ROBÓTICA INDUSTRIAL

Práctica 1:Introducción a Matlab, Análisis cinemático del manipulador RR

Juan Antonio Aldea Armenteros

19 de junio de 2012

1. Problema Cinemático Directo

El problema cinemático directo consiste en determinar, a partir de las variables de las articulaciones, la posición del efector final en el espacio. Un video de la simulación puede verse aquí.

1.1. Problema Cinemático Inverso

Este problema consiste en determinar las variables de articulación a partir del punto del espacio que se desea alcanzar con el robot. En esta práctica las variables de articulación están expresadas como ángulos, esto lleva consigo muchos problemas de continuidad como ya se avisa en el enunciado, sería mucho más conveniente expresarlos mediante cuaterniones cuya interpolación, composición y operatoria en general es mucho más homogénea y estable que las equivalentes con ángulos de Euler. Un video de la simulación de las trayectorias indicadas en el enunciado puede verse aquí.

1.1.1. Características temporales de las trayectorias

A la vista de las representaciones de las derivadas de las trayectorias podemos concluir que:

- Primera trayectoria 1.1.1 y 1.1.2, es bastante suave, sin embargo tiene unas zonas con una gran pendiente, no son aptas para ser usadas en una máquina real.
- Segunda trayectoria 1.1.3 y 1.1.4, es mucho más suave que la anterior, también tiene zonas de gran cambio (esto es relativo a la vista de los órdenes de magnitud) pero los cambios son suaves.
- Tercera trayectoria 1.1.5 y 1.1.6, es realmente buena, todas las variaciones son muy suaves y los órdenes de magnitud de estas son pequeños.

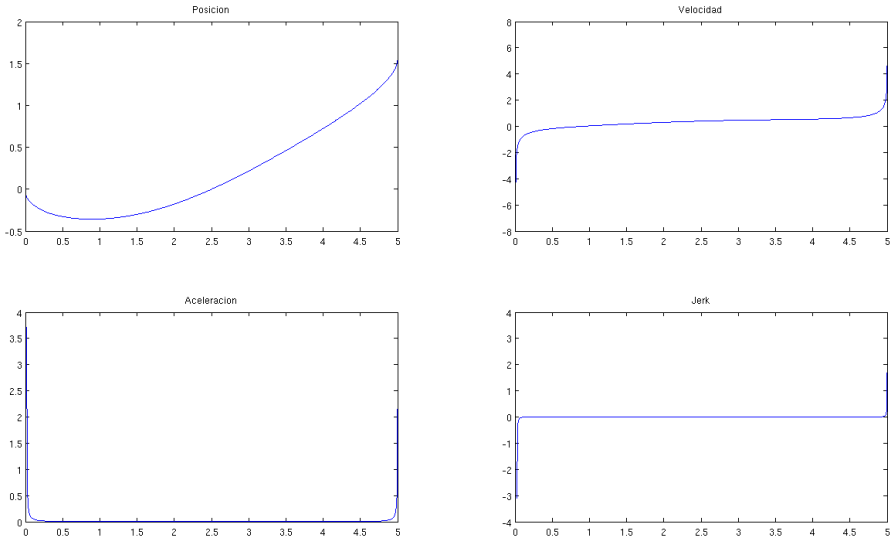

Figura 1.1.1.: Trayectoria 1, $\theta_1(t)$.

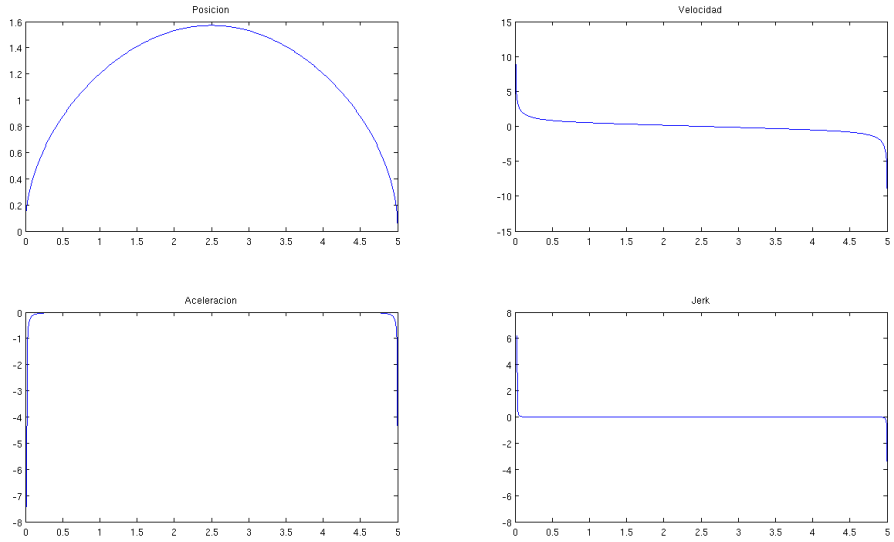

Figura 1.1.2.: Trayectoria 1, $\theta_2(t)$.

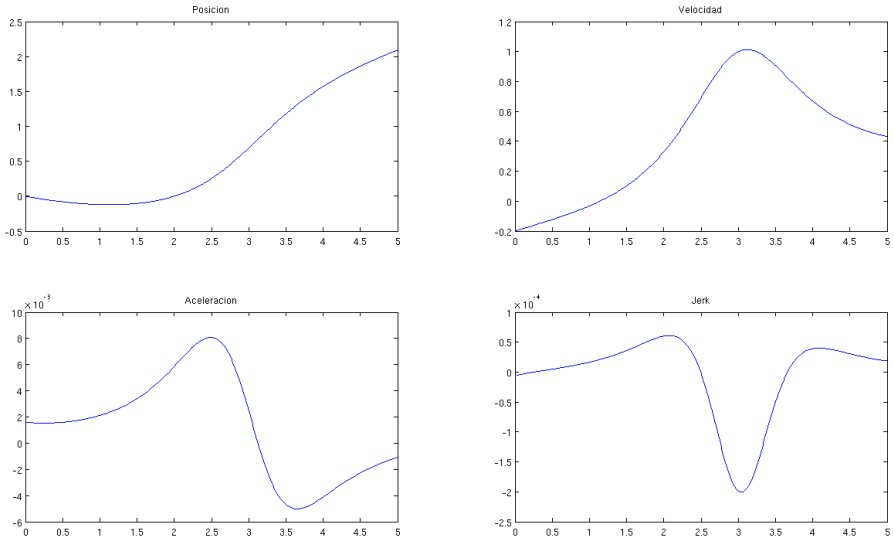

Figura 1.1.3.: Trayectoria 2, $\theta_1(t)$.

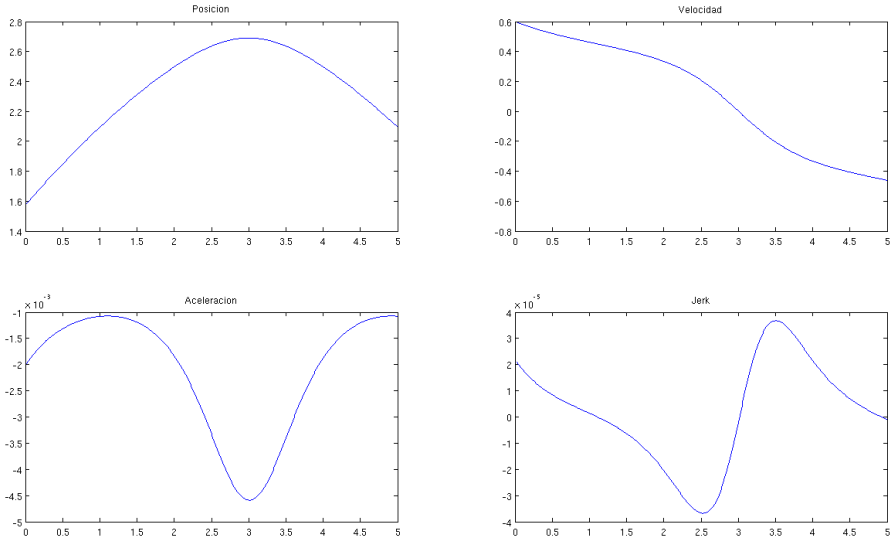

Figura 1.1.4.: Trayectoria 2, $\theta_2(t)$.

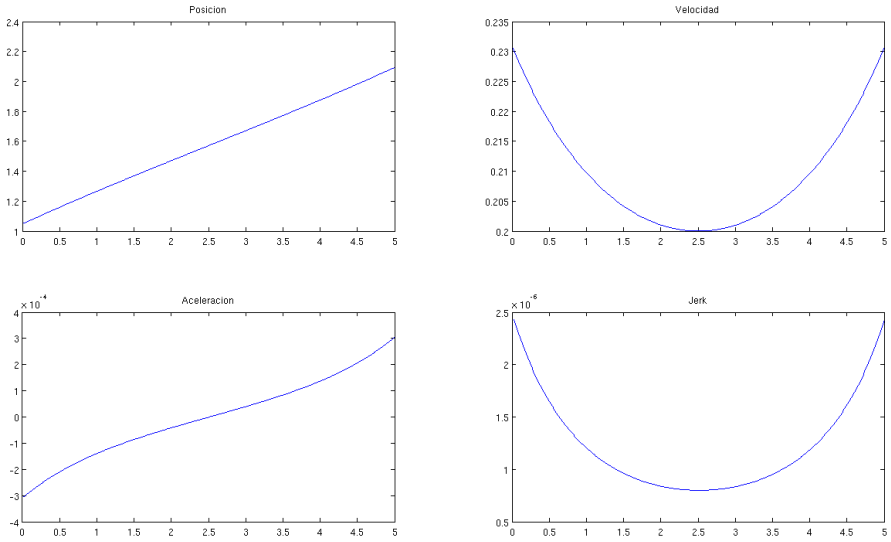

Figura 1.1.5.: Trayectoria 3, $\theta_1(t)$.

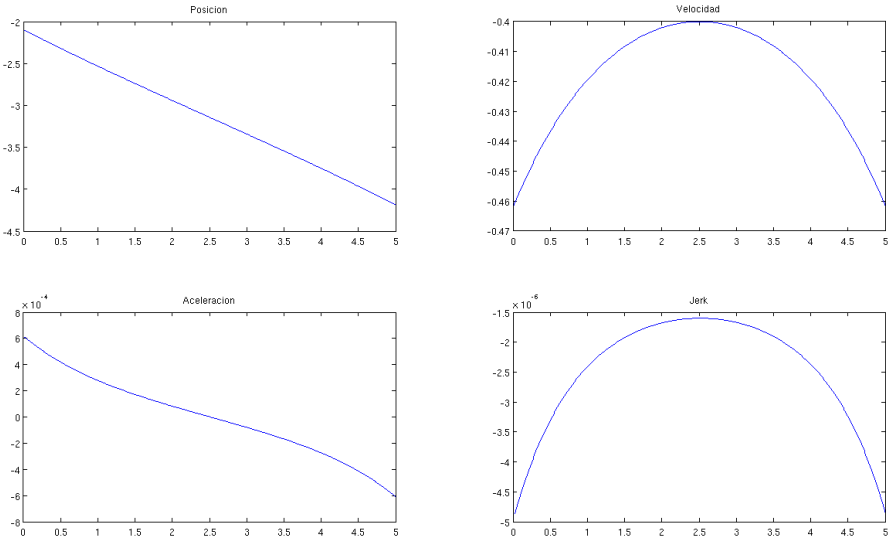

Figura 1.1.6.: Trayectoria 3, $\theta_2(t)$.

APÉNDICES

A. CÓDIGO MATLAB UTILIZADO

Listing A.1: PCD

```
function p = pcd(L1, L2, th1, th2)
    px = L1 .* cos(th1) + L2 .* cos(th1 + th2);
    py = L1 .* sin(th1) + L2 .* sin(th1 + th2);
    p = [px; py];
```

Listing A.2: PCI

```
function angulos = pci(L1, L2, p)
    x = p(1);
    y = p(2);
    th2 = acos((x^2 + y^2 - L1^2 - L2^2)/(2*L1*L2));
    th2 = [th2, -th2];
    th1 = atan2(y * (L1 + L2 * cos(th2)) - x * L2 * sin(th2),...
        x * (L1 + L2 * cos(th2)) + y * L2 * sin(th2));
    angulos =[th1; th2];
```

Listing A.3: Trayectoria PCI

```
function angulos = trayectoria_pci(L1, L2, p0, pn, tiempo, T, animar)
    x0 = p0(1);
    y0 = p0(2);
    xn = pn(1);
    yn = pn(2);
    pasos = tiempo/T;
    x = x0:(xn - x0)/pasos:xn;
    y = y0:(yn - y0)/pasos:yn;

    %angulos correspondientes a la posicion de partida
    angulos_iniciales = pci(L1, L2, p0);

    %elegimos la solucion que tiene el th1 positivo
    if angulos_iniciales(1) >= 0
        angulos_iniciales = angulos_iniciales(:, 1);
    else
         angulos_iniciales = angulos_iniciales(:, 2);
    end

    angulos = zeros(2, pasos + 1);
```

```
21
22      %display('angulos finales');
23      %display(angulos_iniciales*180/pi);
24
25      %como se han creado tantos puntos como pasos hay entre los puntos
26      %puede ser que un angulo tenga menos pasos que el otro porque su
27      %movimiento termino antes, en ese caso los valores que faltan son
28      %iguales al ultimo valor.
29
30      if numel(x) < pasos + 1
31          x = [x, xn * ones(1, pasos - numel(x)) + 1];
32      end
33
34      if numel(y) < pasos + 1
35          y = [y, yn * ones(1, pasos - numel(y) + 1)];
36      end
37
38      %los primeros angulos son los angulos iniciales
39      angulos(:, 1) = angulos_iniciales;
40      for i = 2:pasos + 1
41          angulos_i = pci(L1, L2, [x(i), y(i)]);
42          %elige la solucion mas proxima a la anterior, evita saltos.
43          if abs(angulos_i(1) - angulos(1, i-1)) <...
44                  abs(angulos_i(3) - angulos(1, i-1))
45              angulos(:, i) = angulos_i(:, 1);
46          else
47              angulos(:, i) = angulos_i(:, 2);
48          end
49
50          if angulos_i(1) == 0 && angulos_i(3) == 0 &&...
51                  sum(angulos(:, 1) ~= angulos(:, end)) > 0
52              posibilidades = [0, pi/2, pi, 4/3*pi, 2*pi];
53              th1_anterior = ones(size(posibilidades)) * angulos(1, i-1);
54              diferencias = abs(posibilidades - th1_anterior);
55              [~, indice] = min(diferencias);
56              angulos(1, i) = posibilidades(indice);
57          end
58
59          if abs(angulos(2,i) - angulos(2, i - 1)) > pi
60              v_anterior = [sin(angulos(2, i-1)), cos(angulos(2, i-1))];
61              v_actual = [sin(angulos(2, i)), cos(angulos(2, i))];
62              diferencia = acos(dot(v_anterior, v_actual));
63
64              angulos(2, i) = angulos(2, i-1) + ...
65                  sign(angulos(2, i-1)) .* diferencia;
66
67              angulos(2,i)=mod(angulos(2,i), sign(angulos(2, i))*2*pi);
68          end
69      end
```

```
70
71      p = pcd(1, 1, angulos(1,:), angulos(2, :));
72      if animar == true
73          for i = 1:pasos + 1
74              plot(p(1, 1:i), p(2, 1:i));
75              hold(gca, 'on');
76              robot(1, angulos(1, i), p(:,i));
77              hold(gca, 'off');
78              pause(T);
79          end
80      end
81  end
```

Listing A.4: Trayectoria PCD

```
1   function trayectoria_pcd(duracion, periodo)
2   n_pasos = duracion / periodo;
3   %n_pasos = 1000;
4
5   th1 = 0:(pi/2)/n_pasos:pi/2;
6   th2 = 0:(pi/2)/n_pasos:pi/2;
7
8   p = pcd(1, 1, th1, th2);
9
10  for i = 1:n_pasos + 1
11      plot(p(1, 1:i), p(2, 1:i));
12      hold(gca, 'on');
13      robot(1, th1(i), p(:,i));
14      hold(gca,'off');
15      pause(periodo);
16  end
```

Listing A.5: Script PCI

```
1   animar = false;
2   duracion = 5;
3   T = 0.01;
4
5   angulos1 = trayectoria_pci(1, 1, [2; 0], [ 0; 2], duracion, T, animar);
6   angulos2 = trayectoria_pci(1, 1, [1; 1], [-1; 0], duracion, T, animar);
7   angulos3 = trayectoria_pci(1, 1, [1; 0], [-1; 0], duracion, T, animar);
8
9   angulos = angulos3;
10  tiempo = 0:T:duracion;
11  variable = 2;
12
13  subplot(2, 2, 1),
14      plot(tiempo, angulos(variable, :));
15      title('Posicion')
16
```

```
subplot(2, 2, 2),
    plot(tiempo(2:end), diff(angulos(variable, :), 1)./T);
    title('Velocidad')

subplot(2, 2, 3),
    plot(tiempo(3:end), diff(angulos(variable, :), 2)./T);
    title('Aceleracion');

subplot(2, 2, 4),
    plot(tiempo(4:end), diff(angulos(variable, :), 3)./T);
    title('Jerk');
```