ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
----------o0o----------

BÁO CÁO BÀI TẬP LỚN
Môn: Kỹ thuật Robot

MÔ PHỎNG VÀ GIẢI CÁC BÀI TOÁN CHO MÔ HÌNH ROBOT SCARA SỬ DỤNG MATLAB

GVHD: TS. NGUYỄN HOÀNG GIÁP
SVTH: ĐÀO MINH TRIẾT
MSSV: 1814426

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2021

LỜI CẢM ƠN

Qua project lần này, em xin gửi lời cảm ơn đến những gì thầy TS. Nguyễn Hoàng Giáp đã truyền dạy trong suốt một học kỳ qua. Nhờ những kiến thức đầy đủ đó đã giúp em có thể hiểu và hoàn thành được project này.

Thông qua đó giúp em hiểu được những khái niệm cơ bản nhất của môn học Kỹ thuật Robot, và cách mà ứng dụng chúng vào thực tế.

Tp. Hồ Chí Minh, ngày 07 tháng 12 năm 2021.

Sinh viên

Đào Minh Triết

TÓM TẮT NỘI DUNG

Thực hiện mô phỏng lại được một hệ robot SCARA. Vận dụng những kiến thức đã học giải các bài toán động học thuận, động học ngược, quy hoạch quỹ đạo, tính Jacobian…

Mục lục

I. Tìm hiểu:

1. Robot Scara:

Với sự phát triển không ngừng của công nghệ, Robot đã và đang là lựa chọn đáng tin cậy, với nhiều tính năng hiện đại, dần thay thế hầu hết các hoạt động cho con người.

Cùng sự phát triển không ngừng đó, thì thị trường robot cũng có nhiều sản phẩm phong phú và đa dạng. Vì lẽ đó để dễ dàng cho người sử dụng, người ta đã phân loại robot ra để đưa ra những phương án tối ưu phát huy hết tiềm năng của robot đó. Và Scara cũng là một loại Robot được ứng dụng phổ biến trong các dây chuyền tự động, hay hàn xì ở các công xưởng chế tạo. Do lớp robot này được phân loại vào dạng robot kết cấu. Cũng giống với những người anh em cùng phân lớp như robot dạng tay người, dạng trụ, dạng cầu, dạng đề các… các trúc sẽ bao gồm các khớp được nối lại với nhau, có thể là khớp dạng trượt hoặc xoay. SCARA là cách viết tắt của Selective Compliance Assembly Robot Arm.

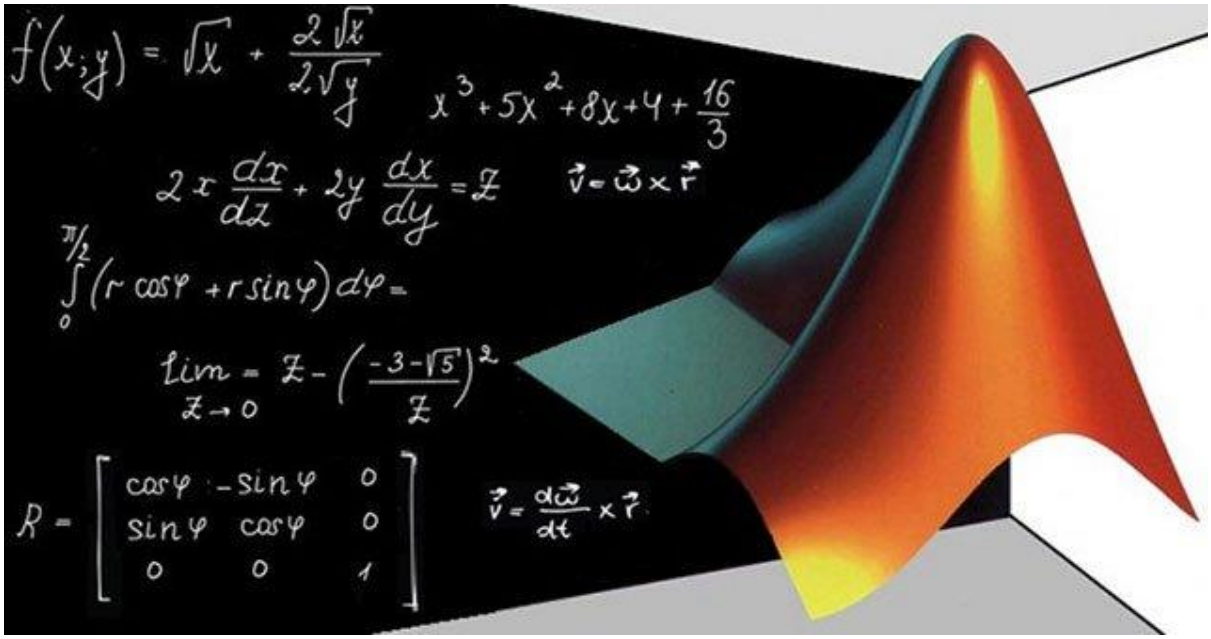
Một hệ SCARA hiện nay thường gồm có 4 bậc tự do: với 3 khớp xoay và 1 khớp tịnh tiến.

Mẫu SCARA Yaskawa MYS650L

SCARA được biết đến là một trong những loại robot công nghiệp phổ biến nhất hiện nay bởi tính ứng dụng cao của nó. Với cấu tạo như trên sẽ cho phép loại robot này gắp vật một cách khá linh hoạt, cùng độ tốc độ và sự chính xác cao trong các công việc lặp đi lặp lại. Vì thế đây là sự lựa chọn rất phù hợp để thay thế con người trong những công việc mang tính chất lặp đi lặp lại, dễ gây nhàm chán.

2. Phần mềm mô phỏng matlab:

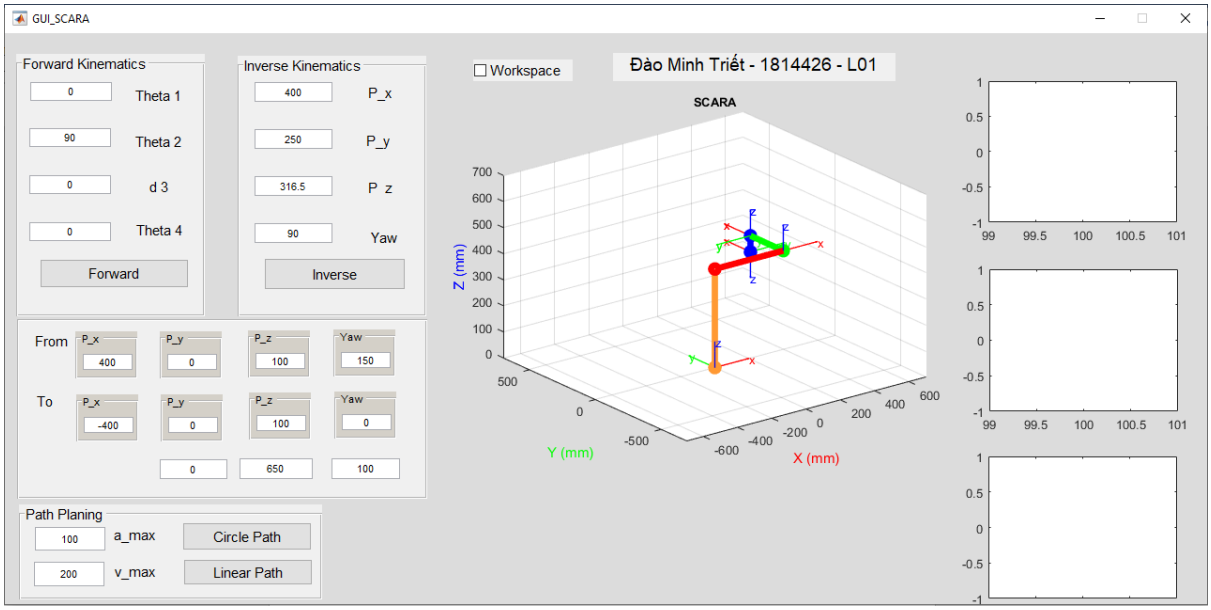
Phần mềm MATLAB – Matrix Laboratory là một môi trường cung cấp các phép tính số học và lập trình. Nó hỗ trợ cả những phép tính ma trận, vẽ đồ thị hàm số, thiết kế giao diện người dùng cũng như nhiều ứng dụng tích hợp để cho phép ta thực hiện các mô phỏng khác nhau. Từ đó, đây là phần mềm vô cùng thông dụng và dễ dàng đáp ứng được những yêu cầu của bài toán báo cáo này.



II. Quá trình thực hiện:

1. Thiết kế giao diện chính:

Bằng cách sử dụng lệnh *guide* có sẵn của Matlab, tiến hành thiết kế nên một giao diện dùng để tương tác giống như hình:



Trên giao diện này sẽ thực hiện các bài toán như động học thuận, động học ngược, tính vận tốc khớp… đồng thời thể hiện mô phỏng quá trình di chuyển của robot, vẽ đồ thị quy hoạch quỹ đạo…

2. Bài toán động học thuận (Forward Kinematic):

Động học thuận là bài toán cơ bản nhất của kỹ thuật Robot, đây có thể xem như là nền tảng cho những bài toán khác. Từ dữ liệu đưa vào là các góc xoay Theta1, 2 và 4 cùng với độ dịch chuyển d3, ta sẽ thiết lập được các ma trận chuyển vị, thể hiện mối quan hệ giữa hai khớp liền kề nhau, từ đó nhân các ma trận đấy lại với nhau và thu được ma trận chuyển vị giữa end-effector và gốc tọa độ.

Ở đây, em sẽ tiến hành code như sau:

```
function [T01,T02,T03,T04] = EF_HomoTransform(t1,t2,d3,t4)
 a1 = 400;
 a2 = 250;
 d1 = 378;
 d4 = -61.5;

T01 = [cos(t1) -sin(t1) 0 a1*cos(t1); sin(t1) cos(t1) 0 a1*sin(t1);0 0 1 d1;0 0 0 1];
T12 = [cos(t2) -sin(t2) 0 a2*cos(t2); sin(t2) cos(t2) 0 a2*sin(t2);0 0 1 0;0 0 0 1];
T23 = [1 0 0 0; 0 1 0 0; 0 0 1 d3; 0 0 0 1];
T34 = [cos(t4) sin(t4) 0 0; sin(t4) -cos(t4) 0 0; 0 0 -1 d4; 0 0 0 1];

T02 = T01*T12;
T03 = T01*T12*T23;
T04 = T01*T12*T23*T34;
```

Function thực hiện tính toán các ma trận chuyển vị

```
function [EF] = draw_plot3d(theta1,theta2,d3,theta4,draw_workspace_check)
theta1=theta1*pi/180;
theta2=theta2*pi/180;
theta4=theta4*pi/180;
[T01,T02,T03,T04] = EF_HomoTransform(theta1,theta2,d3,theta4);
N0 = [1 0 0; 0 1 0; 0 0 1];
A = T01;%*[0; 0; 0; 1]
B = T02;%*[0; 0; 0; 1]
C = T03;%*[0; 0; 0; 1]
EF = T04;%*[0; 0; 0; 1]

P0=[0 0 0];
P1=[0 0 378];
P2=transpose(A(1:3,4));
P3=transpose(B(1:3,4));
P4=transpose(C(1:3,4));
P5=transpose(EF(1:3,4));
Q1=[P0(1) P1(1) P2(1) P3(1) P4(1) P5(1)];
Q2=[P0(2) P1(2) P2(1,2) P3(2) P4(2) P5(2)];
Q3=[P0(3) P1(3) P2(1,3) P3(3) P4(3) P5(3)];

X1=[P0(1,1) P1(1,1)]; X2=[P1(1,1) P2(1,1)];X3=[P2(1,1) P3(1,1)]; X4=[P3(1,1) P4(1,1)];X5=[P4(1,1) P5(1,1)];
Y1=[P0(1,2) P1(1,2)]; Y2=[P1(1,2) P2(1,2)];Y3=[P2(1,2) P3(1,2)]; Y4=[P3(1,2) P4(1,2)];Y5=[P4(1,2) P5(1,2)];
Z1=[P0(1,3) P1(1,3)]; Z2=[P1(1,3) P2(1,3)];Z3=[P2(1,3) P3(1,3)]; Z4=[P3(1,3) P4(1,3)];Z5=[P4(1,3) P5(1,3)];


cla reset;
if draw_workspace_check == 1
    draw_workspace;
    hold on;
end

plot3(X1,Y1,Z1,'-o','color',[255, 153, 51] /255,'LineWidth',5);
if draw_workspace_check == 0
    hold on;
end
plot3(X2,Y2,Z2,'-o','color',[255, 0, 0] / 255,'LineWidth',5);
plot3(X3,Y3,Z3,'-o','color',[0, 255, 0] / 255,'LineWidth',5);
plot3(X4,Y4,Z4,'-o','color',[255, 0, 255] / 255,'LineWidth',5);
plot3(X5,Y5,Z5,'-o','color',[0, 0, 255] / 255,'LineWidth',5);
```

```
draw_axis(P5, EF);
draw_axis(P4, C);
draw_axis(P3, B);
draw_axis(P2, A);
draw_axis(P0, N0);

axis([-1000,1000,-1000,1000,0,1000]);
title('SCARA','Color',[0 0 0],'FontSize',10);
            xlabel('X (mm)','Color',[1 0 0]);
            ylabel('Y (mm)','Color',[0 1 0]);
            zlabel('Z (mm)','Color',[0 0 1]);
grid on;
rotate3d on;

end
```

Fuction bao gồm việc thực hiện động học thuận và vẽ robot

Từ những ma trận trả về, ta tiến hành trích ra các thông số thể hiện vị trí của khớp đó so với góc tọa độ, rồi từ đó tiến hành quá trình vẽ ra robot.

3. Bài toán động học ngược (Inverse Kinematic)

Đây cũng là một bài toán quan trong không khác gì động học thuận. Với cách làm ngược lại động học thuận, cho sẽ cho đầu vào là điểm cần đến, cũng chính là tọa độ của end_effector. Công việc của người lập trình là tìm ra cách giải để tìm ra góc tương ứng cho các khớp, từ đó đưa về bài toán động học thuận để giải tiếp.

Với dữ liệu đưa vào là Px, Py, Pz và góc Yaw.

```
function [Th_1,Th_2,d_3,Th_4] = Inverse_Kinematics(Px,Py,Pz,Yaw)
    a1 = 400;
    a2 = 250;
    d1 = 378;
    d4 = -61.5;
    Th_2 = acos((Px^2+Py^2-a1^2-a2^2)/(2*a1*a2));
    Th_1 = atan(Py/Px)-atan((a2*sin(Th_2))/(a1+a2*cos(Th_2)));
    if (Px < 0 )
        Th_1 = pi + Th_1;
    end
    Th_1 = Th_1*180/pi;
    Th_2 = Th_2*180/pi;
    d_3 = (Pz - 378 + 61.5);
    Th_4 = round(Yaw - Th_1 -Th_2);
```

4. Path Planing:

4.1 Đường thẳng (Linear):

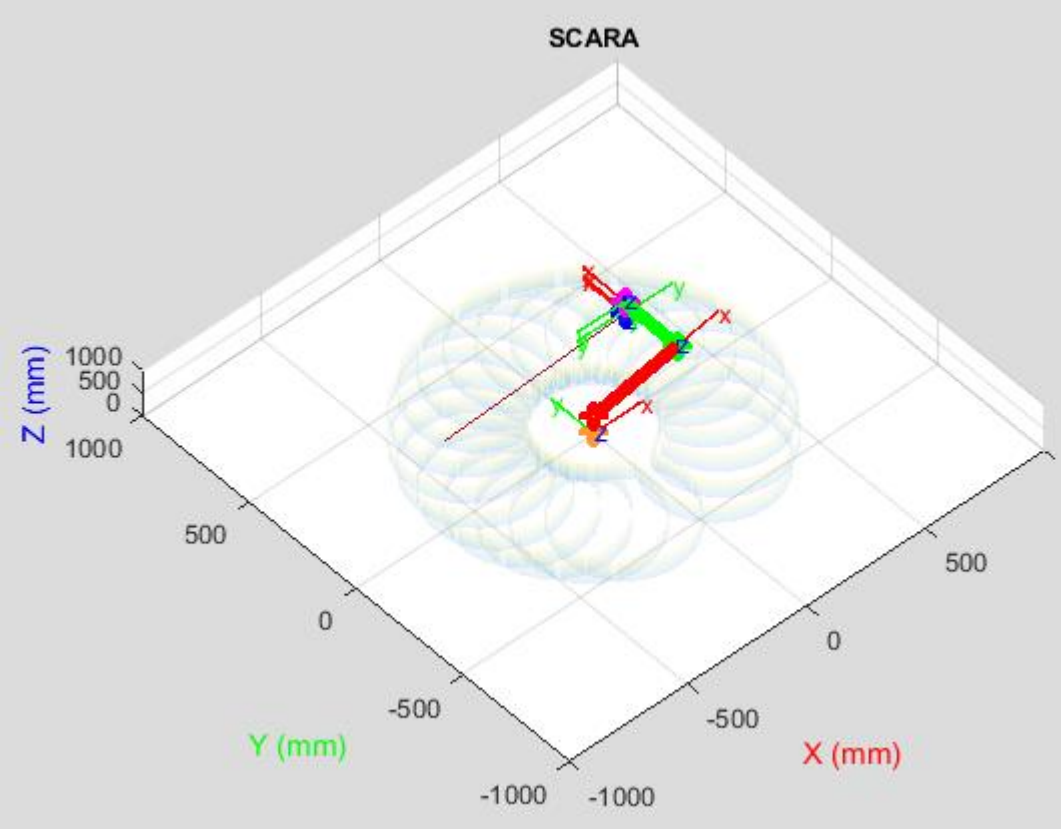
Với ý tưởng sẽ từ hai điểm xuất phát – đích đến, từ đó tính ra độ dài đoạn đường cần di chuyển, sau đó thực hiện chia nhỏ các đoạn ra rồi đưa vào bài toán Inverse Kinematic.

Với dữ liệu đầu vào, sẽ thực hiện tính được vector_EF, từ đó tính ra độ dài đoạn dịch chuyển. Rồi từ những thông số a_max, v_max cùng q_max mới tính được, ta đưa vào bài toán Trajectory Planning. Với dữ liệu trả về tiến hành tính toán sự dịch chuyển trên các trục X Y và Z dựa vào hệ số góc.

```
function [q_x,q_y,q_z,a,v,q,t_max] = linear_path_planning(a_max,v_max,x,y,z,x_1,y_1,z_1)

vector_EF = [x-x_1;y-y_1;z-z_1];

% Do dai dich chuyen tu vi tri cu den vi tri moi
q_max = sqrt(vector_EF(1)^2+vector_EF(2)^2+vector_EF(3)^2);

% He so goc cua vector_EF len 3 truc toa do
cosa_x = (dot(vector_EF,[1;0;0]))/q_max;
cosa_y = (dot(vector_EF,[0;1;0]))/q_max;
cosa_z = (dot(vector_EF,[0;0;1]))/q_max;

[n,delta_q,a,v,q,t_max] = quy_hoach_van_toc_3(q_max,a_max,v_max);
```

```
% Truc Ox
qx = x_1;
for i=1:n
        q_x(i) = qx + delta_q(i)*cosa_x;
        qx = q_x(i);
end

% Truc Oy
qy = y_1;
for i=1:n
        q_y(i) = qy + delta_q(i)*cosa_y;
        qy = q_y(i);
end

% Truc Oz
qz = z_1;
for i=1:n
        q_z(i) = qz + delta_q(i)*cosa_z;
        qz = q_z(i);
end
```

Vì bài toán sẽ sử dụng kết quả của Tranjectory Planning nên phần đó sẽ được trình bày ở phía sau. Và kết quả thực hiện:



4.2 Đường tròn (Circle):

Các thực hiện cũng sẽ giống với bài toán đường thẳng, cũng sẽ cần hai điểm đầu-cuối, cùng 1 điểm trung gian nơi mà end_effector sẽ đi qua. Ý tưởng sẽ là từ 3 điểm đó, xác định được tâm đường tròn ngoại tiếp tam giác ABC và bán kính. Từ những tọa độ đó, cũng sẽ tính nên độ dài cung tròn cũng chính là q_max. Sau khi đã có q_max thì cũng tiến hành tính toán giống với đường thẳng.

Trong đoạn code dưới đây, O chính là tâm của cung tròn, còn R chính là bán kính của đường tròn ngoại tiếp tam giác. Các góc alpha_1 và alpha_2 tính ra được chính là góc tại điểm đầu và góc tại điểm cuối.

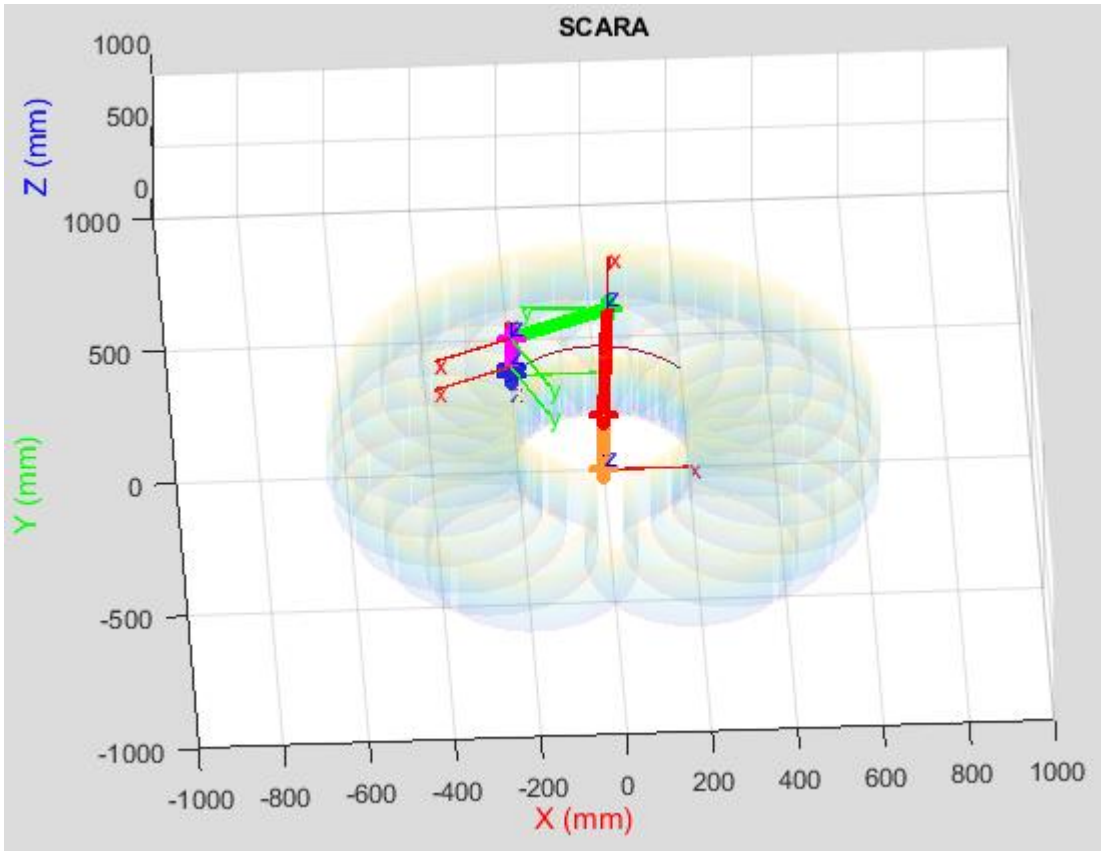
```
function [x,y,z,a,v,q,t_max,n] = circle_path_planning_test(v_max,a_max,x_1,y_1,z_1,x_2,y_2,z_2,x_0,y_0,z_0)

S = 1/2*abs((x_1-x_0)*(y_2-y_0)-(x_2-x_0)*(y_1-y_0));
o1 = sqrt((x_1-x_0)^2+(y_1-y_0)^2);
o2 = sqrt((x_2-x_1)^2+(y_2-y_1)^2);
o3 = sqrt((x_2-x_0)^2+(y_2-y_0)^2);

AC = [x_2-x_0;y_2-y_0;z_2-z_0];
AB = [x_1-x_0;y_1-y_0;z_1-z_0];
n = cross(AC,AB);
d = n'*[x_0;y_0;z_0];

% Tam duong tron:
M = [n';
    2*(x_1-x_0) 2*(y_1-y_0) 2*(z_1-z_0);
    2*(x_2-x_0) 2*(y_2-y_0) 2*(z_2-z_0)];
N = [d;x_1^2+y_1^2+z_1^2-x_0^2-y_0^2-z_0^2;x_2^2+y_2^2+z_2^2-x_0^2-y_0^2-z_0^2];
O = M\N;
R = o1*o2*o3/(4*S);
```

```
% Tam duong tron:
M = [n';
    2*(x_1-x_0) 2*(y_1-y_0) 2*(z_1-z_0);
    2*(x_2-x_0) 2*(y_2-y_0) 2*(z_2-z_0)];
N = [d;x_1^2+y_1^2+z_1^2-x_0^2-y_0^2-z_0^2;x_2^2+y_2^2+z_2^2-x_0^2-y_0^2-z_0^2];
O = M\N;
R = o1*o2*o3/(4*S);

alpha_0 = acos((2*R*R-(x_0-O(1)-R)^2-(y_0-O(2))^2)/(2*R*R));

if y_0 < O(2)
    alpha_0 = 2*pi - alpha_0;
end

alpha_1 = acos((2*R*R-(x_1-O(1)-R)^2-(y_1-O(2))^2)/(2*R*R));

if y_1 < O(2)
    alpha_1 = 2*pi-alpha_1;
end

if y_2 < O(2)
    alpha = 2*pi-alpha;
end

if ((alpha>alpha_0)&&(alpha>alpha_1))||((alpha<alpha_0)&&(alpha<alpha_1))
    if alpha_1 > alpha_0
        alpha_1 = alpha_1-2*pi;
    else
        alpha_0 = alpha_0-2*pi;
    end
end

q_max = R*abs(alpha_1-alpha_0)

[n,~,a,v,q,t_max] = quy_hoach_van_toc_3(q_max,a_max,v_max);
```

Tính toán tọa độ của End_effector tại từng vị trí:

```
for k = 1:n
    x(k) = O(1)+R*cos(alpha_0+(q(k)/q_max)*(alpha_1-alpha_0));
    y(k) = O(2)+R*sin(alpha_0+(q(k)/q_max)*(alpha_1-alpha_0));
    if ((x(k)-x_0)^2+(y(k)-y_0)^2)<((x_2-x_0)^2+(y_2-y_0)^2)
        z(k) = z_0 + sqrt(((x(k)-x_0)^2+(y(k)-y_0)^2)/((x_2-x_0)^2+(y_2-y_0)^2))*(z_2-z_0);
    else
        z(k) = z_2 + sqrt(((x(k)-x_0)^2+(y(k)-y_0)^2)/((x_2-x_1)^2+(y_2-y_1)^2))*(z_1-z_2);
    end
end
end
```

Kết quả thực hiện:



5. Trajectory Planning:

Đây là bài toán quy hoạch và giám sát các thuộc tính về đường đi, vận tốc và gia tốc của end_effector.

Với dữ liệu đầu vào là v_max, a_max cùng q_max được tính thông qua hai bài đã nên phía trên.

Đầu tiên sẽ tiến hành kiểm tra dữ liệu đầu vào, nếu v_max đưa vào quá giới hạn thì sẽ tự tiến hành điều chỉnh

```
if v_max > sqrt(s_max*a_max/2)
    v_max = sqrt(s_max*a_max/2);
end
```

Tiếp theo đó là tiến hành chia đoạn ra

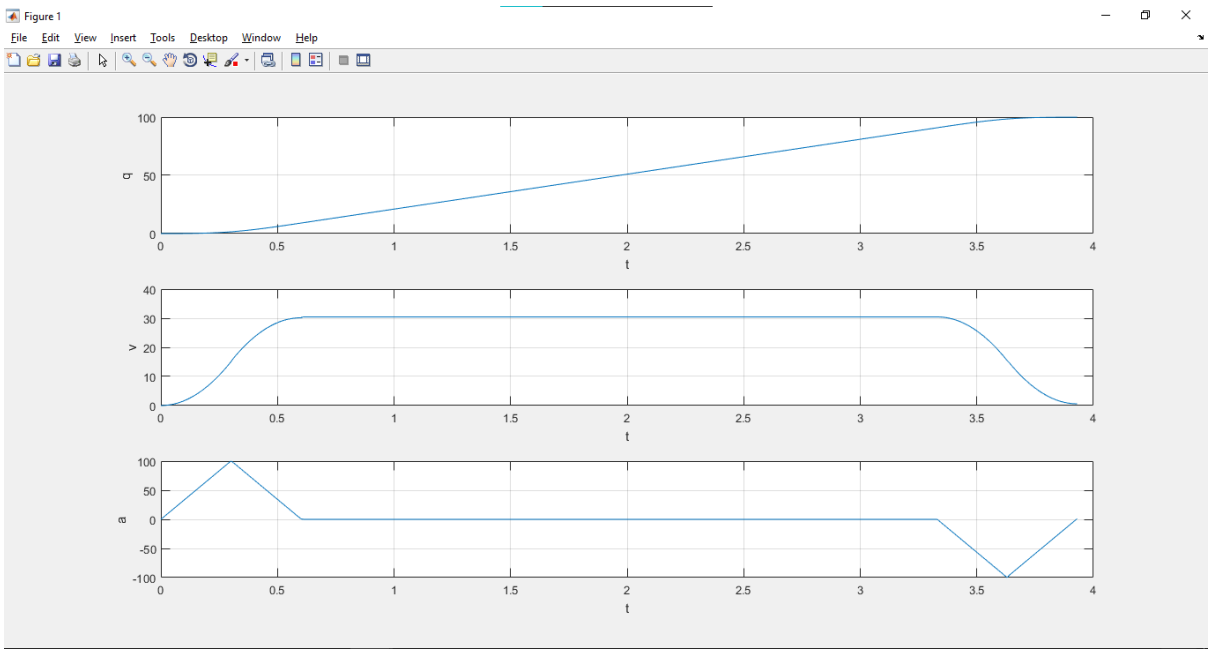
```
T = 0.01;
t1 = v_max/a_max;
t2 = 2*t1;
t3 = s_max/v_max;
t4 = t3 + t1;
t5 = t3 + 2*t1;
t_max = 0:T:t5;
b = a_max/t1;
%detalT = (s_max-2*b*t1^3)/v_max;

n1 = floor(t1/T);
n2 = floor(t2/T);
n3 = floor(t3/T);
n4 = floor(t4/T);
n5 = floor(t5/T);
q(1) = 0;
k = (a_max^2)/v_max;
```

Với những khoảng đã chia ra, tiến hành tính toán các giá trị:

```
for i = 1:n1
            a(i+1) = (k*(i*T));
            v(i+1) = (k*(i*T)^2)/2;
            q(i+1) = (k*(i*T)^3)/6;
            delta_q(i) = q(i+1) - q(i);
end
for i = n1+1:n2
            a(i+1) = a(n1+1) + (k*(n1*T)*(T))-(k*(((i-n1)*T)));
            v(i+1) = v(n1+1) + (k*((T*n1)^2))*T/2 + (k*(n1*T)*(((i-n1)*T))) -(k*(((i-n1)*T)^2))/2;
            q(i+1) = q(n1+1) + (k*((T*n1)^2)*((i-n1)*T))/2 + (k*(n1*T)*(((i-n1)*T)^2))/2 -(k*(((i-n1)*T)^3))/6;
            delta_q(i) = q(i+1) - q(i);
end
V_max = (k*(n1*T)^2)/2 + k*(n1*T)*(n2-n1)*T - (k*((n2 - n1)*T)^2)/2;
for i = n2+1:n3
            a(i+1) = 0*i;
            v(i+1) = v(n2+1) + V_max*T;
            q(i+1) = q(n2+1) + V_max*(i-n2)*T;
            delta_q(i) = q(i+1) - q(i);
```

```
for i = n3+1:n4
            a(i+1) = a(n3+1) - (k*(((i-n3)*T)));
            v(i+1) = v(n3+1) - (k*(((i-n3)*T)^2))/2;
            q(i+1) = q(n3+1) + V_max*(i-n3)*T - (k*(((i-n3)*T)^3))/6;
            delta_q(i) = q(i+1) - q(i);
end
for i = n4+1:n5
            a(i+1) = a(n4+1) + ((k*(n4-n3)*T*T)) + (k*((i-n4)*T));
            v(i+1) = v(n4+1) + (V_max -(k*((n4-n3)*T)^2)/2)*T - (k*(n4-n3)*T*((i-n4)*T)) + (k*((i-n4)*T)^2)/2;
            q(i+1) = q(n4+1) + (V_max -(k*((n4-n3)*T)^2)/2)*(i-n4)*T - (k*(n4-n3)*T*((i-n4)*T)^2)/2 + (k*((i-n4)*T)^
            delta_q(i) = q(i+1) - q(i);
end
n = n5;
```

Kết quả ví dụ:



6. Differential Kinematic (Geometic Jacobian)

6.1 Ma trận Jacobian:

Thực hiện tìm ma trận Jacobian:

```
syms theta1 theta2 d3 theta4
[T01,T02,T03,T04] = EF_HomoTransform(theta1,theta2,d3,theta4);
z0 = [0; 0; 1]; p0 = [0; 0; 0];
z1 = T01(1:3,3);p1 = T01(1:3,4);
z2 = T02(1:3,3);p2 = T02(1:3,4);
z3 = T03(1:3,3);p3 = T03(1:3,4);
z4 = T04(1:3,3);p4 = T04(1:3,4);
a = p4 - p0;
Jp1 = [-z0(3)*a(2)+z0(2)*a(3); z0(3)*a(1)-z0(1)*a(3); -z0(2)*a(1)+z0(1)*a(2)];
a = p4 - p1;
Jp2 = [-z1(3)*a(2)+z1(2)*a(3); z1(3)*a(1)-z1(1)*a(3); -z1(2)*a(1)+z1(1)*a(2)];
a = p4 - p2;
Jp3 = z2;
a = p4-p3;
Jp4 = [-z3(3)*a(2)+z3(2)*a(3); z3(3)*a(1)-z3(1)*a(3); -z3(2)*a(1)+z3(1)*a(2)];
Jo1 = z0; Jo2 = z1; Jo3 = [0; 0; 0]; Jo4 = z3;
A = cat(1,Jp1,Jo1);
B = cat(1,Jp2,Jo2);
C = cat(1,Jp3,Jo3);
D = cat(1,Jp4,Jo4);
Jacobian = simplify(cat(2,A,B,C,D));
```

Kết quả:

```
Jacobian1 =

[ - 250*sin(theta1 + theta2) - 400*sin(theta1), -250*sin(theta1 + theta2), 0, 0]
[   250*cos(theta1 + theta2) + 400*cos(theta1),  250*cos(theta1 + theta2), 0, 0]
[                                           0,                          0, 1, 0]
[                                           0,                          0, 0, 0]
[                                           0,                          0, 0, 0]
[                                           1,                          1, 0, 1]
```

Ta sẽ thu về được một ma trận Jacobian 4x4 vì đã loại bỏ đi những hàng chỉ mang giá trị 0.

6.2 Vận tốc tức thời các khớp:

Sau khi đã có ma trận Jacobian, ta tiến hành tính toán vận tốc tức thời các khớp dựa vào vận tốc tại thời điểm đó của end-effector:

$$v_e = \begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = J(p)\dot{q}$$

Với J(p) là Jacobian, ta sẽ có vận tốc tức thời như sau:

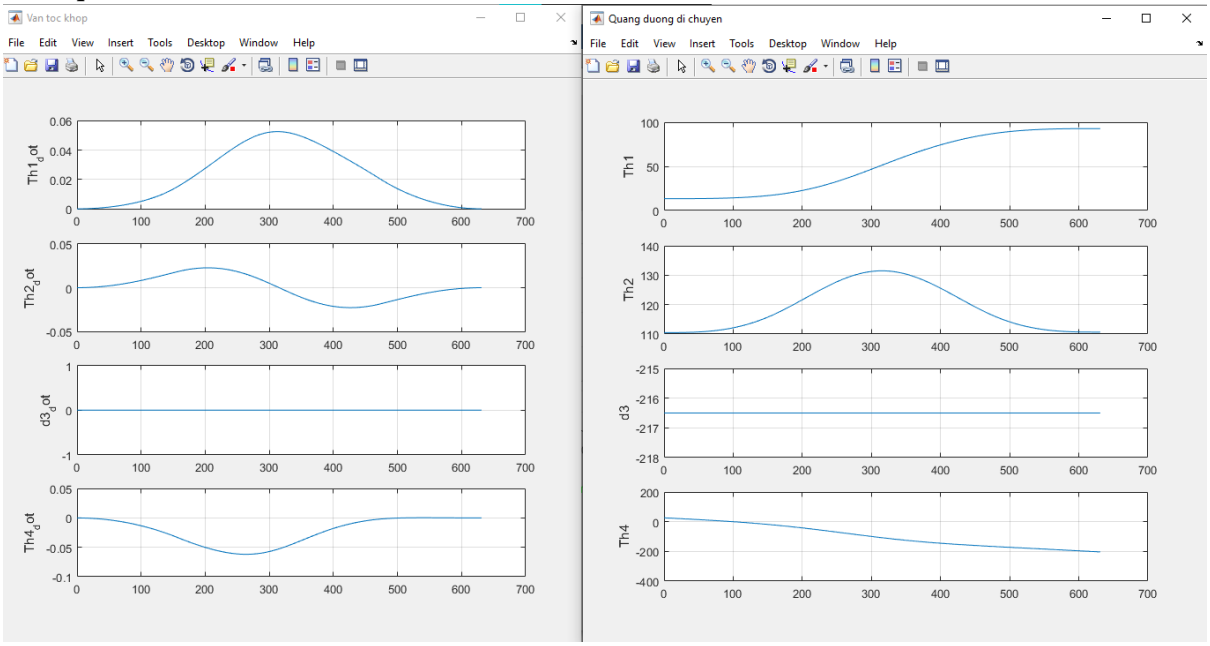
$$\dot{q} = J(p)^{-1} \times v_e$$

Tiến hành code:

```
v_end = [(q_x(i)-q_x(i-1))/T;(q_y(i)-q_y(i-1))/T;(q_z(i)-q_z(i-1))/T;(yaw_draw(i)-yaw_draw(i-1))/T];
J = [-250*sin(Th1(i)*pi/180+Th2(i)*pi/180)-400*sin(Th1(i)*pi/180) -250*sin(Th1(i)*pi/180+Th2(i)*pi/180) 0 0;
    250*cos(Th1(i)*pi/180+Th2(i)*pi/180)+400*cos(Th1(i)*pi/180) 250*cos(Th1(i)*pi/180+Th2(i)*pi/180) 0 0;
    0 0 1 0;
    1 1 0 1];
v_joint = inv(J)*v_end;
theta1_dot(i) = v_joint(1,1);
theta2_dot(i) = v_joint(2,1);
d3_dot(i) = v_joint(3,1);
theta4_dot(i) = v_joint(4,1);
```

Trong đó v_end là vận tốc của end_effector theo các phương x, y, z và góc yaw. J là ma trận Jacobian đã tính ra ở trên. Các giá trị theta1_dot, theta2_dot, d3_dot và theta4_dot lần lượt là vận tốc tức thời của các khớp.

Kết quả ví dụ:


Van toc khop
File Edit View Insert Tools Desktop Window Help
Th1_dot
Th2_dot
d3_dot
Th4_dot
Quang duong di chuyen
File Edit View Insert Tools Desktop Window Help
Th1
Th2
d3
Th4

III. Kết quả thực hiện:

Kết quả chi tiết sẽ được trình bày ở phần video kèm theo.

Nhận xét:

- Những gì làm được:

+ Thành công trong việc mô phỏng lại các chức năng cơ bản của một hệ SCARA.

+ Vận dụng được những kiến thức đã học vào thực hiện một dự án mô phỏng nhỏ.

+ Thực hiện được các bài toàn cơ bản như động học thuận, động học ngược, các bài toán Path Planning và Trajcectory Planning, tính vận tốc góc thông qua ma trận Jacobian, thể hiện nên đặc tính khi đi qua điểm kỳ dị.

- Những điểm hạn chế:

+ Thời gian chạy mô hình chưa tối ưu.
+ Còn nhiều điểm phải cải thiện, Guide chưa tối giản và đẹp mắt, chưa chạy được các đồ thị q, v, a theo thời gian thực.

Tài liệu tham khảo:

[1] Slide bài giảng môn *Kỹ thuật Robot* – TS. Nguyễn Hoàng Giáp

[2] Robotic Modelling Planning and Control – *Bruno Siciliano – Leno Sciavicco – Luigi Villani – Giuseppe Oriolo.*