_pos(3);
% calculating the inverse position kinematics
t1 = atan2(y,x);

% solving for theta 2
d = sqrt(x^2 + y^2)-L1x;
r = sqrt(L3^2 + L4^2);
a = sqrt(d^2 + (z-L1y)^2);
F = (-r^2 + a^2 + L2^2)/(2*a*L2);
beta = atan2(sqrt(1-F^2),F);
gamma = atan2(z-L1y,d);
t2 = beta + gamma - pi/2;

% solving for theta 3
phi = atan2(L3,L4);
G = (-L2^2 + a^2 + r^2)/(2*a*r);
delta = atan2(sqrt(1-G^2),G);
n = (pi/2 - gamma) + delta + phi;
t3 = pi/2-n-t2;

% alternate method for theta 3
phi2 = atan2(720,120);
G2 = (r^2 + 600^2 - a^2)/(2*r*600);
psi = atan2(sqrt(1-G2^2),G2);
sigma = pi - psi - t2;
t3 = phi2 - t2 - sigma;

% calculating the inverse orientation kinematics
R04 =[-sin(t2 + t3)*cos(t1),  sin(t1), cos(t2 + t3)*cos(t1);
    -sin(t2 + t3)*sin(t1), -cos(t1), cos(t2 + t3)*sin(t1);
    cos(t2 + t3),        0,         sin(t2 + t3)];

R47 = (R04')*desired_orient;
% these are based on forward kinematics of a spherical wrist
t4 = atan2(R47(2,3),R47(1,3));
t5 = atan2(sqrt(1-R47(3,3)^2),R47(3,3));
t6 = atan2(R47(3,2),-R47(3,1));
end

function p = plot_robot(q)
L1y = 475; L1x = 150; L2 = 600; L3 = 120; L4 = 720;
t1 = q(1);
t2 = q(2);
t3 = q(3);
DH_table = [t1 L1y L1x sym(pi)/2;
    t2+sym(pi/2) 0 L2 0;
    t3 0 L3 sym(pi/2);
    0 L4 0 0];
T01 = tdh(DH_table(1,:));
T12 = tdh(DH_table(2,:));
T23 = tdh(DH_table(3,:));
T34 = tdh(DH_table(4,:));
```

```
T02 = T01*T12;
T03 = T02*T23;
T04 = T03*T34;
T = [T01 T02 T03 T04];
pos = [0;0;0];
for i = 1:4
    pos = [pos [T(1,i*4); T(2,i*4); T(3,i*4)]];
end
p = plot3(pos(1,:), pos(2,:), pos(3,:),'Linewidth',2,'Color','r',...
    'Marker','o');
end


function p = animator(t,desired_pos)
initial_pos = [870;0;1195]; % home position of end effector
increment = (desired_pos - initial_pos)./10;
inter_pos = initial_pos + increment*t;
% same with orientation
desired_orientation = eye(3,3);
desiredT = [desired_orientation inter_pos; 0 0 0 1];
% run inverse kinematics solved geometrically
[th1, th2, th3, th4, th5, th6] = ikin(desiredT);
q = [th1;th2;th3];
p = plot_robot(q);
end

function h = circle(x,y,r,bounds)
th = linspace(bounds(1),bounds(2),1000);
xunit = r * cos(th) + x;
zunit = r * sin(th) + y;
h = plot3(xunit,zeros(1,1000),zunit);
end
```

Published with MATLAB® R2019b