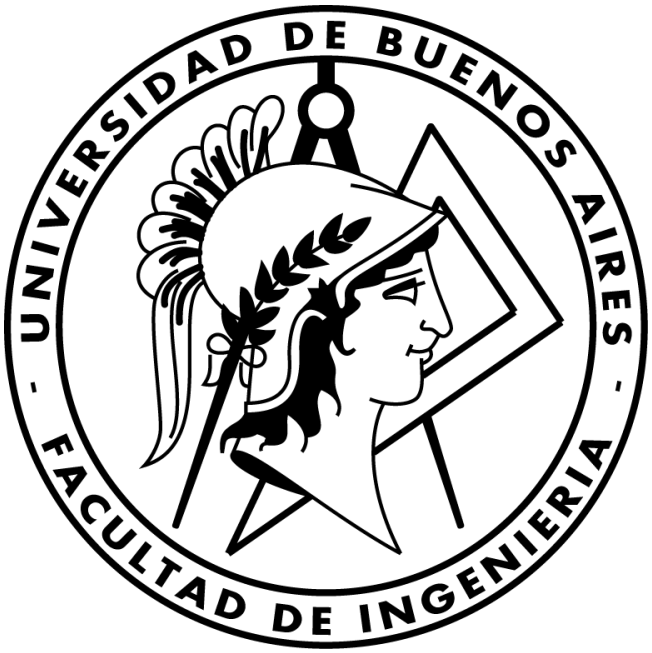
UNIVERSIDAD DE BUENOS AIRES
Facultad de Ingeniería



Trabajo Práctico Nº 4

Robótica (85.15)

Generación de Trayectorias

Integrantes:

Vera Guzmán, Ramiro Augusto - ramiro.vera.g@gmail.com - #95887

Eicheinbaum, Daniel Matias - leinaxd@gmail.com - #95233

---

1. Introducción

El objetivo del presente trabajo consiste en desarrollar algoritmos de generación de trayectorias basados en un robot SCARA. Cabe destacar los algoritmos serán pensados para ser ejecutados en tiempo real para realizar trayectorias tipo *joint* y cartesianas. El enunciado del mismo sera presentado al final del documento.

Figura 1: Robot SCARA

Abuso de notación: En el presente trabajo se trabajó con posiciones, velocidades y aceleraciones. Por comodidad, si la posición se denomina q, la velocidad se denominará *qdot* haciendo referencia a la expresión mecánica de la velocidad y la misma idea se aplicará a la aceleración como *qddot*.

1.1. Ejercicio 1

El objetivo del punto es a partir de una entrada de la forma

$$POSE0 = [x0; y0; z0; Roll; conf]^t \tag{1}$$

obtener los parámetros q del SCARA.

Para desarrollar el algoritmo pedido se partió del problema inverso del SCARA (desarrollado previamente).

Dado que el problema inverso utiliza la posición cartesiana y los elementos de la matriz de rotación, bastaba con modificar el calculo pero sustituyéndolo por el parámetro *Roll* Sabiendo que *Roll* representa la rotación de la herramienta respecto al origen ; *Roll* es la suma de los parámetros q1, q2 y q4 por lo que basta modificar el problema inverso para obtener el resultado.

El algoritmo, denominado ***tp4_punto1.m***, es ejecutado y puesto a prueba en el archivo a primera sección de ***main.m***

Se muestra el contenido de ***tp4_punto1.m***

```
%% TP4 - Punto 1
%ATENCION: POSE0 tiene el parametro de ROLL en grados. El parametro de
%salida q esta en radianes. Sin embargo se imprime al final la conversion
%en grados para facil chequeo.

function q = tp4_punto1(POSE0,conf)
%Tipo de Robot : Scara
%Parametros
a1 = 200; %mm
a2 = 200; %mm
a = [a1,a2];

%POSE0 = [x0; y0; z0; Roll; conf]^T
```

```
% Tanto x0,y0,z0 deben estar en mm
% Roll debe estar en grados
p = POSE0(1:3);
Roll = POSE0(4);

%% Inversa de Scara

%% Obtengo q2
    if(p(1)^2+p(2)^2 > (a1+a2)^2 || p(1)^2+p(2)^2 < (a1-a2)^2 )
        disp('Punto no alcanzable')
        q = NaN;
        return
    end

    c2 = ( p(1)^2+p(2)^2-(a1^2+a2^2) )/(2*a1*a2);
    s2 =  conf(1)*sqrt(1-c2^2);

    %%%%%%%%%%%%%%%%%%%%
    q(2) = atan2(s2,c2);
    %%%%%%%%%%%%%%%%%%%%

    if(abs(q(2))<10^-10)
        q(2)=0;
    end
%% Obtengo q1
    if( p(1)==0 && p(2)==0 )
        disp('Punto no alcanzable')
        q = NaN;
        return
    end

    s1 = ( a2*(p(2)*c2 - p(1)*s2) + a1*p(2) )/(p(1)^2 + p(2)^2);
    c1 = ( a2*(p(2)*s2 + p(1)*c2) + a1*p(1) )/(p(1)^2 + p(2)^2);

    %%%%%%%%%%%%%%%%%%%%
    q(1) = atan2(s1 ,c1);
    %%%%%%%%%%%%%%%%%%%%

    if(abs(q(1))<10^-10)
        q(1)=0;
    end
%% Obtengo q3

    %%%%%%%%%%
    q(3) = p(3);
    %%%%%%%%%%

    if(abs(q(3))<10^-10)
            q(3)=0;
    end
%% Obtengo q4

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    q(4) = deg2rad(Roll) -q(1) -q(2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if(abs(q(4))<10^-10)
            q(4)=0;
    end

    qgrad = [rad2deg(q(1:2)),q(3),rad2deg(q(4))];

    disp('Los valores obtenidos de q (rad) son:')
    disp(q)
    disp('Los valores obtenidos de q (deg) son:')
    disp(qgrad)


%% Analizo topes mecanicos

% Analizo topes Mecanicos

if  ( q(2) < deg2rad(-150) ) || ...
    ( q(2) > deg2rad(150)  ) || ...
    ( q(3) < -250 )          || ...
    ( q(3) > -50  )

    disp('El robot llego a su tope mecanico')
    if  ( q(2) < deg2rad(-150) )
        disp(['q2 se fija en -150 grados'] )
        q(2) = deg2rad(-150);
```

```
    end
    if ( q(2) > deg2rad(150)  )
        disp(['q2 se fija en 150 grados'])
        q(2) = deg2rad(150);
    end
    if( q(3) < -250 )
        disp(['q3 se fija en -250 mm'] )
        q(3) = -250;
    end
    if ( q(3) > -50 )
        disp(['q3 se fija en -50 mm'] )
        q(3) = -50;
    end
end

end
```

1.2. Ejercicio 2

El ejercicio 2 pedía realizar una trayectoria con interpolación *joint* entre 2 posiciones dadas como

$$POSEA = [-200; 200; -100; 0; 1]^t \tag{2}$$

$$POSEB = [200; 200; -200; 90; 1]^t \tag{3}$$

de manera que el robot comience en la POSEA pase por la POSEB y regrese a la POSEA.

Se ejecuto el algoritmo anterior para obtener los parámetros q asociados a las poses, qa y qb y luego de eso se aplico el algoritmo visto en la cátedra, desarrollado en el archivo ***tp4_punto2.m***. Cabe mencionar que el algoritmo es capaz de ejecutarse en un sistema de tiempo real.

El mismo esta puesto a prueba en la sección dos del archivo ***main.m***. Se adjunta el codigo ***tp4_punto2.m*** seguido de las imágenes asociadas al mismo.

```
%% TP4 - Punto 2


function [q,qdot,qddot,sameSegment,Tj] = tp4_punto2(tseg,A,B,C,td,Ts,tacc)
    sameSegment = true;

    if tseg == -tacc
        setTj(0);
    end

    Tj = getTj;

    DA = A-B;
    DC = C-B;

    if tseg >= (Tj - tacc)

        vq1max = 90;
        vq2max = 180;
        vq3max = 1000;
        vq4max = 360;
        vmax = [vq1max,vq2max,vq3max,vq4max]';

        %Como tengo todo en ms divido vmax por 1000
        Tj = max( [max(abs(DA./(vmax / 1000))),2*tacc,td] );
        Tj = ceil(Tj/Ts)*Ts;
        setTj(Tj);

    end

    tp = tseg + tacc;
    tm = tseg - tacc;
    DCT = DC /Tj;
    DAT = DA /tacc;

    if tseg <= tacc
```

```
37
38          qddot = ( DCT + DAT )/(2*tacc);
39          qdot  = ( DCT*tp + DAT*tm )/(2*tacc);
40          q     = ( DCT * tp^2 + DAT * tm^2)/(4*tacc) + B;
41
42      elseif tseg <= Tj - tacc + Ts
43          qddot = zeros(size(B));
44          qdot  = DCT;
45          q     = DCT * tseg + B;
46
47          if tseg == Tj - tacc + Ts
48              sameSegment = false;
49          end
50      else
51          disp('Inesperado')
52      end
53
54  end
```

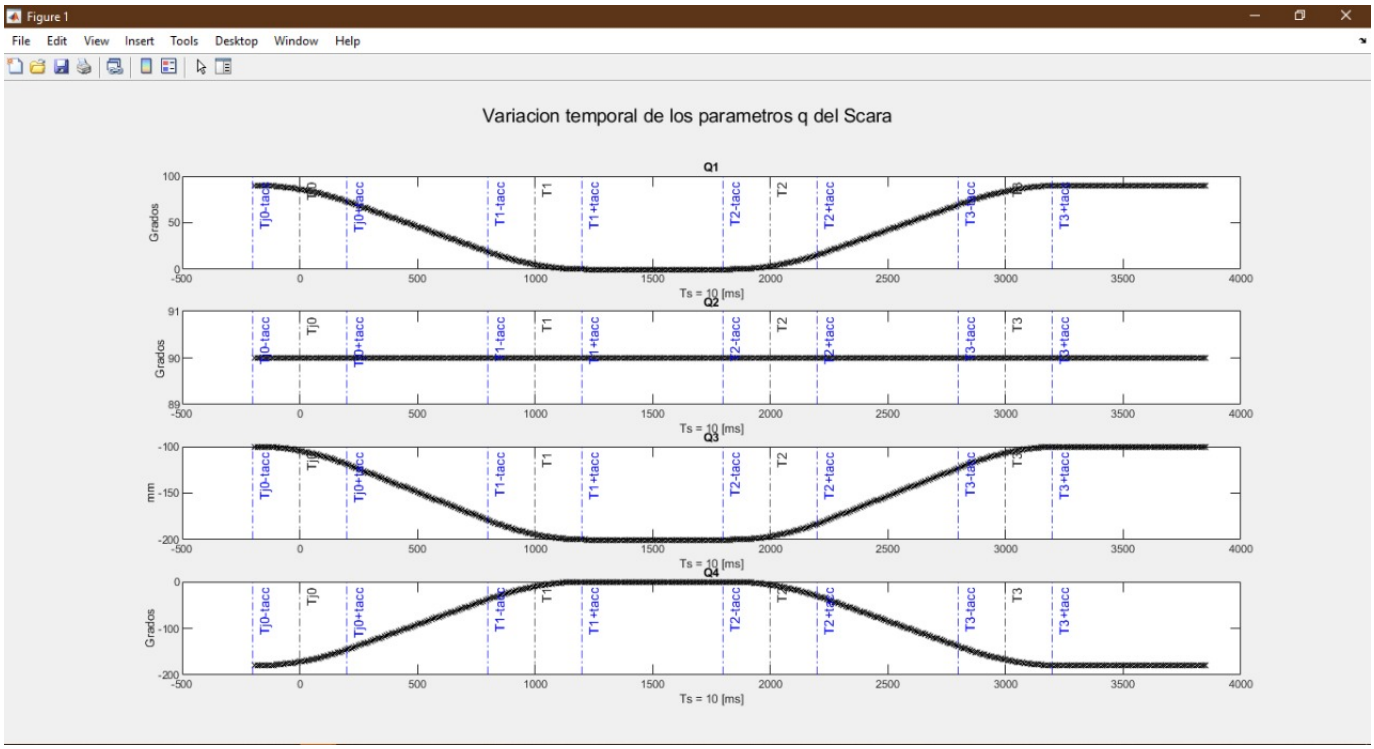


Figura 2: Parámetros Q para movimiento Joint

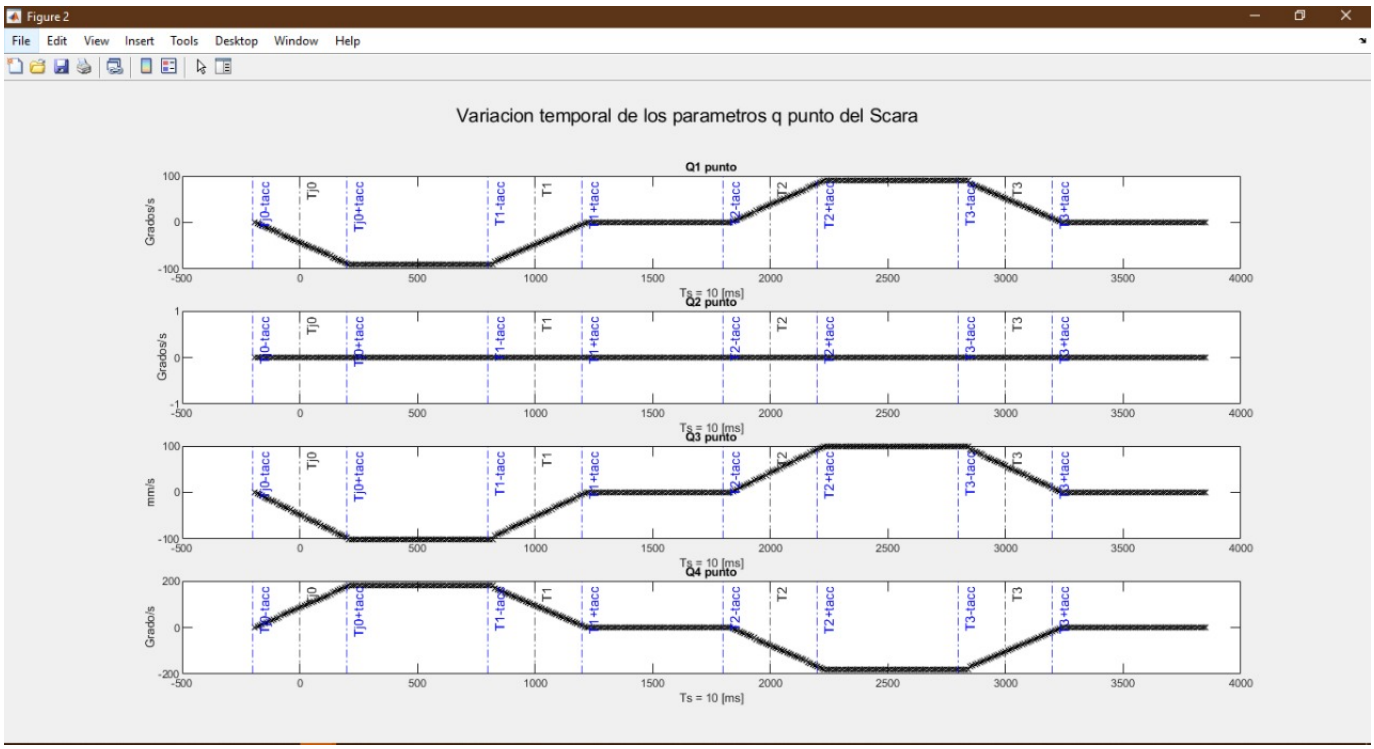


Figura 3: Parámetros Q para movimiento Joint

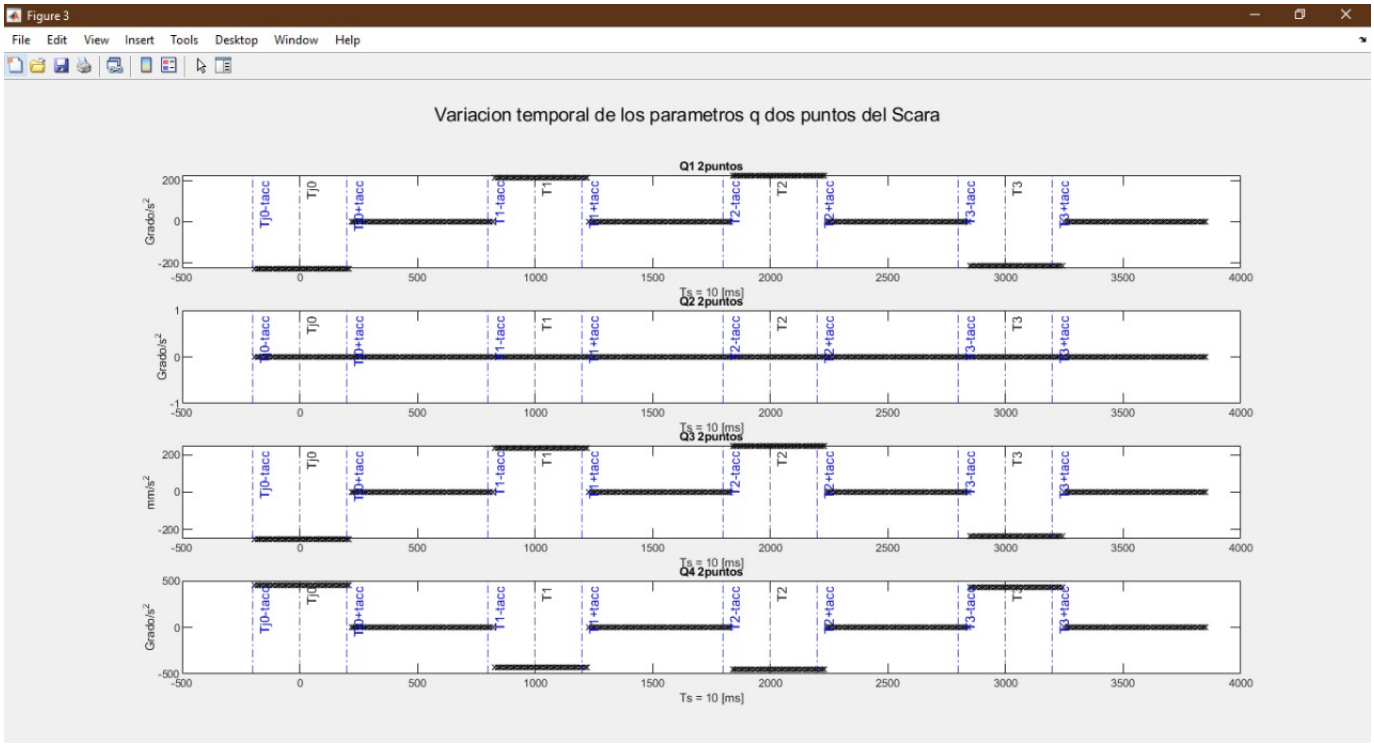


Figura 4: Parámetros Q para movimiento Joint

Se aprecian que las formas de Q,Qdot y Qddot siguen las formas acordes al Metodo de Paul. Además cabe destacar que q1dot (la velocidad de q1) se ajusta a la velocidad máxima admisible (90 º/seg) y esta es la limitante entre las demás velocidades, mostrando que el algoritmo cumple los objetivos de suavizar los movimientos, limitar las velocidades y evitar los impulsos.

La trayectoria final resulta un movimiento *joint*. Es interesante notar que el movimiento no tiene por que ser perfectamente circular, puesto que solo pedía ser necesario tocar la POSEA

y POSEB que si lo realiza.

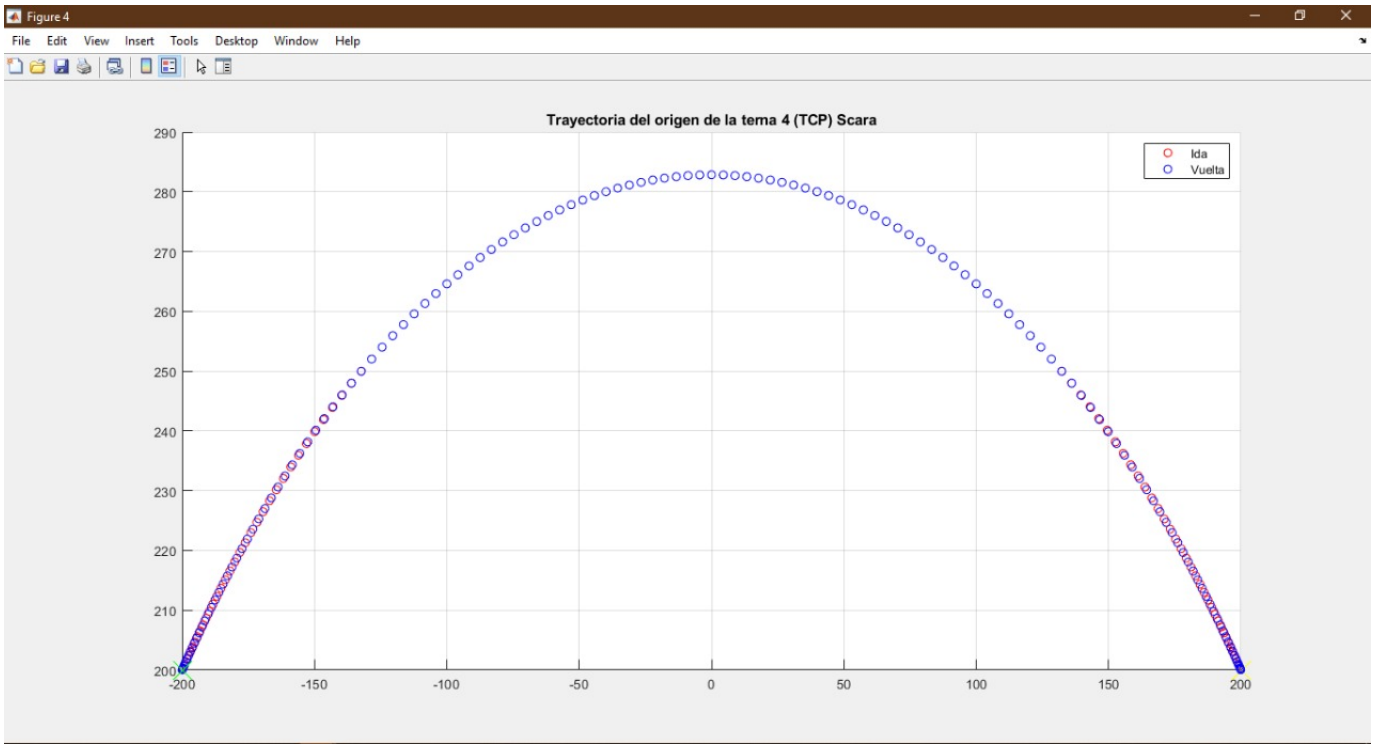


Figura 5: Trayectoria joint entre las posiciones

1.3. Ejercicio 3

El ejercicio 3 pedía realizar un movimiento de tipo cartesiano basado en la información del punto 2.

Para realizar este algoritmo se utilizo la interpolación *joint* pero no sobre los parámetros qa y qb sino que sobre las mismas poses en si. Para ello fue necesario modificar el algoritmo de interpolación, como se explica en las diapositivas de las cátedras.

Se hicieron cambios adicionales teniendo en cuenta que el calculo de Tj implica una competencia entre tiempos y dependiendo el parámetro a interpolar se debe corregir lo asociado a vmax. En el interpolador *joint* se realizaba DA./vmax puesto que se hacia q/qdot = t en cada elemento. En este interpolador es necesario obtener las velocidades máximas cartesianas para obtener los tiempos cartesianos mínimos.

Para ello se obtuvo la matriz jacobiana del SCARA[1] según la terna 0. Luego, a partir de la pose A y pose B se obtuvieron con el primer algoritmo los parámetros q de ambas poses y así los jacobianos asociados a dichas poses. Basto simplemente realizar la operación $J.vmax^t$ para obtener las velocidades cartesianas máximas. Como criterio se opto por tomar el mínimo entre ambas.

El código puede verse en el archivo ***tp4_punto3.m*** y puesto a prueba en la sección 3 del archivo ***main.m***.

Se adjunta ***tp4_punto3.m*** y los resultados obtenidos

```
1 %% TP4 - Punto 3
2
```

1. https://thydzik.com/academic/robotics-315/chap5.pdf pag. 20

```
3
4  function [q,qdot,qddot,sameSegment,POSEActual,Tj] = tp4_punto3(tseg,A,B,C,td,Ts,tacc,qant,
       qdotant,maxCartesianV)
5
6  %Armo DBA(-tacc)
7      DBA = B\A;
8      RBA = DBA(1:3,1:3);
9      pBA = DBA(1:3,4);
10 %Obtengo angulos de Euler
11 %sol1=[phi1,theta1,psi1];
12     sol1 = getEulerAnglesfromR(RBA,'ZYZ',0);
13     phiBA   = sol1(1);
14     thetaBA = sol1(2);
15     psiBA   = sol1(3);
16 %Armo ThetaA
17     ThetaA = [pBA;thetaBA;psiBA];
18 %Armo DBC(Tj)
19     DBC = B\C;
20     RBC = DBC(1:3,1:3);
21     pBC = DBC(1:3,4);
22 %Obtengo angulos de Euler
23 %sol1=[phi1,theta1,psi1];
24     sol1 = getEulerAnglesfromR(RBC,'ZYZ',0);
25     phiBC   = sol1(1);
26     thetaBC = sol1(2);
27     psiBC   = sol1(3);
28 %Armo ThetaC
29     ThetaC = [pBC;thetaBC;psiBC];
30 %Armo ThetaB
31     ThetaB = zeros(size(ThetaC));
32
33
34 %Idem interpolacion joint pero modificado para poner phi
35 sameSegment = true;
36 if tseg == -tacc
37     setTj(0);
38 end
39     Tj = getTj;
40
41     DA = ThetaA;
42     DC = ThetaC;
43
44 if tseg >= (Tj - tacc)
45
46 %     vq1max = 90;
47 %     vq2max = 180;
48 %     vq3max = 1000;
49       vq4max = 360;
50 %     vmax = [vq1max,vq2max,vq3max,vq4max]';
51
52     %Como tengo todo en ms divido vmax por 1000
53     vmax = [maxCartesianV(1:3)',deg2rad(vq4max),deg2rad(vq4max)]';
54     Tj = max( [max(abs(DA./(vmax / 1000))),2*tacc,td] );
55     %Tj = max( [2*tacc,td] );
56     Tj = ceil(Tj/Ts)*Ts;
57     setTj(Tj);
58
59 end
60
61 tp = tseg + tacc;
62 tm = tseg - tacc;
63 DCT = DC /Tj;
64 DAT = DA /tacc;
65
66 if tseg <= tacc %Zona1
67
68     %ThetaDDot = ( DCT + DAT )/(2*tacc);
69     %ThetaDot  = ( DCT*tp + DAT*tm )/(2*tacc);
70     Theta     = ( DCT .* tp^2 + DAT .* tm^2)/(4*tacc) ;
71     Phi       = ( phiBC - phiBA )*tp/(2*tacc) + phiBA;
72
73
74 elseif tseg <= Tj - tacc + Ts %Zona2
75     %ThetaDDot = zeros(size(B));
76     %ThetaDot  = DCT;
77     Theta     = DCT .* tseg;
78     Phi       = phiBC;
79
80     if tseg == Tj - tacc + Ts
81         sameSegment = false;
```

```
    end
else
    disp('Inesperado')
end

%Armo D
R = getRfromEulerAngles([Phi,Theta(4,1),Theta(5,1)],'ZYZ');

D(1:3,1:3) = R;
D(1:3, 4 ) = Theta(1:3);
D( 4 ,1:3) = [0,0,0];
D( 4 , 4 ) = 1;

%Obtengo Pose actual
POSEActual = B*D;

%A partir del problema cinematico inverso obtengo q
q = Inv_SCARA(POSEActual,1,0,[200,200]);

qdot = ( q - qant )/Ts ;
qddot = ( qdot - qdotant )/Ts;

end
```

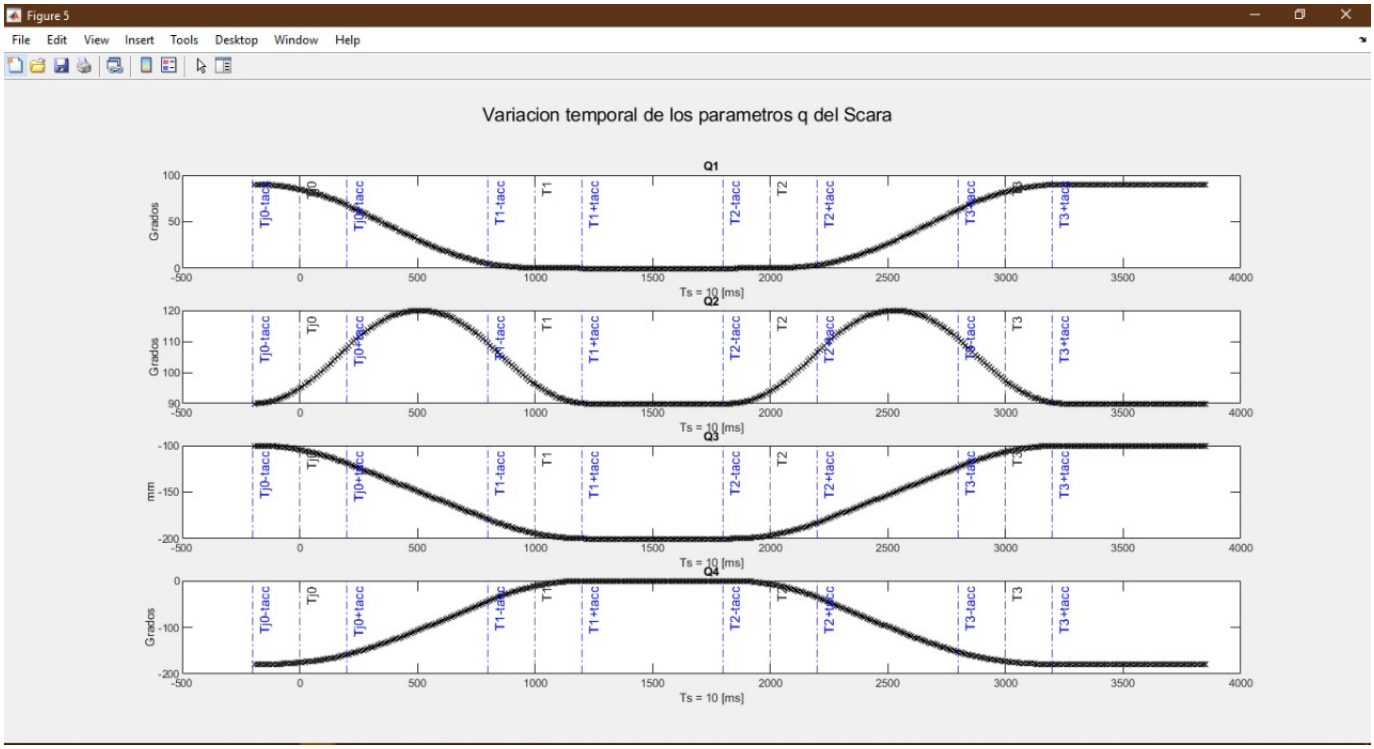


Figura 6: Parámetros Q para movimiento Cartesiano

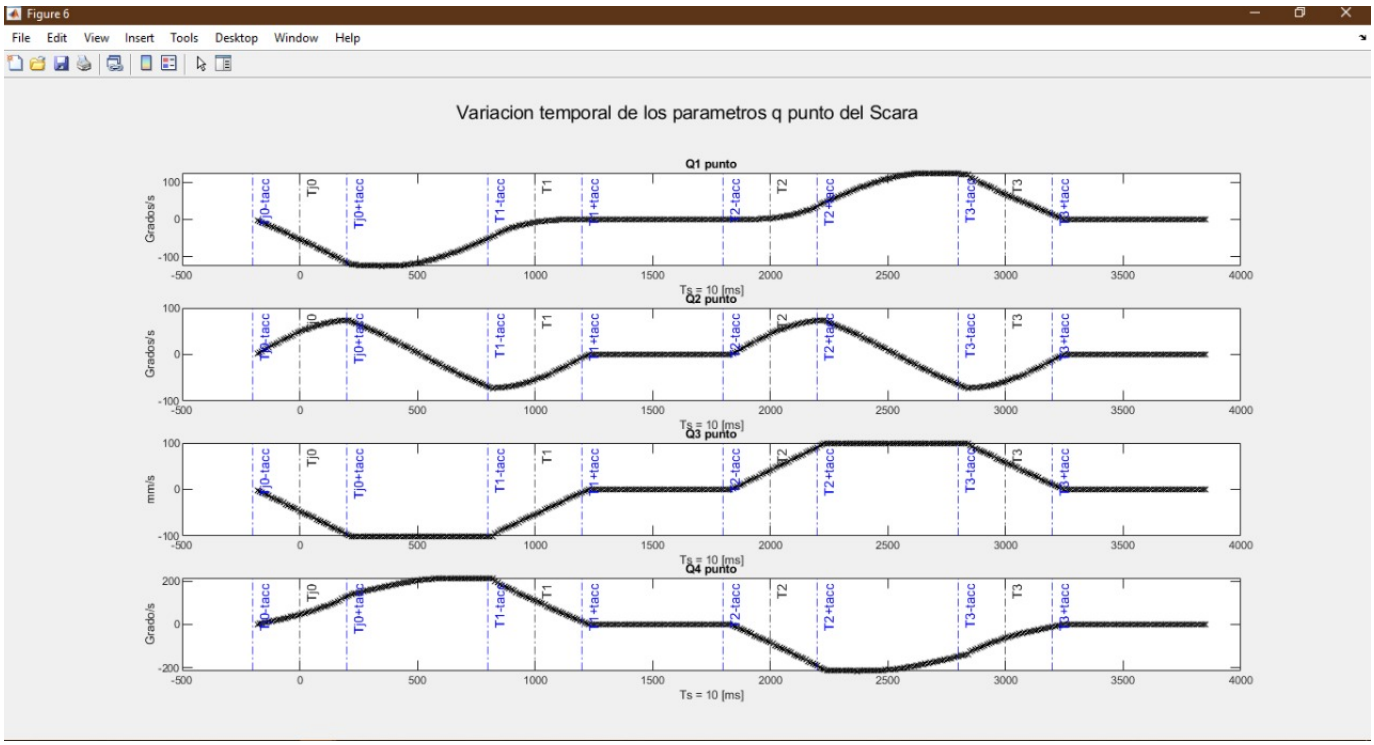


Figura 7: Parámetros Q dot para movimiento Cartesiano

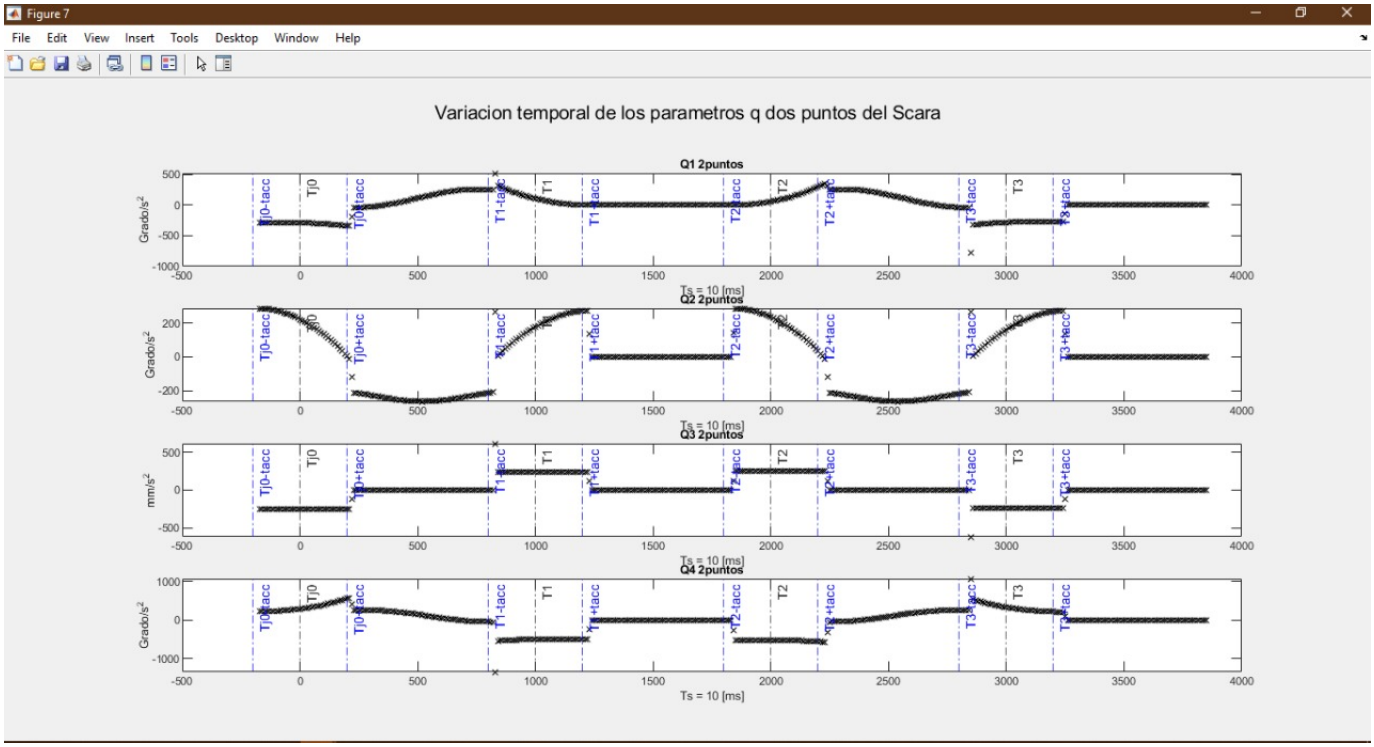


Figura 8: Parámetros Q ddot para movimiento Cartesiano

Al ver los gráficos notamos que ya no se cumple la forma de señal sobre los parámetros q obtenido en la trayectoria *joint*. Esto es esperable puesto que ahora se esta realizando una interpolación sobre los parámetros cartesianos y no sobre los de *joint*.

Se grafica la trayectoria obtenida entre las poses.

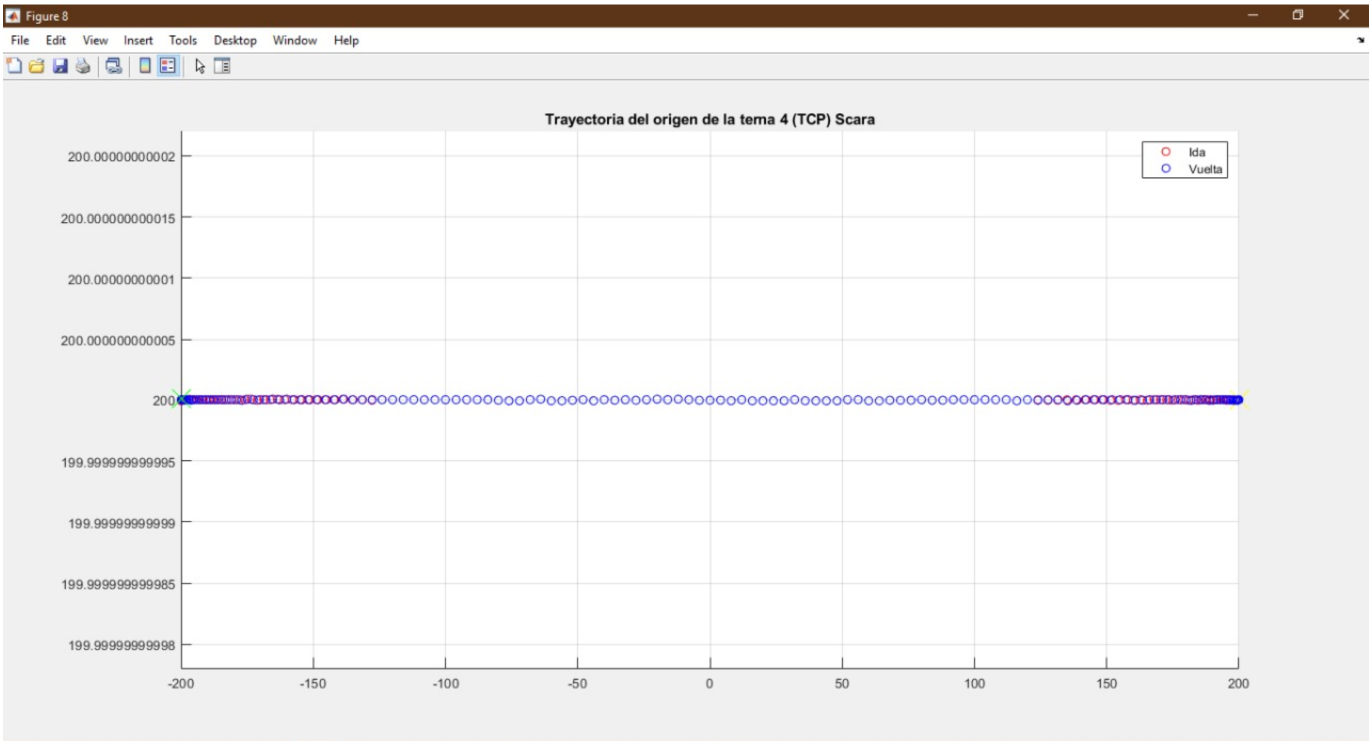


Figura 9: Trayectoria Cartesiana

2. Test

Los tres algoritmos presentados son ejecutado en distintas secciones del archivo ***main.m***, adjunto a continuación.

```

%% En este codigo seran invocadas y probadas las funciones
clear all
close all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%     Trabajo Practico No.4       %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% Generacion de Trayectorias    %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%             2020               %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Punto 1

% Test 1 : Funcionamiento de la funcion -> tp4_punto1.m
% Para testear rellenar q_ref_deg con los valores de q1,q2,q4 en rad y q3
% en mm.
% El test realizara el problema directo para obtener los parametros de
% x0,y0,z0 y obtendra Roll. Se espera que la funcion tp4_punto1 devuelva lo
% mismo que se introdujo en el vector q_ref_deg

%Directo
% q_ref_deg= [q1,q2,q3,q4] <- COMPLETAR CON LO QUE SE DESEE PROBAR

%q_ref_deg=[0,0,-50,0]
%q_ref_deg=[0,0,-50,-90]
q_ref_deg=[0,90,-50,-90]
q_ref_rad=[deg2rad(q_ref_deg(1)),deg2rad(q_ref_deg(2)),-50,deg2rad(q_ref_deg(4))]

% En caso de singularidad el sistema siempre opta por conf=1 (codo +)
[POSE,conf] = Direct_SCARA(q_ref_rad);

%Inverso
x0=POSE(1,4);
y0=POSE(2,4);
z0=POSE(3,4);
Roll = q_ref_deg(1)+ q_ref_deg(2) + q_ref_deg(4) ;

```

```
% Test del punto 1
q = tp4_punto1([x0,y0,z0,Roll],conf);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Punto2
clear all
close all
clc

POSE1 = [-200; 200;-100; 0; 1];
POSE2 = [200; 200;-200; 90; 1];

qa = tp4_punto1(POSE1(1:4),POSE1(5));
qb = tp4_punto1(POSE2(1:4),POSE2(5));

% Entiendo como que el robot tiene que tocar el punto por lo que duplico el
% valor de la posicion de los ejes de manera que "se detenga" en los puntos
Qd = [qa',qa',qb',qb',qa',qa'];
k = 0;
k = k + 1 ;qa = Qd(:,k);
k = k + 1 ;qb = Qd(:,k);
k = k + 1 ;qc = Qd(:,k);

Td =  [1000 1000 1000 1000 1000];
a = 0;
a = a + 1 ;td = Td(a);

tacc = 200;

proccessAll = true;
sameSegment = true;
tseg = - tacc;
n = -tacc;

qOb         = [];
qdotOb      = [];
qddotOb     = [];
N = [];
TjObs = [];

Ts = 10;  %ms

while(proccessAll)

    [q,qdot,qddot,sameSegment,Tj]=tp4_punto2(tseg,qa,qb,qc,td,Ts,tacc);
    tseg = tseg + Ts ;
    n = n + Ts;

    if sameSegment == false
        k = k + 1 ;
        %Cambio de segmento
        if k > length(Qd)
            %No tengo mas segmentos que analizar
            proccessAll = false;
        else
            %Defino nuevo segmento
            qa = q;
            qb = qc;
            qc = Qd(:,k);

            a = a + 1;
            td = Td(a);

            tseg = -tacc + Ts;
            TjObs  = [TjObs, Tj];
        end
    end

% Ploteo en T real dentro del while para debug
% Solo se plotea la variacion angular con las referencias en el tiempo
% transcurrido. Desventaja: Plotear ralentiza la ejecucion del algoritmo

%     figure(1)
%     subplot(4,1,1)
%         hold on
%         scatter(n,rad2deg(qb(1)),'r.');
%         scatter(n,rad2deg(qc(1)),'b.');
%         scatter(n,rad2deg(q(1)),'kx')
```

```
118 %         ylabel('Grados')
119 %         %xlabel('ms')
120 %         %title('q1')
121 %
122 %     subplot(4,1,2)
123 %         hold on
124 %         plot(n,rad2deg(qb(2)),'-r');
125 %         plot(n,rad2deg(qc(2)),'-y');
126 %         plot(n,rad2deg(q(2)),'xk')
127 %         ylabel('Grados')
128 %         %xlabel('ms')
129 %         %title('q2')
130 %
131 %     subplot(4,1,3)
132 %         hold on
133 %         plot(n,qb(3),'xr');
134 %         plot(n,qc(3),'xy');
135 %         plot(n,q(3),'xk')
136 %         ylabel('mm')
137 %         %xlabel('ms')
138 %         %title('q3')
139 %
140 %     subplot(4,1,4)
141 %         hold on
142 %         plot(n,rad2deg(qb(4)),'xr');
143 %         plot(n,rad2deg(qc(4)),'xy');
144 %         plot(n,rad2deg(q(4)),'xk')
145 %         ylabel('Grados')
146 %         %xlabel('ms')
147 %         %title('q4')
148
149 %Almaceno la evolucion de las variables de manera de graficarlas
150 %todas juntas al finalizar el algoritmo. Sin embargo se pueden
151 %graficar en tiempo real descomentando la seccion anterior
152
153  qOb         = [qOb , q]         ;
154  qdotOb      = [qdotOb , qdot]   ;
155  qddotOb     = [qddotOb , qddot];
156  N =[N n];
157
158
159 end
160
161 % Graficos
162
163 figure(1)
164 set(gcf, 'WindowState', 'maximized');
165 subplot(4,1,1)
166     plot(N,rad2deg(qOb(1,:)),'kx')
167     plotBorders(TjObs,tacc)
168     ylabel('Grados')
169     xlabel(['Ts = ',num2str(Ts),' [ms]'])
170     title('Q1')
171 subplot(4,1,2)
172     plot(N,rad2deg(qOb(2,:)),'kx')
173     plotBorders(TjObs,tacc)
174     ylabel('Grados')
175     xlabel(['Ts = ',num2str(Ts),' [ms]'])
176     title('Q2')
177 subplot(4,1,3)
178     plot(N,qOb(3,:),'kx')
179     plotBorders(TjObs,tacc)
180     ylabel('mm')
181     xlabel(['Ts = ',num2str(Ts),' [ms]'])
182     title('Q3')
183 subplot(4,1,4)
184     plot(N,rad2deg(qOb(4,:)),'kx')
185     plotBorders(TjObs,tacc)
186     ylabel('Grados')
187     xlabel(['Ts = ',num2str(Ts),' [ms]'])
188     title('Q4')
189 suptitle('Variacion temporal de los parametros q del Scara')
190
191 figure(2)
192 set(gcf, 'WindowState', 'maximized');
193 subplot(4,1,1)
194     plot(N,rad2deg(qdotOb(1,:)*1000),'kx')
195     plotBorders(TjObs,tacc)
196     ylabel('Grados/s')
197     xlabel(['Ts = ',num2str(Ts),' [ms]'])
```

```
    title('Q1 punto')
subplot(4,1,2)
    plot(N,rad2deg(qdotOb(2,:)*1000),'kx')
    plotBorders(TjObs,tacc)
    ylabel('Grados/s')
    xlabel(['Ts = ',num2str(Ts),' [ms]'])
    title('Q2 punto')
subplot(4,1,3)
    plot(N,qdotOb(3,:)*1000,'kx')
    plotBorders(TjObs,tacc)
    ylabel('mm/s')
    xlabel(['Ts = ',num2str(Ts),' [ms]'])
    title('Q3 punto')
subplot(4,1,4)
    plot(N,rad2deg(qdotOb(4,:)*1000),'kx')
    plotBorders(TjObs,tacc)
    ylabel('Grado/s')
    xlabel(['Ts = ',num2str(Ts),' [ms]'])
    title('Q4 punto')
suptitle('Variacion temporal de los parametros q punto del Scara')


figure(3)
set(gcf, 'WindowState', 'maximized');
subplot(4,1,1)
    plot(N,rad2deg(qddotOb(1,:)*1000^2),'kx')
    plotBorders(TjObs,tacc)
    ylabel('Grado/s^2')
    xlabel(['Ts = ',num2str(Ts),' [ms]'])
    title('Q1 2puntos')
subplot(4,1,2)
    plot(N,rad2deg(qddotOb(2,:)*1000^2),'kx')
    plotBorders(TjObs,tacc)
    ylabel('Grado/s^2')
    xlabel(['Ts = ',num2str(Ts),' [ms]'])
    title('Q2 2puntos')
subplot(4,1,3)
    plot(N,qddotOb(3,:)*1000^2,'kx')
    plotBorders(TjObs,tacc)
    ylabel('mm/s^2')
    xlabel(['Ts = ',num2str(Ts),' [ms]'])
    title('Q3 2puntos')
subplot(4,1,4)
    plot(N,rad2deg(qddotOb(4,:)*1000^2),'kx')
    plotBorders(TjObs,tacc)
    ylabel('Grado/s^2')
    xlabel(['Ts = ',num2str(Ts),' [ms]'])
    title('Q4 2puntos')
suptitle('Variacion temporal de los parametros q dos puntos del Scara')


figure(4)
set(gcf, 'WindowState', 'maximized');
for k =1:length(qOb)
    q = qOb(:,k);
    [A,conf]=Direct_SCARA(q);
    px = A(1,4);
    py = A(2,4);
    hold on
    if  abs(px - 200)<0.001 && abs(py - 200)<0.001
        plot(px,py,'yx','MarkerSize',20)
    elseif abs(px - (-200))<0.001  && abs(py - 200)<0.001
        plot(px,py,'gx','MarkerSize',20)
    else
        if k < floor(length(qOb)/2)
            p1=plot(px,py,'ro'); %Ida
        else
            p2=plot(px,py,'bo'); %Vuelta
        end
    end
end
    grid on
    title('Trayectoria del origen de la terna 4 (TCP) Scara')
    legend([p1,p2],{'Ida','Vuelta'})

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Punto 3
clear all
```

```
%close all
clc


POSE1 = [-200; 200;-100; 0; 1];
POSE2 = [200; 200;-200; 90; 1];

qa = tp4_punto1(POSE1(1:4),POSE1(5));
qb = tp4_punto1(POSE2(1:4),POSE2(5));

Ja = getJacobianSCARA(qa,[200,200]);
Jb = getJacobianSCARA(qb,[200,200]);

vq1max = 90;
vq2max = 180;
vq3max = 1000;
vq4max = 360;
vmax = [vq1max,vq2max,vq3max,vq4max]';
vmaxrad = [deg2rad(vmax(1:2))',vq3max,deg2rad(vq4max)]';

maxA = Ja *vmaxrad;
maxB = Jb *vmaxrad;
maxCartesianV = abs(min(maxA,maxB));



[POSEA,conf] = Direct_SCARA(qa);
[POSEB,conf] = Direct_SCARA(qb);

Qd = {POSEA,POSEA,POSEB,POSEB,POSEA,POSEA};
k = 0;
k = k + 1 ;A = Qd{1};
k = k + 1 ;B = Qd{k};
k = k + 1 ;C = Qd{k};

Td =  [1000 1000 1000 1000 1000];
a = 0;
a = a + 1 ;td = Td(a);

tacc = 200;

proccessAll = true;
sameSegment = true;
tseg = - tacc;
n = -tacc;

qOb         = [];
qdotOb      = [];
qddotOb     = [];
N = [];
TjObs = [];

Ts = 10; %ms

qant    =  10;
qdotant =  100;

while(proccessAll)

    [q,qdot,qddot,sameSegment,POSEActual,Tj]=tp4_punto3(tseg,A,B,C,td,Ts,tacc,qant,qdotant,
        maxCartesianV);
    tseg = tseg + Ts ;
    n = n + Ts;

    qant    =  q;
    qdotant =  qdot;

    if sameSegment == false
        k = k + 1 ;
        %Cambio de segmento
        if k > length(Qd)
            %No tengo mas segmentos que analizar
            proccessAll = false;
        else
            %Defino nuevo segmento
            A = POSEActual;
            B = C;
            C = Qd{k};

            a = a + 1;
```

```
357             td = Td(a);
358
359             tseg = -tacc + Ts;
360             TjObs  = [TjObs, Tj];
361         end
362     end
363
364 % Ploteo en T real dentro del while para debug
365 % Solo se plotea la variacion angular con las referencias en el tiempo
366 % transcurrido. Desventaja: Plotear ralentiza la ejecucion del algoritmo
367
368 %     figure(1)
369 %     subplot(4,1,1)
370 %         hold on
371 %         scatter(n,rad2deg(qb(1)),'r.');
372 %         scatter(n,rad2deg(qc(1)),'b.');
373 %         scatter(n,rad2deg(q(1)),'kx')
374 %         ylabel('Grados')
375 %         %xlabel('ms')
376 %         %title('q1')
377 %
378 %     subplot(4,1,2)
379 %         hold on
380 %         plot(n,rad2deg(qb(2)),'-r');
381 %         plot(n,rad2deg(qc(2)),'-y');
382 %         plot(n,rad2deg(q(2)),'xk')
383 %         ylabel('Grados')
384 %         %xlabel('ms')
385 %         %title('q2')
386 %
387 %     subplot(4,1,3)
388 %         hold on
389 %         plot(n,qb(3),'xr');
390 %         plot(n,qc(3),'xy');
391 %         plot(n,q(3),'xk')
392 %         ylabel('mm')
393 %         %xlabel('ms')
394 %         %title('q3')
395 %
396 %     subplot(4,1,4)
397 %         hold on
398 %         plot(n,rad2deg(qb(4)),'xr');
399 %         plot(n,rad2deg(qc(4)),'xy');
400 %         plot(n,rad2deg(q(4)),'xk')
401 %         ylabel('Grados')
402 %         %xlabel('ms')
403 %         %title('q4')
404
405 % Almaceno la evolucion de las variables de manera de graficarlas
406 % todas juntas al finalizar el algoritmo. Sin embargo se pueden
407 % graficar en tiempo real descomentando la seccion anterior
408
409  qOb         = [qOb , q']          ;
410  qdotOb      = [qdotOb , qdot']    ;
411  qddotOb     = [qddotOb , qddot'];
412  N =[N n];
413
414
415 end
416
417 % Graficos
418
419 figure(5)
420 set(gcf, 'WindowState', 'maximized');
421 subplot(4,1,1)
422     plot(N,rad2deg(qOb(1,:)),'kx')
423     plotBorders(TjObs,tacc)
424     ylabel('Grados')
425     xlabel(['Ts = ',num2str(Ts),' [ms]'])
426     title('Q1')
427 subplot(4,1,2)
428     plot(N,rad2deg(qOb(2,:)),'kx')
429     plotBorders(TjObs,tacc)
430     ylabel('Grados')
431     xlabel(['Ts = ',num2str(Ts),' [ms]'])
432     title('Q2')
433 subplot(4,1,3)
434     plot(N,qOb(3,:),'kx')
435     plotBorders(TjObs,tacc)
436     ylabel('mm')
```

```
437     xlabel(['Ts = ',num2str(Ts),' [ms]'])
438     title('Q3')
439 subplot(4,1,4)
440     plot(N,rad2deg(qOb(4,:)),'kx')
441     plotBorders(TjObs,tacc)
442     ylabel('Grados')
443     xlabel(['Ts = ',num2str(Ts),' [ms]'])
444     title('Q4')
445 suptitle('Variacion temporal de los parametros q del Scara')
446
447 figure(6)
448 set(gcf, 'WindowState', 'maximized');
449 subplot(4,1,1)
450     plot(N(2:end),rad2deg(qdotOb(1,2:end)*1000),'kx')
451     plotBorders(TjObs,tacc)
452     ylabel('Grados/s')
453     xlabel(['Ts = ',num2str(Ts),' [ms]'])
454     title('Q1 punto')
455 subplot(4,1,2)
456     plot(N(2:end),rad2deg(qdotOb(2,2:end)*1000),'kx')
457     plotBorders(TjObs,tacc)
458     ylabel('Grados/s')
459     xlabel(['Ts = ',num2str(Ts),' [ms]'])
460     title('Q2 punto')
461 subplot(4,1,3)
462     plot(N(2:end),qdotOb(3,2:end)*1000,'kx')
463     plotBorders(TjObs,tacc)
464     ylabel('mm/s')
465     xlabel(['Ts = ',num2str(Ts),' [ms]'])
466     title('Q3 punto')
467 subplot(4,1,4)
468     plot(N(2:end),rad2deg(qdotOb(4,2:end)*1000),'kx')
469     plotBorders(TjObs,tacc)
470     ylabel('Grado/s')
471     xlabel(['Ts = ',num2str(Ts),' [ms]'])
472     title('Q4 punto')
473 suptitle('Variacion temporal de los parametros q punto del Scara')
474
475
476 figure(7)
477 set(gcf, 'WindowState', 'maximized');
478 subplot(4,1,1)
479     plot(N(3:end),rad2deg(qddotOb(1,3:end)*1000^2),'kx')
480     plotBorders(TjObs,tacc)
481     ylabel('Grado/s^2')
482     xlabel(['Ts = ',num2str(Ts),' [ms]'])
483     title('Q1 2puntos')
484 subplot(4,1,2)
485     plot(N(3:end),rad2deg(qddotOb(2,3:end)*1000^2),'kx')
486     plotBorders(TjObs,tacc)
487     ylabel('Grado/s^2')
488     xlabel(['Ts = ',num2str(Ts),' [ms]'])
489     title('Q2 2puntos')
490 subplot(4,1,3)
491     plot(N(3:end),qddotOb(3,3:end)*1000^2,'kx')
492     plotBorders(TjObs,tacc)
493     ylabel('mm/s^2')
494     xlabel(['Ts = ',num2str(Ts),' [ms]'])
495     title('Q3 2puntos')
496 subplot(4,1,4)
497     plot(N(3:end),rad2deg(qddotOb(4,3:end)*1000^2),'kx')
498     plotBorders(TjObs,tacc)
499     ylabel('Grado/s^2')
500     xlabel(['Ts = ',num2str(Ts),' [ms]'])
501     title('Q4 2puntos')
502 suptitle('Variacion temporal de los parametros q dos puntos del Scara')
503
504
505 figure(8)
506 set(gcf, 'WindowState', 'maximized');
507 for k =1:length(qOb)
508     q = qOb(:,k);
509     [A,conf]=Direct_SCARA(q);
510     px = A(1,4);
511     py = A(2,4);
512     hold on
513     if  abs(px - 200)<0.0001 && abs(py - 200)<0.0001
514         plot(px,py,'yx','MarkerSize',20)
515     elseif abs(px - (-200))<0.0001  && abs(py - 200)<0.0001
516         plot(px,py,'gx','MarkerSize',20)
```

```
517     else
518         if k < floor(length(qOb)/2)
519             p1=plot(px,py,'ro'); %Ida
520         else
521             p2=plot(px,py,'bo'); %Vuelta
522         end
523     end
524 end
525     grid on
526     title('Trayectoria del origen de la terna 4 (TCP) Scara')
527     legend([p1,p2],{'Ida','Vuelta'})
528
529
530 %% Funciones Auxiliares para ploteo
531  function plotBorders(TjObs,tacc)
532     hold on
533         xline(-tacc,'b-.','Tj0-tacc','DisplayName','Tj1');
534     xline(0,'-.','Tj0','DisplayName','Tj1');
535     xline(tacc,'b-.','Tj0+tacc','DisplayName','Tj1');
536     sum =0;
537     for k=1:length(TjObs)
538         sum = TjObs(k)+ sum;
539         xline(sum-tacc,'b-.',['T',num2str(k),'-tacc'],'DisplayName','Tj1');
540         xline(sum,'-.',['T',num2str(k)],'DisplayName','Tj1');
541         xline(sum+tacc,'b-.',['T',num2str(k),'+tacc'],'DisplayName','Tj1');
542     end
543
544  end
```

3. Conclusiones

Como conclusiones se observa que ambos algoritmos generan trayectorias suaves permitiendo que el robot realice tanto movimientos tipo Joint como Cartesianos de forma adecuada.

La interpolación Joint tiene la ventaja que permite controlar las variables articulares (posición, velocidad y aceleración angular), siendo estas las mas relacionadas a la maquinaria y a los motores. Por lo que utilizar movimientos tipo joint permitiría, en una primera instancia, cuidar la vida útil del robot. La desventaja yase en que en la realidad no siempre se requieren movimientos de tipo joint.

La interpolación Cartesiana resuelve el problema de los movimiento lineales, permitiendo independizarse de los motores y hacer actividades que requieran movimientos lineales, por ej., pintar, apilar, entre otras. Sin embargo, no se tiene tanto control sobre las velocidades de los motores ni las aceleraciones. Puede observarse en las velocidades angulares existe dos zonas en donde la velocidad supera el máximo permitido, gastando la vida útil de los motores.

En el algoritmo propuesto se obtiene las velocidades máximas cartesianas en las poses de inicio y de destino, pero eso no garantiza que en alguna pose intermedia las velocidades máximas sean superiores, por lo que en una primera instancia seria necesario aplicar el jacobiano para cada punto de la trayectoria para obtener el Tj previo a la ejecución del algoritmo de interpolación cartesiano

4. Anexo

Trabajo Práctico No.4

Generación de Trayectorias

2020

1. Problema Cinemático Inverso

Sea el robot de la figura 1 con arquitectura *SCARA* (ejercicio 3 de la tira 1) y parámetros cinemáticos:

$$\begin{aligned} a_1 &= 200\texttt{mm} \\ a_2 &= 200\texttt{mm} \end{aligned}$$

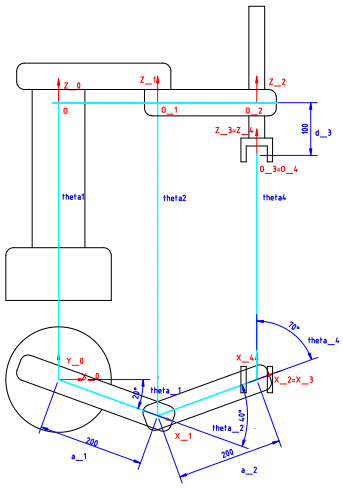


Figura 1: Robot R-R-P-R

Escribir una función en octave/matlab que calcule las variables *joint* $\mathbf{q}$ a partir de una `POSE` ingresada como parámetro según las siguientes definiciones:

$$\begin{aligned} \mathbf{q} &= [\theta_1, \theta_2, d_3, \theta_4]^T \\ \texttt{POSE}^0 &= [x_0, y_0, z_0, \text{Roll}, \text{conf}]^T \end{aligned}$$

donde la posición se expresa en `mm` y la orientación Roll de la herramienta en grados.

Considerar que el robot cuenta con topes mecánicos en los ejes 2 y 3 que imponen las siguientes restricciones en las variables *joint*:

$$\begin{aligned} -150^\circ \leq \quad & q_2 \quad \leq 150^\circ \\ -250\mathtt{mm} \leq \quad & q_3 \quad \leq -50\mathtt{mm} \end{aligned}$$

2. Generador de Trayectorias Joint

Para el robot de la sección 1 escribir un programa en octave/matlab que genere el movimiento *joint* entre posiciones objetivo.
El tiempo de aceleración y las velocidades máximas de los ejes son:

$$\begin{aligned} t_{acc} &= 200\mathtt{ms} \\ v_{1max} &= 90^\circ/\mathtt{s} \\ v_{2max} &= 180^\circ/\mathtt{s} \\ v_{3max} &= 1000\mathtt{mm}/\mathtt{s} \\ v_{4max} &= 360^\circ/\mathtt{s} \end{aligned}$$

Tener en cuenta las siguientes consideraciones:

1. Las entradas al programa deben ser las POSEs por las que debe pasar la herramienta y el tiempo deseado de cada movimiento.
2. Las salidas del programa deben ser:
   - gráficas de la evolución en el tiempo de las posiciones *joint*, sus velocidades y aceleraciones
   - un gráfico de la trayectoria del origen de la terna 4 (TCP) proyectada en el plano $[\mathbf{X}_0, \mathbf{Y}_0]$
3. Pensar el algoritmo para que pueda funcionar en un sistema de tiempo real.

Como ejemplo del programa desarrollado, imprimir las salidas para el movimiento realizado a máxima velocidad donde el robot parte de reposo desde $\mathtt{POSE}_1$ pasa por $\mathtt{POSE}_2$ y regresa a $\mathtt{POSE}_1$ deteniéndose. Las POSE*s* están expresadas en la terna 0 y definen a continuación:

$$\begin{aligned} \mathtt{POSE}_1 &= [-200, 200, -100, 0, 1]^T \\ \mathtt{POSE}_2 &= [200, 200, -200, 90, 1]^T \end{aligned}$$

3. Generador de Trayectorias Cartesiano

Implementar un generador de trayectorias cartesiano de manera de repetir el punto anterior con un movimiento en línea recta. Mostrar en un gráfico la evolución temporal de las variables articulares, sus velocidades y aceleraciones. Graficar también la trayectoria vista desde la terna 0.

Mostrar en forma comparada los gráficos de evolución temporal y de trayectoria del movimiento *joint* y del cartesiano.