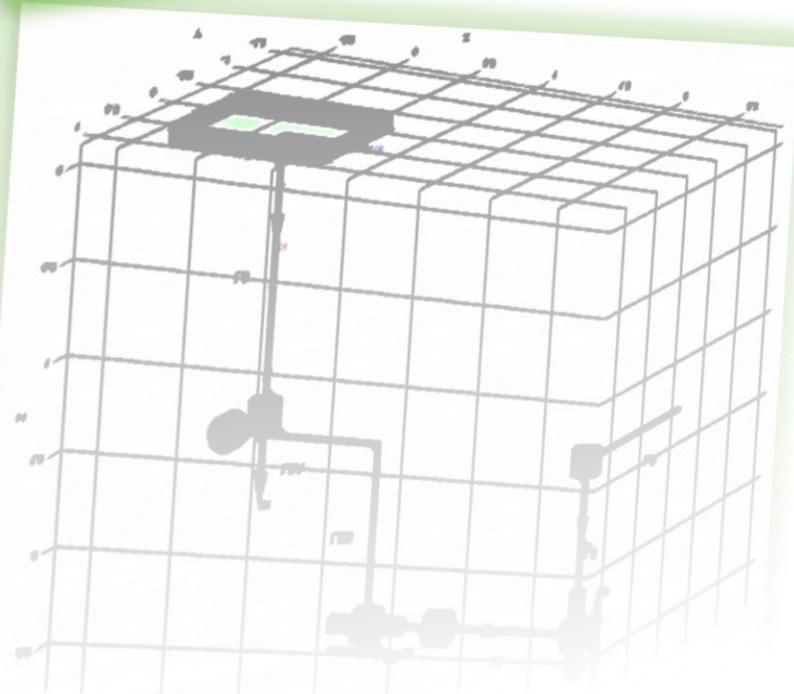
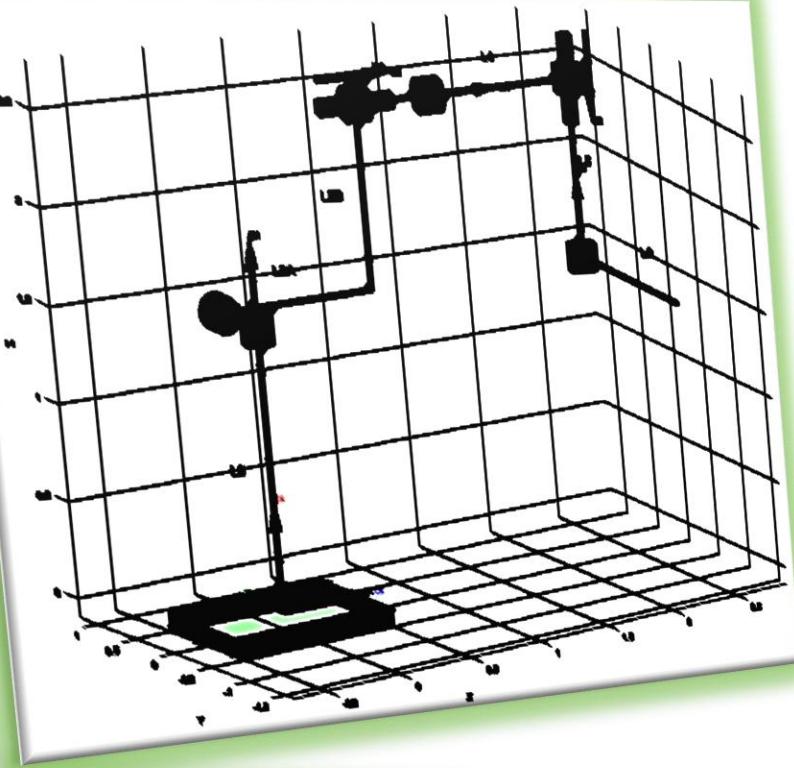




TRABAJO DE CURSO



FUNDAMENTOS DE ROBÓTICA



ÁLVARO GARCÍA LORA
MEMORIA TRABAJO FINAL
3º GIERM

ÍNDICE

0) INTRODUCCIÓN.....	1
1) ANÁLISIS CINEMÁTICO.....	2
1. Parámetros de Denavit-Hartenberg.....	2
2. Matrices de Transformación Homogéneas	3
3. Cinemática Directa	4
4. Cinemática Inversa	10
5. Trayectoria Circular	15
6. Jacobianos directos e inversos	18
2) ANÁLISIS DINÁMICO	20
1. Parámetros y Modelo Dinámico.....	20
1.1. Parámetros Dinámicos.....	20
1.2. Modelo Dinámico.....	22
2. Simulador de dinámica	23
3. Comprobación dinámica	24
3) CONTROL CINEMÁTICO	26
1. Generador de trayectorias	26
4) CONTROL DINÁMICO	29
1. Diseño controlador PD	29
2. Diseño controlador PID.....	30
3. Diseño controlador por Par Calculado	31
4. Resultados del control	31
4.1. Controlador PD.....	33
4.2. Controlador PID.....	43
4.3. Controlador por Par Calculado	47
4.4. Velocidades de movimiento	52
4.5. Conclusiones.....	54

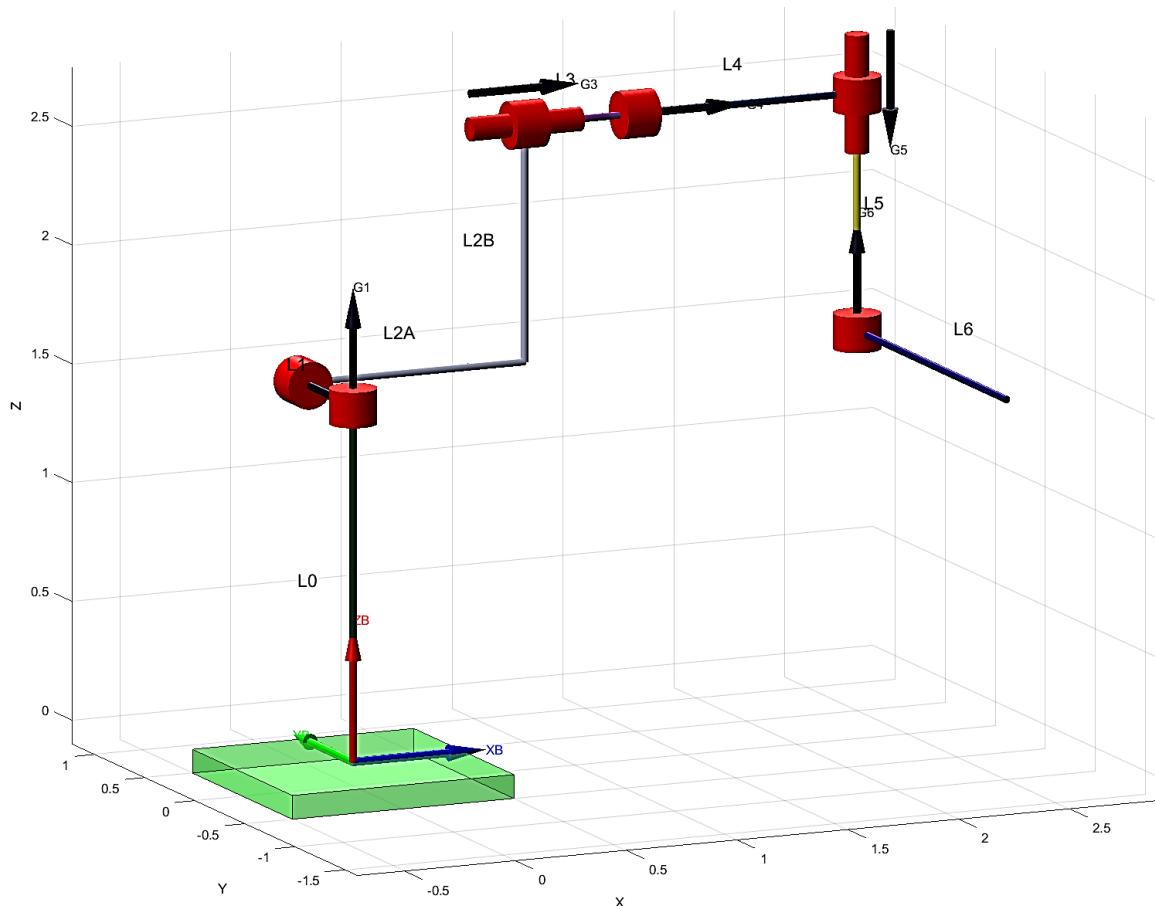


O) INTRODUCCIÓN

El objetivo del presente trabajo es tratar de forma práctica los tres aspectos principales que componen los contenidos teóricos de la asignatura Fundamentos de Robótica perteneciente al tercer curso del Grado en Ingeniería Electrónica, Robótica y Mecatrónica.

En este sentido, partiendo de un robot manipulador de 6 grados de libertad dado, se realizará el análisis cinemático, análisis dinámico y control de éste. Por cuestión de simplicidad se trabajará mayormente con sus tres primeras articulaciones, reduciendo el problema a 3 grados de libertad.

La obtención de la cinemática directa e inversa del robot resulta de vital importancia para conseguir tener una relación inmediata entre la posición del efecto final del manipulador y el valor necesario de sus articulaciones para conseguirlo. El siguiente paso será lograr conseguir un modelo que describa de manera bastante aproximada la dinámica de nuestro sistema. Una vez alcanzado ese punto, tendremos la tarea de mover el extremo del robot entre dos posiciones, tratándose de un problema de seguimiento de trayectorias. Desarrollaremos el control cinemático que genere dicha trayectoria y veremos diversas técnicas de control aplicándolas al conjunto, de forma que se cumplan los requisitos establecidos con la mayor calidad y el menor error cometido posible.

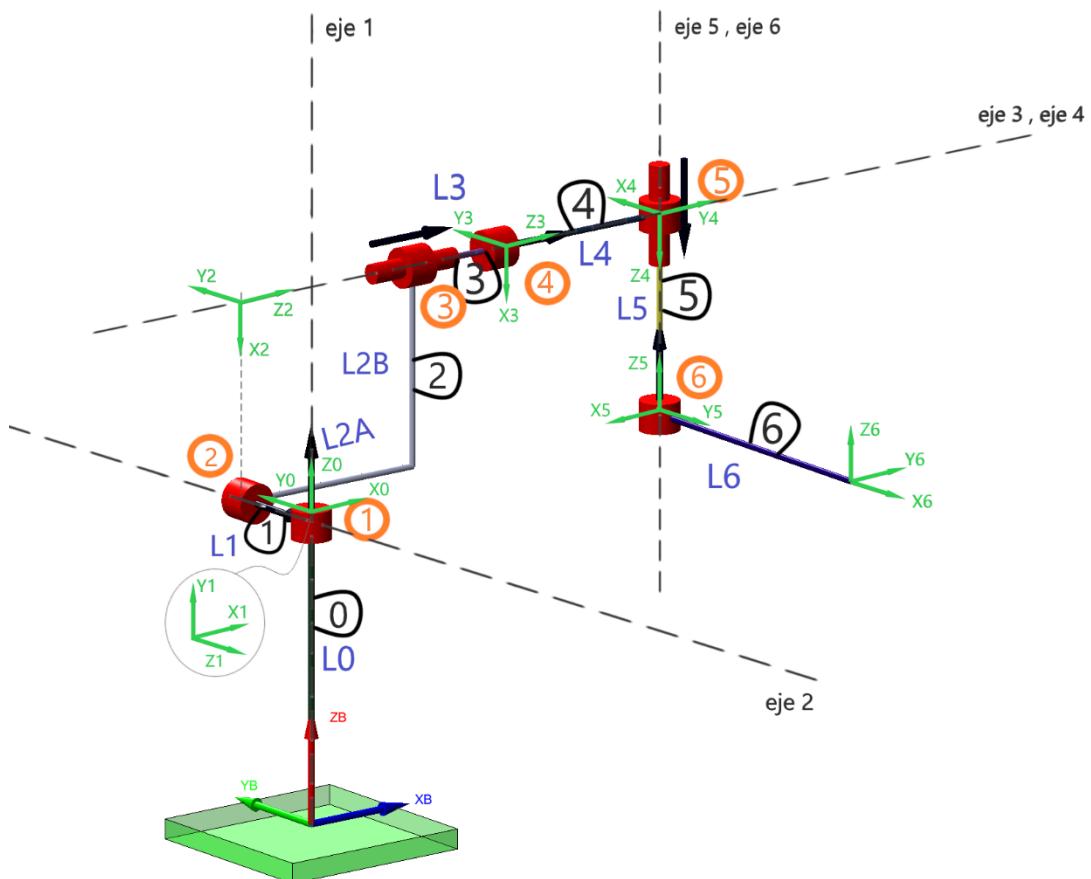


1) ANÁLISIS CINEMÁTICO

1. Parámetros de Denavit-Hartenberg

Aplicando el algoritmo del método de Denavit-Hartenberg podemos obtener los parámetros característicos de cada una de las articulaciones, en este caso, del robot de 6 grados de libertad.

A continuación, se muestra el esquema de la configuración del robot bajo estudio en su posición HOME con los sistemas de ejes correspondientes, así como la tabla del manipulador completo:



ARTICULACIÓN	θ_i (rad)	d_i	a_i	α_i (rad)
0	0	L0	0	0
1	$q_1 + 0$	0	0	$\pi/2$
2	$q_2 - \pi/2$	-L1	-L2B	- $\pi/2$
3	0	$q_3 + L2A + L3$	0	0
4	$q_4 + \pi/2$	L4	0	$\pi/2$
5	$-\pi/2$	$q_5 + L5$	0	π
6	$q_6 + \pi/2$	0	L6	0

Tabla D-H Robot 6 grados de libertad

El fichero de verificación numérica [tablaDH.m](#) ha sido completado con los parámetros recogidos en la tabla anterior.

2. Matrices de Transformación Homogéneas

A partir de los parámetros de las distintas articulaciones obtenidos en el apartado anterior podemos obtener las matrices de transformación homogéneas correspondientes a cada par de articulaciones consecutivas de nuestro robot. El código que se ha empleado para calcular dichas MTH es el siguiente:

```

syms L0 L1 L1A L1B L2 L2A L2B L3 L3A L3B L4 L4A L4B L5 L5A L5B L6 L6A L6B real
syms q1 q2 q3 q4 q5 q6
PI = sym(pi);

%Parámetros de DH
theta0=0; d0=L0; a0=0; alfa0=0;
theta1=q1; d1=0; a1=0; alfa1=PI/2;
theta2=q2-PI/2; d2=-L1; a2=-L2B; alfa2=-PI/2;
theta3=0; d3=L3+L2A+q3; a3=0; alfa3=0;
theta4=q4+pi/2; d4=L4; a4=0; alfa4=PI/2;
theta5=-PI/2; d5=L5+q5; a5=0; alfa5=PI;
theta6=q6+PI/2; d6=0; a6=L6; alfa6=0;

%Matrices TH
TB0 = MDH(theta0,d0,a0,alfa0);
T01 = MDH(theta1,d1,a1,alfa1);
T12 = MDH(theta2,d2,a2,alfa2);
T23 = MDH(theta3,d3,a3,alfa3);
T34 = MDH(theta4,d4,a4,alfa4);
T45 = MDH(theta5,d5,a5,alfa5);
T56 = MDH(theta6,d6,a6,alfa6);

%Matriz global TB6
TB6 = simplify(TB0*T01*T12*T23*T34*T45*T56);

```

Las matrices obtenidas son las que se presentan:

$$TB0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T01 = \begin{bmatrix} \cos(q1) & 0 & \sin(q1) & 0 \\ \sin(q1) & 0 & -\cos(q1) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T12 = \begin{bmatrix} \cos\left(q2 - \frac{\pi}{2}\right) & 0 & -\sin\left(q2 - \frac{\pi}{2}\right) & -L2B \cdot \cos\left(q2 - \frac{\pi}{2}\right) \\ \sin\left(q2 - \frac{\pi}{2}\right) & 0 & \cos\left(q2 - \frac{\pi}{2}\right) & -L2B \cdot \sin\left(q2 - \frac{\pi}{2}\right) \\ 0 & -1 & 0 & -L1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

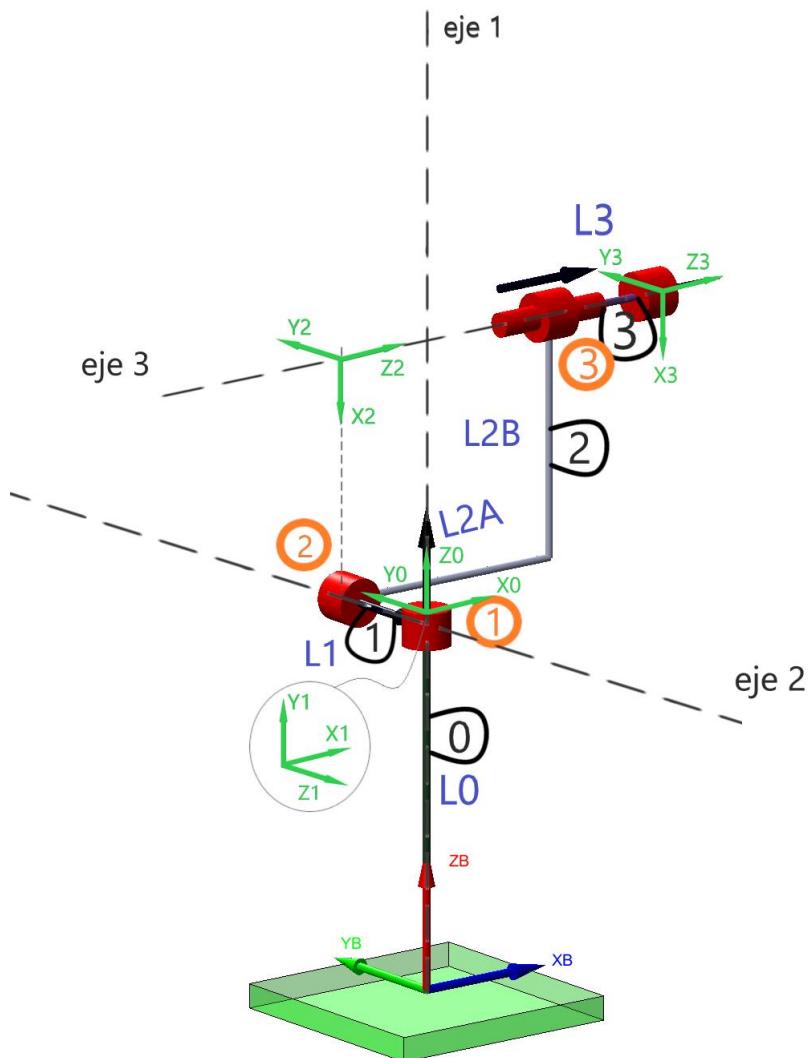
$$T23 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L3 + L2A + q3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T34 = \begin{bmatrix} \cos\left(q4 + \frac{\pi}{2}\right) & 0 & \sin\left(q4 + \frac{\pi}{2}\right) & 0 \\ \sin\left(q4 + \frac{\pi}{2}\right) & 0 & -\cos\left(q4 + \frac{\pi}{2}\right) & 0 \\ 0 & 1 & 0 & L4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{45} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & L_5 + q_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{56} = \begin{bmatrix} \cos\left(q_6 + \frac{\pi}{2}\right) & -\sin\left(q_6 + \frac{\pi}{2}\right) & 0 & L_6 \cdot \cos\left(q_6 + \frac{\pi}{2}\right) \\ \sin\left(q_6 + \frac{\pi}{2}\right) & \cos\left(q_6 + \frac{\pi}{2}\right) & 0 & L_6 \cdot \sin\left(q_6 + \frac{\pi}{2}\right) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Las matrices anteriores han servido para completar el fichero de verificación [matricesTH.m](#).

3. Cinemática Directa

En este apartado se calcularán las ecuaciones del modelo cinemático directo, trabajando con el robot reducido a sus tres primeros grados de libertad. Por tanto, el esquema que se usará en adelante será el siguiente:



Realizando de nuevo el algoritmo de D-H para conseguir la tabla de parámetros de las 3 primeras articulaciones e implementando el código correspondiente al igual que hicimos en el primer apartado de este análisis cinemático, obtenemos los siguientes resultados:

ARTICULACIÓN	θ_i (rad)	d_i	a_i	α_i (rad)
0	0	L_0	0	0
1	$q_1 + 0$	0	0	$\pi/2$
2	$q_2 - \pi/2$	$-L_1$	$-L_2B$	$-\pi/2$
3	0	$q_3 + L_2A + L_3$	0	0

Tabla D-H Robot 3 grados de libertad

- Posición:

```

syms L0 L1 L1A L1B L2 L2A L2B L3 L3A L3B L4 real
syms q1 q2 q3
PI = sym(pi);

%Parámetros de DH
theta0=0; d0=L0; a0=0; alfa0=0;
theta1=q1; d1=0; a1=0; alfa1=PI/2;
theta2=q2-PI/2; d2=-L1; a2=-L2B; alfa2=-PI/2;
theta3=0; d3=L3+L2A+q3; a3=0; alfa3=0;

%Matrices TH
TB0 = MDH(theta0,d0,a0,alfa0);
T01 = MDH(theta1,d1,a1,alfa1);
T12 = MDH(theta2,d2,a2,alfa2);
T23 = MDH(theta3,d3,a3,alfa3);

%Matriz global TB3
TB3 = simplify(TB0*T01*T12*T23);

%Posicion del efecto final (origen del marco referencia 3)
p=TB3(1:3,4);

```

Ecuaciones de posición de la cinemática directa obtenidas:

$$\begin{aligned}
x &= -L_1 \cdot \sin(q_1) - L_2B \cdot \cos(q_1) \cdot \sin(q_2) + \cos(q_1) \cdot \cos(q_2) \cdot (L_3 + L_2A + q_3) \\
y &= L_1 \cdot \cos(q_1) - L_2B \cdot \sin(q_1) \cdot \sin(q_2) + \cos(q_2) \cdot \sin(q_1) \cdot (L_3 + L_2A + q_3) \\
z &= L_0 + L_2B \cdot \cos(q_2) + \sin(q_2) \cdot (L_3 + L_2A + q_3)
\end{aligned}$$

Las cuales se han sacado de la cuarta columna, correspondiente a la posición, dentro de la matriz TB3:

$$TB3 = \begin{bmatrix} \cos(q_1) \cdot \sin(q_2) & -\sin(q_1) & \cos(q_1) \cdot \cos(q_2) & -L_1 \cdot \sin(q_1) - L_2B \cdot \cos(q_1) \cdot \sin(q_2) + \cos(q_1) \cdot \cos(q_2) \cdot (L_3 + L_2A + q_3) \\ \sin(q_1) \cdot \sin(q_2) & \cos(q_1) & \cos(q_2) \cdot \sin(q_1) & L_1 \cdot \cos(q_1) - L_2B \cdot \sin(q_1) \cdot \sin(q_2) + \cos(q_2) \cdot \sin(q_1) \cdot (L_3 + L_2A + q_3) \\ -\cos(q_2) & 0 & \sin(q_2) & L_0 + L_2B \cdot \cos(q_2) + \sin(q_2) \cdot (L_3 + L_2A + q_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Orientación:**

Para caracterizar la orientación del efecto final, obtendremos los ángulos de Euler (ϕ, θ, ψ) según el convenio ZXZ del marco de referencia 3 respecto a la base.

La obtención de ϕ, θ y ψ se realiza comparando las matrices de rotación de nuestro robot con la matriz teórica de rotación correspondiente al convenio ZXZ, ambas mostradas a continuación y obteniendo relaciones entre ellas. Finalmente se consigue tener los ángulos requeridos en función de posiciones de la matriz de rotación RB3.

Partiendo de la matriz TB3 podemos tener fácilmente RB3, matriz de rotación de nuestro robot de 3 gdl:

$$RB3 = \begin{bmatrix} \cos(q1) \cdot \sin(q2) & -\sin(q1) & \cos(q1) \cdot \cos(q2) \\ \sin(q1) \cdot \sin(q2) & \cos(q1) & \cos(q2) \cdot \sin(q1) \\ -\cos(q2) & 0 & \sin(q2) \end{bmatrix}$$

Matriz teórica de rotación correspondiente al convenio ZXZ:

$$R_{zxz} = \text{rotz}(\phi) \cdot \text{rotx}(\theta) \cdot \text{rotz}(\psi)$$

$$R_{zxz} = \begin{bmatrix} \cos(\phi) \cdot \cos(\psi) - \cos(\theta) \cdot \sin(\phi) \cdot \sin(\psi) & -\cos(\phi) \cdot \sin(\psi) - \cos(\psi) \cdot \cos(\theta) \cdot \sin(\phi) & \sin(\phi) \cdot \sin(\theta) \\ \cos(\psi) \cdot \sin(\phi) + \cos(\phi) \cdot \cos(\theta) \cdot \sin(\psi) & \cos(\phi) \cdot \cos(\psi) \cdot \cos(\theta) - \sin(\phi) \cdot \sin(\psi) & -\cos(\phi) \cdot \sin(\theta) \\ \sin(\psi) \cdot \sin(\theta) & \cos(\psi) \cdot \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Comparaciones y expresiones obtenidas:

$$\begin{aligned} \sin(\phi) \cdot \sin(\theta) &= RB3(1, 3) \\ -\cos(\phi) \cdot \sin(\theta) &= RB3(2, 3) \end{aligned} \rightarrow \phi = \text{atan2}(RB3(1, 3), -RB3(2, 3))$$

$$\begin{aligned} \cos(\theta) &= RB3(3, 3) \\ \sin(\theta) &= \pm \sqrt{1 - (RB3(3, 3))^2} \end{aligned} \rightarrow \theta = \text{atan2}\left(+\sqrt{1 - (RB3(3, 3))^2}, RB3(3, 3)\right)$$

$$\begin{aligned} \sin(\psi) \cdot \sin(\theta) &= RB3(3, 1) \\ \cos(\psi) \cdot \sin(\theta) &= RB3(3, 2) \end{aligned} \rightarrow \psi = \text{atan2}(RB3(3, 1), RB3(3, 2))$$

Las expresiones vistas tanto de posición como de orientación (ángulos de Euler) han sido usadas para completar los ficheros de verificación [CinematicaDirecta.m](#) y [CinematicaDirectaSimbolica.m](#).

- Ejemplo de comprobación para posición HOME: $q_1 = q_2 = q_3 = 0$

```
>> [xyz,angEuler] = CinematicaDirecta([0,0,0])
```

```
xyz =
```

```
1.5000
0.5000
2.5000
```

```
angEuler =
```

```
1.5708
1.5708
-1.5708
```

Puede comprobarse de forma visual en la propia figura del robot que la posición y orientación resultantes se corresponden con las esperadas. De igual forma, la posición y orientación finales del robot para unas coordenadas articulares dadas puede verse representada de forma gráfica con el siguiente código implementado:

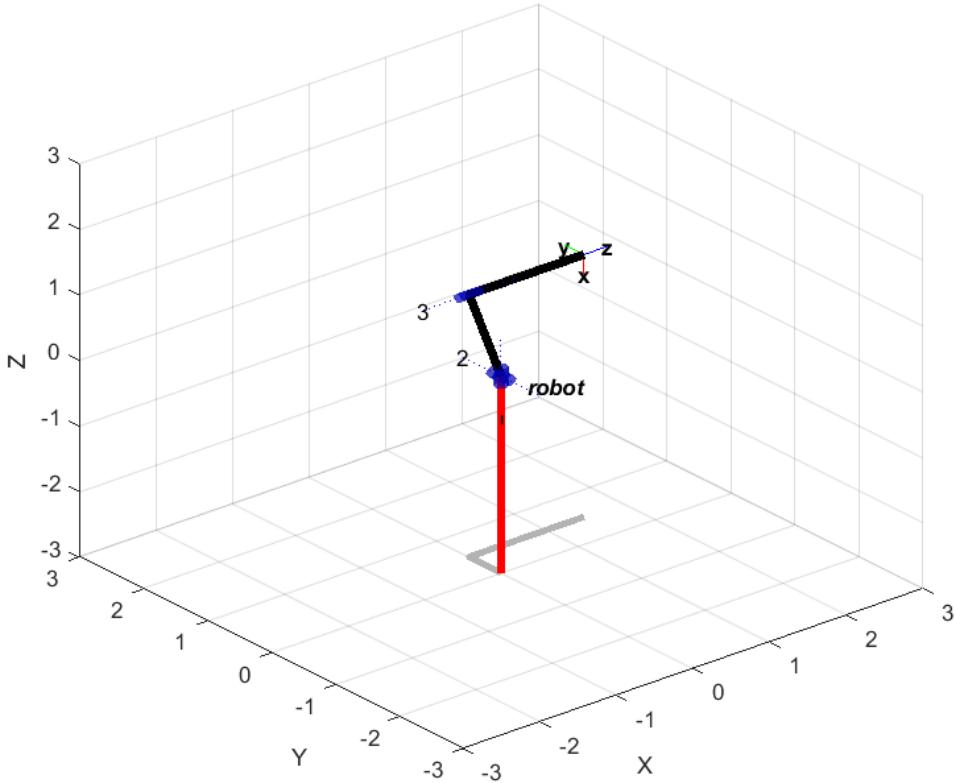
```
%% COMPROBACION ROBOT CINEMÁTICA DIRECTA 3GDL
%No se tiene en cuenta el paso de base a 0, hay que sumarle ese efecto (traslación en z)

%parametros fijos
L0=1.5;L1=0.5;L2A=1;L2B=1;L3=0.5;
%posiciones articulares
    %Posición HOME
    q1=0;q2=-pi/2;q3=L2A+L3;
    %Posición distinta (los términos sumando corresponden a las 'q' que pasamos a cinematicaDirecta)
    q1=q1+0;q2=q2+0;q3=q3+0;

%articulaciones
%L(1)=Link([0,L0,0,0,0]); %articulacion 0
L(1)=Link([q1,0,0,pi/2,0]); %articulacion 1
L(2)=Link([q2,-L1,-L2B,-pi/2,0]); %articulacion 2
L(3)=Link([0,q3,0,0,1]); %articulacion 3

%Generacion del robot
robot = SerialLink(L,'name','robot');
    %Matriz transformación homogénea
T=robot.fkine([q1 q2 q3]);
    %visualización de la posición del robot HOME
figure();
robot.plot([q1 q2 q3])
```

El resultado de ejecutar este código nos proporciona una figura tridimensional del robot donde podemos ver cuál es la configuración del mismo, la posición del extremo final y el sistema de ejes final. Para el caso de la posición de HOME:



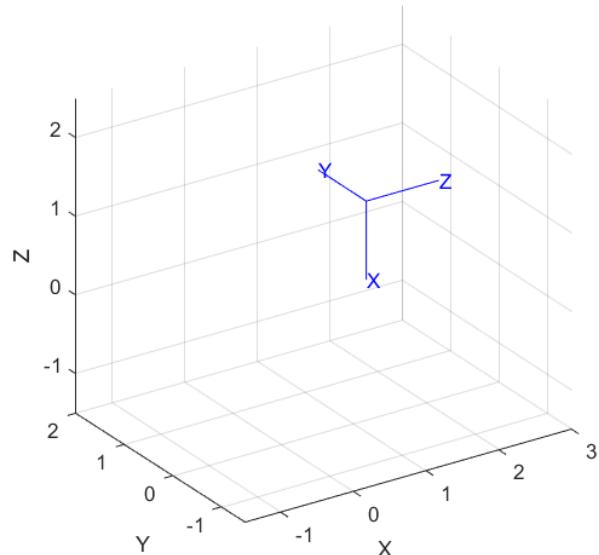
Para comprobar que los ángulos de Euler, vemos que haciendo las rotaciones del sistema de referencia base sobre los ejes móviles Z, X y Z los ángulos ϕ , θ y ψ respectivamente, llegamos al sistema de referencia final. Recordando que obtuvimos para esta posición particular que:

$$\phi = \frac{\pi}{2}; \theta = \frac{\pi}{2}; \psi = -\frac{\pi}{2}$$

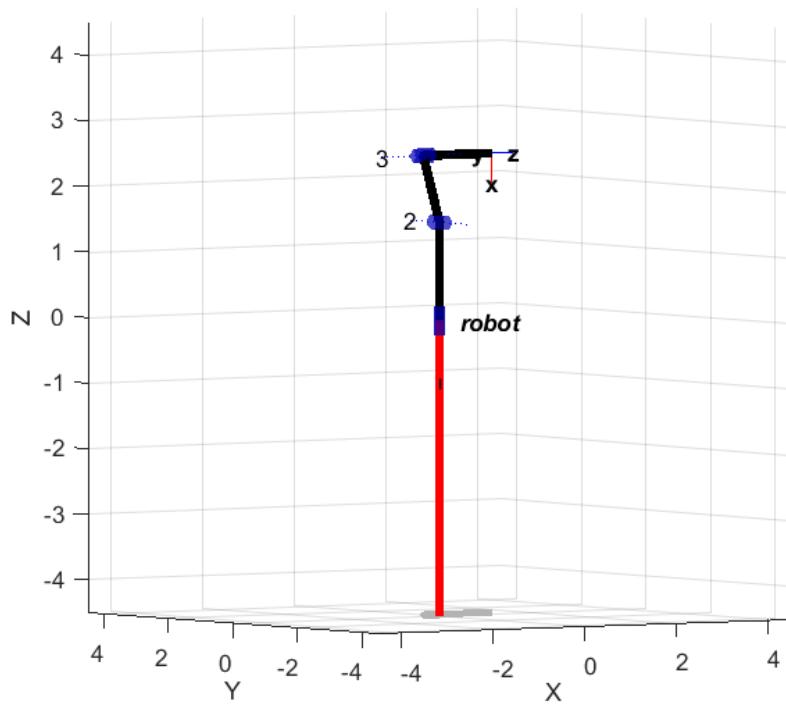
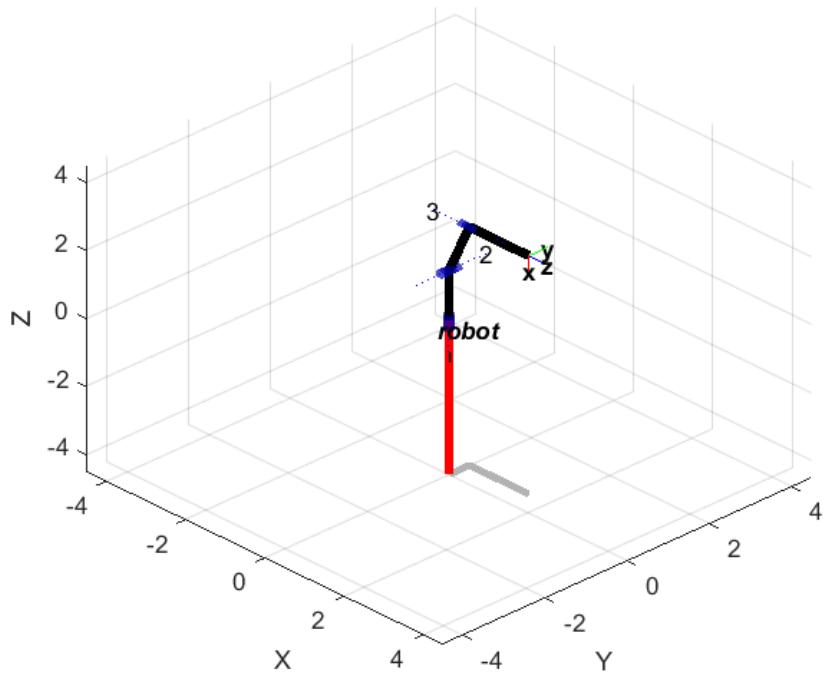
Vemos como se verifica lo comentado. Para esta comprobación es igualmente interesante hacer uso de la función `tranimate()`, pasándole la MTH obtenida anteriormente, T. Esta función nos muestra el movimiento que describe el efecto final, tanto en posición como en orientación, de ahí que resulte especialmente útil su uso.

```
%visualización del movimiento del efecto final
figure();
tranimate(T)
```

NOTA: Recordamos que en estas comprobaciones no se está teniendo en cuenta el desplazamiento en el eje z que introduce el paso del sistema base al 0, es decir, el efecto de TBO ya que la propia generación del robot que se ha planteado en el código no lo permite. Simplemente sabemos que debemos trasladar el resultado 10 unidades en el eje z fijo.



Puede plantearse una solución alternativa para subsanar esta pequeña limitación. Consiste en unir la llamada “articulación 0” que no se tiene en cuenta en el código anterior con la articulación 1. De esta forma la posición del efector final ya sí que sería la correcta. Esto logra un mayor grado de facilidad en la visualización de la posición del robot al no deber tener en cuenta mentalmente ese desplazamiento de L0 en el eje z.



4. Cinemática Inversa

Partiendo de las ecuaciones obtenidas de la cinemática directa:

- 1) $x = -L_1 \cdot \sin(q_1) - L_2B \cdot \cos(q_1) \cdot \sin(q_2) + \cos(q_1) \cdot \cos(q_2) \cdot (L_3 + L_2A + q_3)$
- 2) $y = L_1 \cdot \cos(q_1) - L_2B \cdot \sin(q_1) \cdot \sin(q_2) + \cos(q_2) \cdot \sin(q_1) \cdot (L_3 + L_2A + q_3)$
- 3) $z = L_0 + L_2B \cdot \cos(q_2) + \sin(q_2) \cdot (L_3 + L_2A + q_3)$

Reescribiendo 1) y 2):

- 1) $x = -L_1 \cdot \sin(q_1) + (\cos(q_2) \cdot (L_3 + L_2A + q_3) - L_2B \cdot \sin(q_2)) \cdot \cos(q_1)$
- 2) $y = L_1 \cdot \cos(q_1) + (\cos(q_2) \cdot (L_3 + L_2A + q_3) - L_2B \cdot \sin(q_2)) \cdot \sin(q_1)$

Se usará en adelante la notación siguiente por cuestiones de reducción del tamaño de las ecuaciones: $\sin(q_i) \leftrightarrow S_i$; $\cos(q_i) \leftrightarrow C_i$

Operando con ambas podemos obtener una nueva ecuación que relaciona solamente q_2 y q_3 deshaciéndonos así de q_1 :

$$\begin{aligned} 1)^2 &\rightarrow x^2 = L_1^2 \cdot S_1^2 - 2 \cdot (C_2 \cdot (L_3 + L_2A + q_3) - L_2B \cdot S_2) \cdot C_1 \cdot L_1 \cdot S_1 + (C_2 \cdot (L_3 + L_2A + q_3) - L_2B \cdot S_2)^2 \cdot C_1^2 \\ 2)^2 &\rightarrow y^2 = L_1^2 \cdot C_1^2 + 2 \cdot (C_2 \cdot (L_3 + L_2A + q_3) - L_2B \cdot S_2) \cdot C_1 \cdot L_1 \cdot S_1 + (C_2 \cdot (L_3 + L_2A + q_3) - L_2B \cdot S_2)^2 \cdot S_1^2 \\ 1)^2 + 2)^2 &\rightarrow x^2 + y^2 = L_1^2 + (C_2 \cdot (L_3 + L_2A + q_3) - L_2B \cdot S_2)^2 \end{aligned}$$

Reescribiendo ahora la ecuación 3) y elevándola al cuadrado podemos conseguir la siguiente expresión:

$$3)^2 \rightarrow (z - L_0)^2 = (S_2 \cdot (L_3 + L_2A + q_3) + L_2B \cdot C_2)^2$$

Sumando ambas expresiones que solo dependen de q_2 y q_3 :

$$\begin{aligned} 1)^2 + 2)^2 &\rightarrow x^2 + y^2 = L_1^2 + (C_2 \cdot (L_3 + L_2A + q_3) - L_2B \cdot S_2)^2 \\ 3)^2 &\rightarrow (z - L_0)^2 = (S_2 \cdot (L_3 + L_2A + q_3) + L_2B \cdot C_2)^2 \\ 1)^2 + 2)^2 + 3)^2 &\rightarrow x^2 + y^2 + (z - L_0)^2 = (L_3 + L_2A + q_3)^2 + L_2B^2 + L_1^2 \end{aligned}$$

Tenemos por tanto una expresión de la que únicamente es incógnita q_3 , despejándola:

$$q_3 = \pm \sqrt{x^2 + y^2 + (z - L_0)^2 - L_1^2 - L_2B^2 - L_3 - L_2A}$$

Por tanto, tenemos 2 posibles soluciones en función del signo que acompaña a la raíz, lo cual se tendrá en cuenta en nuestro código implementado.

Una vez conocido el valor de q3 podemos despejar q2 de la ecuación 3) como sigue:

$$3) z = L0 + L2B \cdot C_2(q2) + (L3 + L2A + q3) \cdot S_2$$

$$A \cdot S_2 + B \cdot C_2 = C$$

Donde :

$$A = L3 + L2A + q3$$

$$B = L2B$$

$$C = z - L0$$

$$\begin{aligned} A &= R \cdot \cos(\varphi) & R &= \sqrt{A^2 + B^2} \\ B &= R \cdot \sin(\varphi) & \rightarrow & \varphi = \text{atan2}(B, A) \end{aligned}$$

$$R \cdot C_\varphi \cdot S_2 + R \cdot S_\varphi \cdot C_2 = C$$

Aplicando fórmula conocida : $\sin(\alpha + \beta) = \sin(\alpha) \cdot \cos(\beta) + \cos(\alpha) \cdot \sin(\beta)$

$$R \cdot S(q2 + \varphi) = C$$

$$S(q2 + \varphi) = \frac{C}{R} ; \quad C(q2 + \varphi) = \pm \sqrt{1 - \left(\frac{C}{R}\right)^2}$$

$$q2 = \text{atan2}\left(\frac{C}{R}, \pm \sqrt{1 - \left(\frac{C}{R}\right)^2}\right) - \varphi$$

Nuevamente, tenemos 2 posibles soluciones en función del signo que acompaña a la raíz, también se tendrá en cuenta en el código implementado.

Por último, es turno de conseguir la expresión de q1 conocidas ya q2 y q3. Podríamos optar por despejarla de las ecuaciones 1) o 2).

Sin embargo, probaremos a obtener alguna otra expresión alternativa que sea más sencilla de despejar. Esto puede intentarse a partir de la operación con las matrices de transformación homogéneas que ya tenemos. Para ello hemos desarrollado el código que se muestra, del cual se obtiene una nueva ecuación bastante más sencilla respecto a las previas. De la “alternativa 2” podemos conseguir dicha nueva expresión.

```
syms x y z real
syms o1 o2 o3 n1 n2 n3 a1 a2 a3 real
T=[n1 o1 a1 x;...
   n2 o2 a2 y;...
   n3 o3 a3 z;...
   0 0 0 1];

%ALTERNATIVA 1
simplify(T01*T12*T23);
simplify(inv(TB0)*T);

%ALTERNATIVA 2
simplify(T12*T23)
simplify(inv(T01)*inv(TB0)*T)

%ALTERNATIVA 3
simplify(T23);
simplify(inv(T12)*inv(T01)*inv(TB0)*T);
```

```

[sin(q2), 0, cos(q2), L3*cos(q2) + L2A*cos(q2) - L2B*sin(q2) + q3*cos(q2)]
[-cos(q2), 0, sin(q2), L2B*cos(q2) + L3*sin(q2) + L2A*sin(q2) + q3*sin(q2)]
[0, -1, 0,
[0, 0, 0,
[n1*cos(q1) + n2*sin(q1), o1*cos(q1) + o2*sin(q1), a1*cos(q1) + a2*sin(q1), x*cos(q1) + y*sin(q1)]
[n3, o3, a3, z - L0]
[n1*sin(q1) - n2*cos(q1), o1*sin(q1) - o2*cos(q1), a1*sin(q1) - a2*cos(q1), x*sin(q1) - y*cos(q1)]
[0, 0, 0, 1]

```

$$-L1 = x \cdot S_1 - y \cdot C_1$$

Vemos que q_1 realmente no depende de q_1 ni q_2 . Llevando a cabo el mismo desarrollo que para despejar q_2 conseguimos al fin q_1 :

$$A \cdot S_1 + B \cdot C_1 = C$$

Donde :

$$A = x$$

$$B = -y$$

$$C = L1$$

$$A = R \cdot \cos(\varphi) \rightarrow R = \sqrt{A^2 + B^2}$$

$$B = R \cdot \sin(\varphi) \quad \varphi = \text{atan2}(B, A)$$

$$R \cdot C_\varphi \cdot S_1 + R \cdot S_\varphi \cdot C_1 = C$$

Aplicando fórmula conocida : $\sin(\alpha + \beta) = \sin(\alpha) \cdot \cos(\beta) + \cos(\alpha) \cdot \sin(\beta)$

$$R \cdot S(q1 + \varphi) = C$$

$$S(q1 + \varphi) = \frac{C}{R} ; \quad C(q1 + \varphi) = \pm \sqrt{1 - \left(\frac{C}{R}\right)^2}$$

$$q1 = \text{atan2}\left(\frac{C}{R}, \pm \sqrt{1 - \left(\frac{C}{R}\right)^2}\right) - \varphi$$

Una vez obtenidas las expresiones para hallar las coordenadas articulares a partir de los parámetros fijos característicos del robot y de la posición (x, y, z) en la que se encuentra el extremo de éste desarrollamos una función de Matlab que nos devuelve las posibles soluciones para la cinemática inversa. De las 8 posibles combinaciones de soluciones que existen, únicamente la mitad son soluciones válidas (esto puede saberse fácilmente comprobando, por ejemplo, para la solución de HOME, de forma numérica, si al aplicar la Cinemática Directa ya desarrollada a las posibles soluciones numéricas que resultan de este apartado, todas ellas conducen a la misma solución, que en este caso sería la obtenida al aplicar la cinemática directa con $q = [0, 0, 0]$).

Descartadas las soluciones correspondientes como se ha detallado, se han completado los ficheros de verificación numérica [cinematicalInversaSimbolica.m](#) y [cinematicalInversa.m](#).

En dichas funciones puede verse el código implementado en detalle.

Además, en este código se tiene en cuenta las condiciones para las cuales la cinemática inversa no es posible calcularse. Éstas se resumen a continuación:

$x^2 + y^2 + (z - L_0)^2 - L_1^2 - L_2^2 \geq 0 \rightarrow$ Si no se cumple esta desigualdad q_3 tendría un valor imaginario en lugar de real ya que resultaría en una raíz negativa lo cual provocaría errores en el cálculo de q_2 con el uso de la función atan2().

$\left| \frac{C}{R} \right| \leq 1 \rightarrow$ Condición que debe cumplirse en el cálculo de q_2 y q_1 ya que el seno de un ángulo puede tomar valores entre -1 y 1 únicamente.

Por este motivo, si se da alguna posición en coordenadas cartesianas que no cumpla las condiciones anteriores, no existirá solución para la cinemática inversa y nuestra función devolverá como solución $q = [0 \ 0 \ 0]'$ junto con la variable lógica `sol_OK` con valor 0.

```
>> [q,sol_OK]=CinemáticaInversa([0.2,0,2])
```

```
q =
```

```
0  
0  
0
```

```
sol_OK =
```

```
logical
```

```
0
```

También puede darse el caso en el que para ciertas posiciones existan solamente 2 soluciones en vez de 4, en ese caso la función proporciona únicamente dichas dos soluciones. En el caso más general, obtendremos 4 soluciones y `sol_OK=1`.

```
>> [q,sol_OK]=CinemáticaInversa([4,2,3.5])
```

```
q =
```

```
0.3516 -2.5659 0.3516 -2.5659  
0.2162 2.5120 -2.5120 -0.2162  
3.2697 3.2697 -6.2697 -6.2697
```

```
sol_OK =
```

```
logical
```

```
1
```

Comprobación para la posición de HOME:

$$q_1 = 0; \quad q_2 = 0; \quad q_3 = 0$$

```
>> xyz=CinematicaDirecta([0,0,0])  
  
xyz =  
  
1.5000  
0.5000  
2.5000  
  
>> q=CinematicaInversa(xyz)  
  
q =  
  
-0.0000 -2.4981 -0.0000 -2.4981  
0 1.9656 -1.9656 0  
0 0 -3.0000 -3.0000  
  
>> CinematicaDirecta(q(:,1))  
  
ans =  
  
1.5000  
0.5000  
2.5000  
  
>> CinematicaDirecta(q(:,2))  
  
ans =  
  
1.5000  
0.5000  
2.5000  
  
>> CinematicaDirecta(q(:,3))  
  
ans =  
  
1.5000  
0.5000  
2.5000  
  
>> CinematicaDirecta(q(:,4))  
  
ans =  
  
1.5000  
0.5000  
2.5000
```

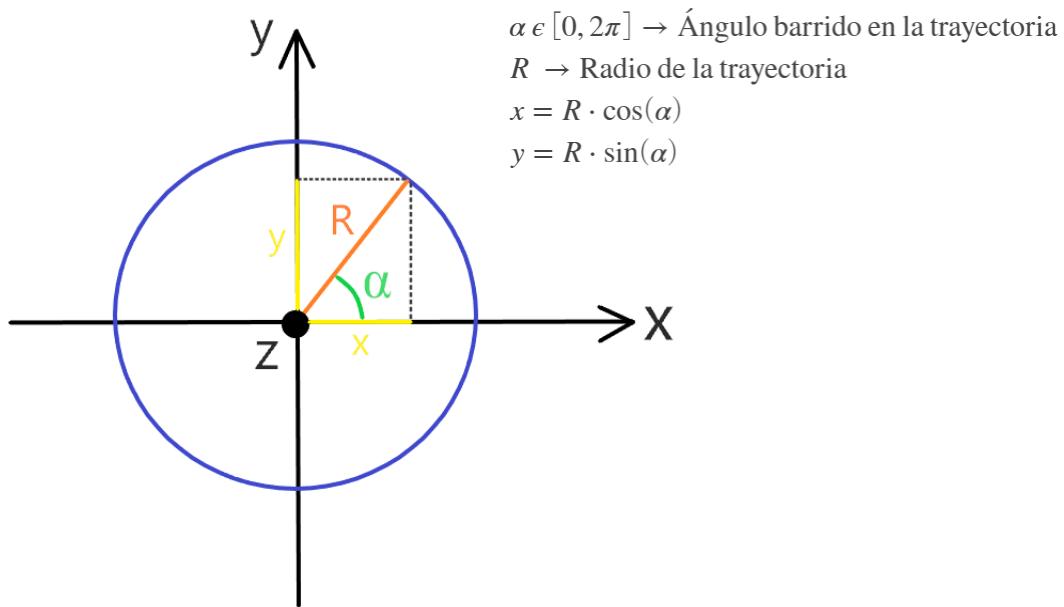
Comprobación para otra posición:

$$q_1 = \frac{\pi}{2}; \quad q_2 = \pi; \quad q_3 = 4$$

```
>> xyz = CinematicaDirecta([pi/2,pi,4])  
  
xyz =  
  
-0.5000  
-5.5000  
0.5000  
  
>> q = CinematicaInversa(xyz)  
  
q =  
  
-1.7521 -4.7124 -1.7521 -4.7124  
-0.3597 -3.1416 -3.1416 -5.9235  
4.0000 4.0000 -7.0000 -7.0000  
  
>> CinematicaDirecta(q(:,1))  
  
ans =  
  
-0.5000  
-5.5000  
0.5000  
  
>> CinematicaDirecta(q(:,2))  
  
ans =  
  
-0.5000  
-5.5000  
0.5000  
  
>> CinematicaDirecta(q(:,3))  
  
ans =  
  
-0.5000  
-5.5000  
0.5000  
  
>> CinematicaDirecta(q(:,4))  
  
ans =  
  
-0.5000  
-5.5000  
0.5000
```

5. Trayectoria Circular

Planteamos la generación de la trayectoria circular en el plano cartesiano X-Y de la siguiente forma:



Haciendo uso de la Cinemática Inversa que ya tenemos disponible del apartado anterior desarrollamos el siguiente código, el cual, se ha implementado en el fichero de verificación [trayectoriaCircular.m](#).

```

N=72; %Número de puntos de la trayectoria

x=zeros(1,N);
y=zeros(1,N);
z=zeros(1,N);
q1=zeros(1,N);
q2=zeros(1,N);
q3=zeros(1,N);

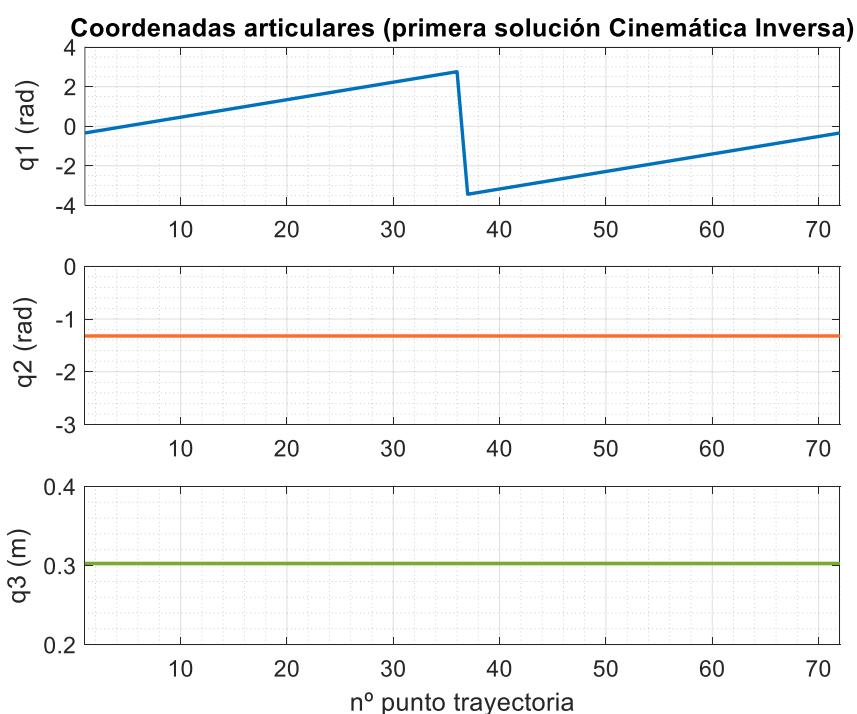
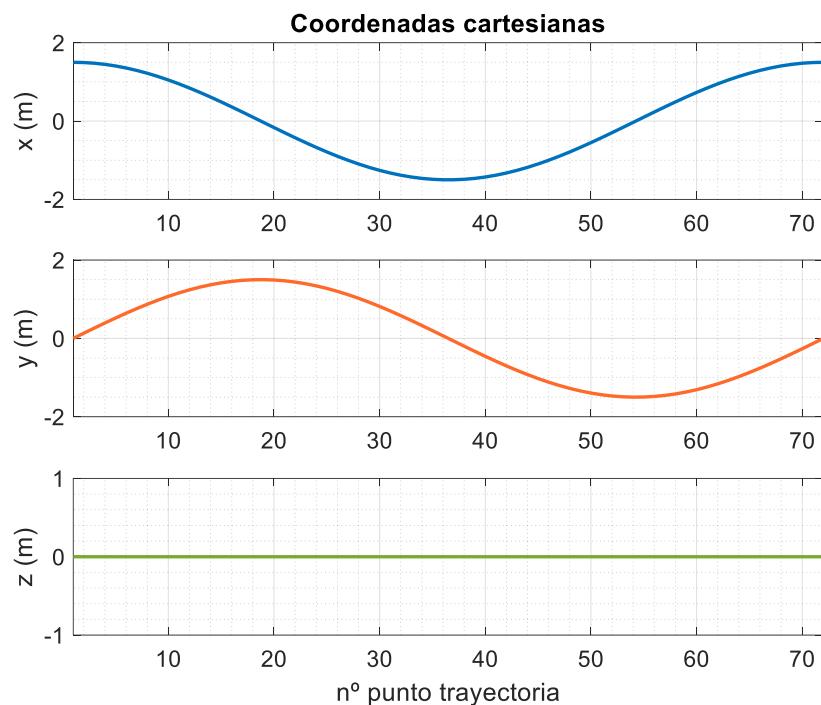
H=0; %Altura a la que se da la trayectoria en el eje z
R=1.5; %Radio de la trayectoria
alpha=0:2*pi/(N-1):2*pi; %Ángulo barrido en la trayectoria circular

for i=1:N
    x(i)=R*cos(alpha(i));
    y(i)=R*sin(alpha(i));
    z(i)=H;
    q=CinematicaInversa([x(i),y(i),z(i)]);
    q1(i)=q(1,1);
    q2(i)=q(2,1);
    q3(i)=q(3,1);
end

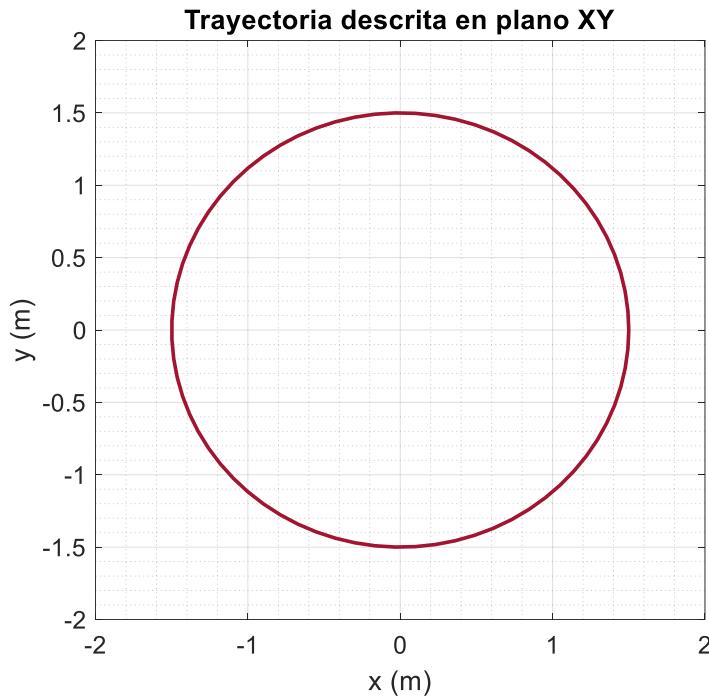
```

Nota: La solución elegida de las 4 posibles para las coordenadas articulares es la primera, ya que tras representar la evolución de las 4 posibles soluciones nos percatamos de que esta primera es la que permite realizar la trayectoria requerida con un menor movimiento de las articulaciones, por tanto, pensamos que es la más eficiente de todas. Las cuatro soluciones serían igualmente válidas, pero por el motivo expuesto, puestos a elegir, nos quedamos con la primera.

La evolución de las distintas coordenadas cartesianas y articulares correspondientes a los puntos de la trayectoria se representan en las siguientes gráficas:



Si representamos las coordenadas cartesianas x e y simultáneamente podremos verificar que efectivamente se describe una trayectoria circular, en nuestro caso de radio 1.5 m.

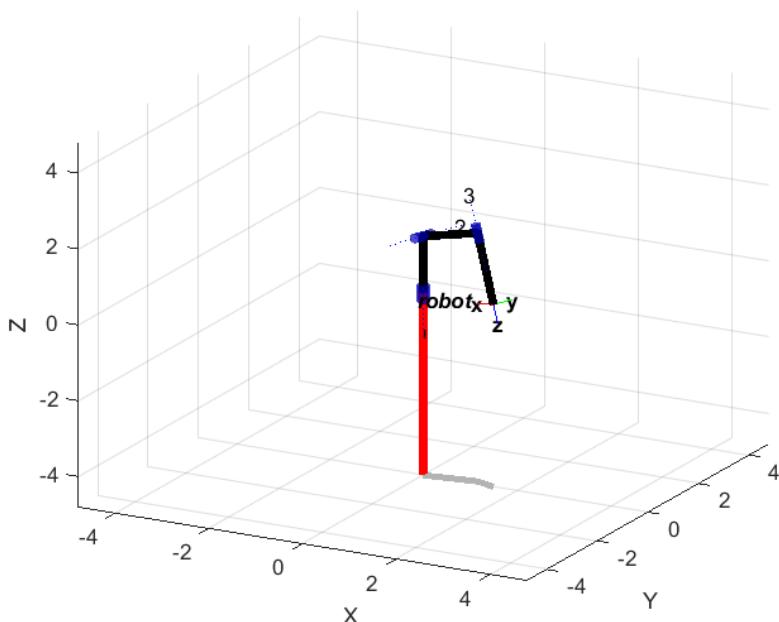
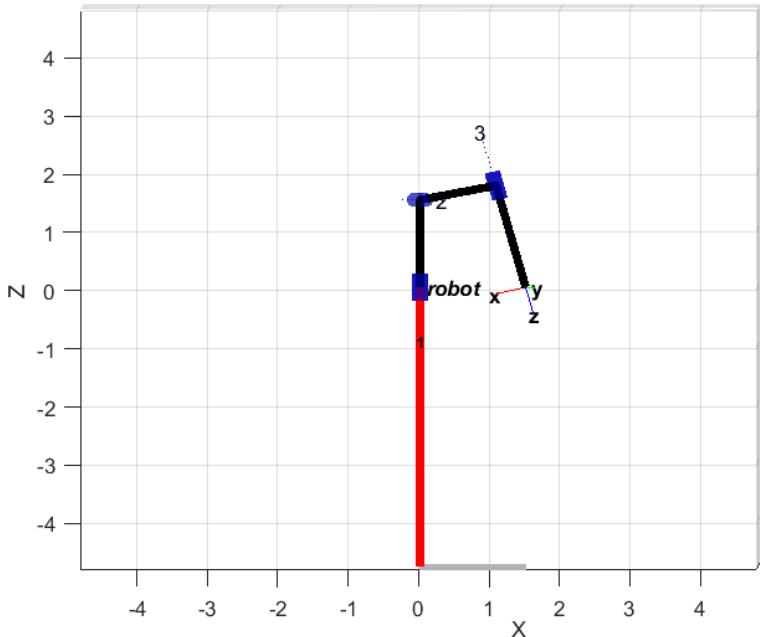


Además, con el siguiente código somos capaces de visualizar esquemáticamente la posición de nuestro robot al inicio de la trayectoria que describe. De igual forma se podría ver la posición del robot en cualquier punto de la misma, con un simple cambio en las coordenadas articulares que se emplean.

```
% POSICIÓN ROBOT INICIO TRAYECTORIA
%parametros fijos
L0=1.5;L1=0.5;L2A=1;L2B=1;L3=0.5;
%posiciones articulares
    %Posición inicial HOME
    q1=0;q2=-pi/2;q3=L2A+L3;
    %Posición distinta (inical trayectoria)
    q1=q1-0.3398;q2=q2-1.3213;q3=q3+0.3028;

%articulaciones
%L(1)=Link([0,L0,0,0,0]); %articulacion 0
%L(1)=Link([q1,0,0,pi/2,0]); %articulacion 1
L(1)=Link([q1,L0,0,pi/2,0]); %articulacion 1
L(2)=Link([q2,-L1,-L2B,-pi/2,0]); %articulacion 2
L(3)=Link([0,q3,0,0,1]); %articulacion 3

%Generación del robot
robot = SerialLink(L,'name','robot');
    %Matriz transformación homogénea
    T=robot.fkine([q1 q2 q3]);
    %visualización de la posición del robot inicio trayectoria
    figure();
    robot.plot([q1 q2 q3])
```



6. Jacobianos directos e inversos

- Jacobiiano Directo

Los puntos singulares del robot se pueden obtener fácilmente a partir del cálculo del jacobiano directo del modelo cinemático directo. Los puntos singulares se obtienen cuando esta matriz (jacobiano) pierde rango, es decir, cuando pasa a ser una matriz no invertible, en otras palabras, cuando el determinante del jacobiano es nulo.

$$J_{\text{dir}} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial z}{\partial q_3} \end{bmatrix} \rightarrow \det(J_{\text{dir}}) = (L_3 + L_2A + q_3) \cdot (C_2 \cdot (L_3 + L_2A + q_3) - S_2 \cdot L_2B)$$

$$\det(J_{\text{dir}}) = 0$$

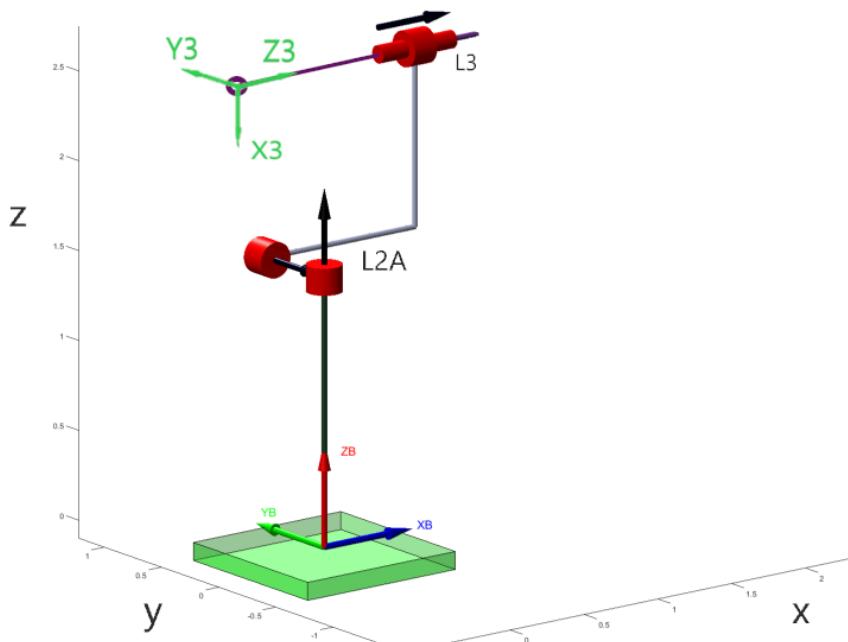
$$(L_3 + L_2A + q_3) = 0 \rightarrow q_3 = -L_3 - L_2A$$

$$(C_2 \cdot (L_3 + L_2A + q_3) - S_2 \cdot L_2B) = 0 \rightarrow \tan(q_2) = \frac{L_3 + L_2A + q_3}{L_2B}$$

En un punto singular no tenemos libertad para mover el robot en la dirección que queramos de manera instantánea. Es importante por tanto ser conscientes de la existencia de ellos en nuestro robot con la finalidad de evitar los posibles giros bruscos, velocidades elevadas e incluso colapso del robot que éstos conllevan.

Podemos analizar el primer punto singular: $q_3 = -L_3 - L_2A$

Cuando la variable articular q_3 toma dicho valor puede verse que el movimiento del extremo del robot en la dirección y (referida a ejes móviles finales) queda inhabilitada. Esto puede observarse en el esquema siguiente, donde se representa la posición que tendría el robot para las coordenadas articulares: $q_3 = -L_3 - L_2A$ y $q_2 = q_1 = 0$



Por otro lado, analizar el segundo punto singular obtenido no es una tarea tan sencilla. Sin embargo, podemos ver fácilmente como para el caso particular en que $q_3 = -L_3 - L_2A$, según la expresión obtenida para este segundo punto singular, $\tan(q_2) = \frac{L_3 + L_2A + q_3}{L_2B}$.

Obtenemos que este se da para $q_2 = 0$, que como puede verse en el esquema anterior, si en esa posición si q_2 es nulo, tenemos efectivamente un punto singular.

- Jacobiano Inverso

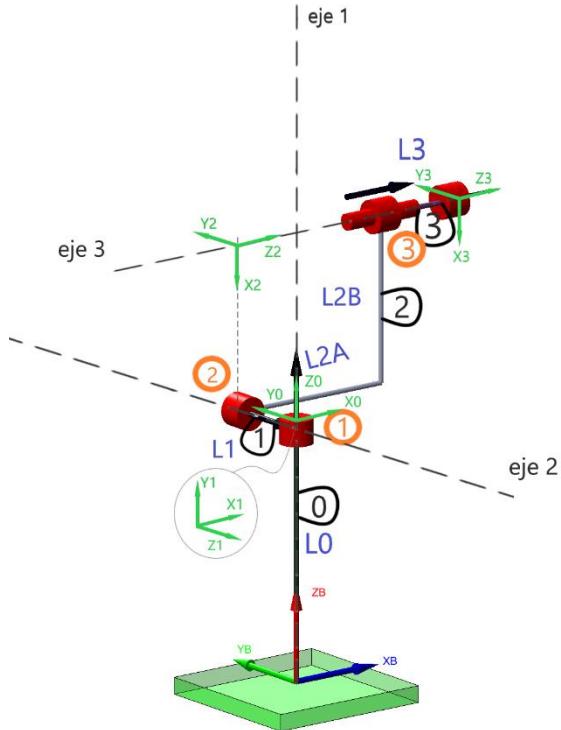
$$J_{\text{inv}} = \begin{bmatrix} \frac{\partial q_1}{\partial x} & \frac{\partial q_1}{\partial y} & \frac{\partial q_1}{\partial z} \\ \frac{\partial q_2}{\partial x} & \frac{\partial q_2}{\partial y} & \frac{\partial q_2}{\partial z} \\ \frac{\partial q_3}{\partial x} & \frac{\partial q_3}{\partial y} & \frac{\partial q_3}{\partial z} \end{bmatrix}$$

También puede hallarse como la matriz inversa del jacobiano directo $\rightarrow J_{\text{inv}} = \text{inv}(J_{\text{dir}})$

Este apartado se ha implementado en forma de código en el fichero [jacobiano.m](#), que genera las matrices jacobianas directa e inversa en modo simbólico.

2) ANÁLISIS DINÁMICO

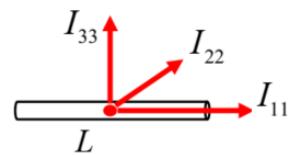
Continuamos con el estudio de la dinámica de nuestro robot. Se considerarán únicamente los 3 primeros grados de libertad. Recordamos el esquema que obtuvimos en la primera parte, en el cual nos apoyaremos de nuevo en esta segunda parte.



1. Parámetros y Modelo Dinámico

1.1. Parámetros Dinámicos

Calcularemos las posiciones de los centros de gravedad de cada enlace y los tensores de inercia asociados referidos al marco de referencia local de cada eslabón. Recordamos que, para nuestros eslabones, los tensores de inercia pueden hallarse según el siguiente resumen:



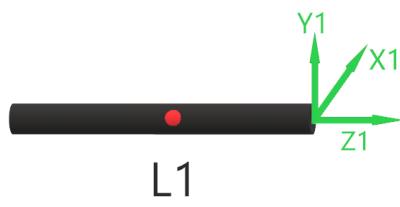
$$I = \begin{pmatrix} I_{11} & -P_{12} & -P_{13} \\ -P_{12} & I_{22} & -P_{23} \\ -P_{13} & -P_{23} & I_{33} \end{pmatrix} \quad I_{22} = I_{33} = \frac{1}{12}ML^2$$

$$I_{11} = \frac{1}{2}M(R_{ext}^2 + R_{int}^2)$$

$$P_{12} = P_{13} = P_{23} = 0$$

Como veremos a continuación, para la barra 1 y 3 los cálculos son inmediatos, sin embargo, en la barra 2, debido a su forma compuesta por dos tramos, el cálculo de la posición del centro de masas no es tan simple y para hallar los tensores de inercia debemos aplicar el Teorema de Steiner.

■ Eslabón 1:



$$\text{Posición del cdg del eslabón 1 : } S_{11} = \left(0, 0, -\frac{L_1}{2} \right)$$

Tensor de inercia del eslabón 1

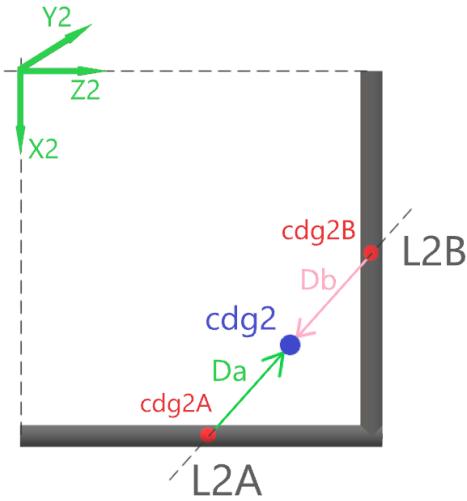
$$I_{xx1} = \frac{m_1 \cdot L_1^2}{12}$$

$$I_{yy1} = \frac{m_1 \cdot L_1^2}{12}$$

$$I_{zz1} = \frac{m_1 \cdot (R_{ext}^2 + R_{int}^2)}{2}$$

$$\rightarrow I_{11} = \begin{bmatrix} I_{xx1} & 0 & 0 \\ 0 & I_{yy1} & 0 \\ 0 & 0 & I_{zz1} \end{bmatrix}$$

■ Eslabón 2:



$$\text{Posición del cdg del eslabón 2A : } P_{cdg\ 2A} = \left(L_{2B}, 0, \frac{L_{2A}}{2} \right)$$

$$\text{Posición del cdg del eslabón 2B : } P_{cdg\ 2B} = \left(\frac{L_{2B}}{2}, 0, L_{2A} \right)$$

$$\text{Posición del cdg del eslabón 2 : } S_{22} = \frac{P_{cdg\ 2A} \cdot M_{2A} + P_{cdg\ 2B} \cdot M_{2B}}{M_{2A} + M_{2B}} \rightarrow$$

$$\rightarrow (M_{2A} = M_{2B}) \rightarrow S_{22} = \frac{P_{cdg\ 2A} + P_{cdg\ 2B}}{2} = \left(\frac{3}{4} \cdot L_{2B}, 0, \frac{3}{4} \cdot L_{2A} \right)$$

Tensor de inercia del eslabón 2A en cdg2A

$$I_{xx2A} = \frac{m_{2A} \cdot L_{2A}^2}{12} \quad \rightarrow \quad I_{2A} = \begin{bmatrix} I_{xx2A} & 0 & 0 \\ 0 & I_{yy2A} & 0 \\ 0 & 0 & I_{zz2A} \end{bmatrix}$$

$$I_{yy2A} = \frac{m_{2A} \cdot L_{2A}^2}{12}$$

$$I_{zz2A} = \frac{m_{2A} \cdot (R_{ext}^2 + R_{int}^2)}{2}$$

$$Da = \begin{bmatrix} x_{Da} \\ y_{Da} \\ z_{Da} \end{bmatrix} = \begin{bmatrix} x_{S22} - x_{P_{cdg\ 2A}} \\ y_{S22} - y_{P_{cdg\ 2A}} \\ z_{S22} - z_{P_{cdg\ 2A}} \end{bmatrix} \leftarrow \text{Distancia de cdg2A a cdg2}$$

$$MDa = \begin{bmatrix} y_{Da}^2 + z_{Da}^2 & -x_{Da} \cdot y_{Da} & -x_{Da} \cdot z_{Da} \\ -x_{Da} \cdot y_{Da} & x_{Da}^2 + z_{Da}^2 & -y_{Da} \cdot z_{Da} \\ -x_{Da} \cdot z_{Da} & -y_{Da} \cdot z_{Da} & x_{Da}^2 + y_{Da}^2 \end{bmatrix}$$

Teorema de Steiner :

$$I_{2A_{cdg}} = I_{2A} + m_{2A} \cdot MDa \rightarrow \text{Tensor de inercia del eslabón 2A en cdg2}$$

Tensor de inercia del eslabón 2B en cdg2B

$$I_{xx2B} = \frac{m_{2B} \cdot (R_{ext}^2 + R_{int}^2)}{2} \quad \rightarrow \quad I_{2B} = \begin{bmatrix} I_{xx2B} & 0 & 0 \\ 0 & I_{yy2B} & 0 \\ 0 & 0 & I_{zz2B} \end{bmatrix}$$

$$I_{yy2B} = \frac{m_{2B} \cdot L_{2B}^2}{12}$$

$$I_{zz2B} = \frac{m_{2B} \cdot L_{2B}^2}{12}$$

$$Db = \begin{bmatrix} x_{Db} \\ y_{Db} \\ z_{Db} \end{bmatrix} = \begin{bmatrix} x_{S22} - x_{P_{cdg\ 2B}} \\ y_{S22} - y_{P_{cdg\ 2B}} \\ z_{S22} - z_{P_{cdg\ 2B}} \end{bmatrix} \leftarrow \text{Distancia de cdg2B a cdg2}$$

$$MDb = \begin{bmatrix} y_{Db}^2 + z_{Db}^2 & -x_{Db} \cdot y_{Db} & -x_{Db} \cdot z_{Db} \\ -x_{Db} \cdot y_{Db} & x_{Db}^2 + z_{Db}^2 & -y_{Db} \cdot z_{Db} \\ -x_{Db} \cdot z_{Db} & -y_{Db} \cdot z_{Db} & x_{Db}^2 + y_{Db}^2 \end{bmatrix}$$

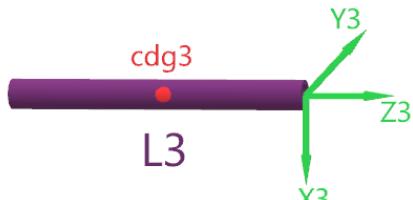
Teorema de Steiner :

$$I_{2B_{cdg}} = I_{2B} + m_{2B} \cdot MDb \rightarrow \text{Tensor de inercia del eslabón 2B en cdg2}$$

Tensor de inercia del eslabón 2

$$I_{22} = I_{2A_{cdg}} + I_{2B_{cdg}}$$

▪ Eslabón 3:



Posición del cdg del eslabón 3 : $S_{33} = \left(0, 0, -\frac{L_3}{2}\right)$

Tensor de inercia del eslabón 3

$$I_{xx3} = \frac{m_3 \cdot L_3^2}{12}$$

$$I_{yy3} = \frac{m_3 \cdot L_3^2}{12} \quad \rightarrow \quad I_{33} = \begin{bmatrix} I_{xx3} & 0 & 0 \\ 0 & I_{yy3} & 0 \\ 0 & 0 & I_{zz3} \end{bmatrix}$$

$$I_{zz3} = \frac{m_3 \cdot (R_{ext}^2 + R_{int}^2)}{2}$$

Los resultados vistos se han implementado en forma de código en el fichero [parametrosDinamicos.m](#) el cual devuelve las posiciones de los centros de gravedad de las tres barras y los tensores de inercia de forma numérica. Los resultados obtenidos finalmente son los siguientes:

$$S_{11} = \begin{bmatrix} 0 \\ 0 \\ -0.25 \end{bmatrix}; S_{22} = \begin{bmatrix} 0.75 \\ 0 \\ 0.75 \end{bmatrix}; S_{33} = \begin{bmatrix} 0 \\ 0 \\ -0.25 \end{bmatrix}$$

$$I_{11} = \begin{bmatrix} 0.6833 & 0 & 0 \\ 0 & 0.6833 & 0 \\ 0 & 0 & 0.2689 \end{bmatrix} \quad I_{22} = \begin{bmatrix} 14.2038 & 0 & 8.1996 \\ 0 & 27.3319 & 0 \\ 8.1996 & 0 & 14.2038 \end{bmatrix}$$

$$I_{33} = \begin{bmatrix} 0.6833 & 0 & 0 \\ 0 & 0.6833 & 0 \\ 0 & 0 & 0.2689 \end{bmatrix}$$

1.2. Modelo Dinámico

Para el cálculo de las ecuaciones dinámicas se ha empleado el método de Newton-Euler estudiado durante el curso. El fichero utilizado ha sido [NE_3GDL.m](#) que se adjunta también de forma complementaria a esta memoria. A partir de la ejecución de este fichero podemos obtener las matrices Ma, Va y Ga que son usadas para completar el fichero de verificación [ModeloDinamico_R3GDL.m](#), el cual a su vez nos proporciona las aceleraciones articulares en función de las posiciones y velocidades articulares, y los pares articulares generalizados.

Empecemos recordando la estructura del modelo dinámico inverso de un robot:

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + F(\dot{q})$$

$q = (q_1 \dots q_N)^T$: vector de coordenadas articulares

$M(q)$: matriz de inercias

$V(q, \dot{q}) = C(q, \dot{q})\dot{q}$: vector de términos centrípetos y de Coriolis

$G(q)$: vector de términos gravitatorios

$F(\dot{q})$: vector de términos fricciones (no conservativas)

τ : fuerzas/pares generalizadas aplicados en las articulaciones

Al incluir el efecto de los actuadores:

$$K_t R I_i = \underbrace{(M(q) + R^2 J_m)}_{\tau} \ddot{q} + \underbrace{(V(q, \dot{q}) + R^2 B_m \dot{q})}_{M_A(q)} + \underbrace{G(q)}_{V_A(q, \dot{q}) + G_A(q)}$$

R : Vector factores de reducción reductoras de motores

J_m : Vector de inercias de los motores

B_m : Vector de coeficientes de fricción viscosa de motores

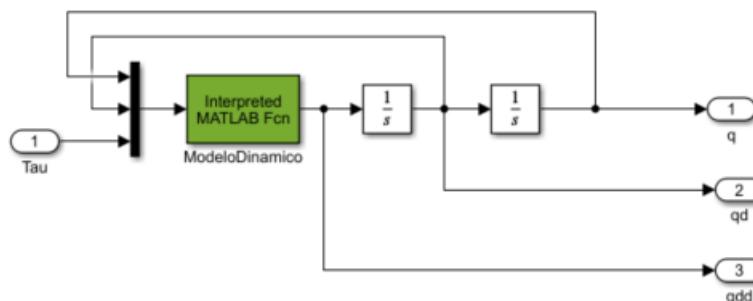
Despejando las aceleraciones articulares obtenemos la siguiente ecuación implementada en nuestro fichero [ModeloDinamico_R3GDL.m](#).

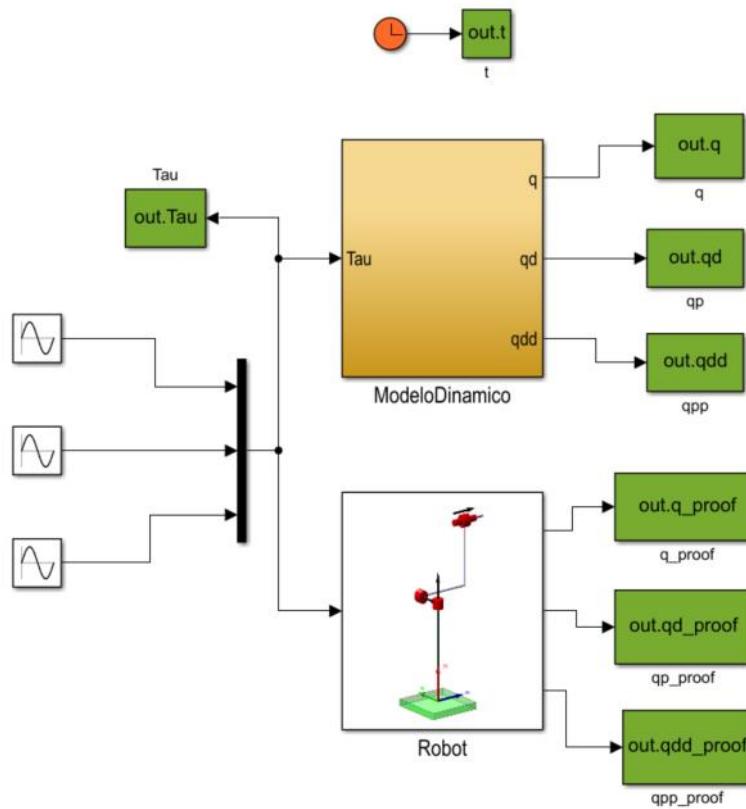
$$\ddot{q} = M_A(q)^{-1} \cdot \left[\tau - V_A(q, \dot{q}) - G_A(q) \right]$$

2. Simulador de dinámica

Una vez que disponemos del modelo dinámico de nuestro robot se ha creado un simulador de la dinámica del mismo, en el cual implementamos e integramos las ecuaciones de la dinámica anteriormente mostrada. Este fichero se encuentra adjunto con el nombre de [SD_R3GDL.m](#). Las funciones que usa son [ModeloDinamico_R3GDL.m](#) y [ModeloDinamico_R3GDL_Proof.p](#).

A continuación, se muestra el esquemático básico que permite simular la ecuación, así como los bloques que componen el modelo de Simulink desarrollado:

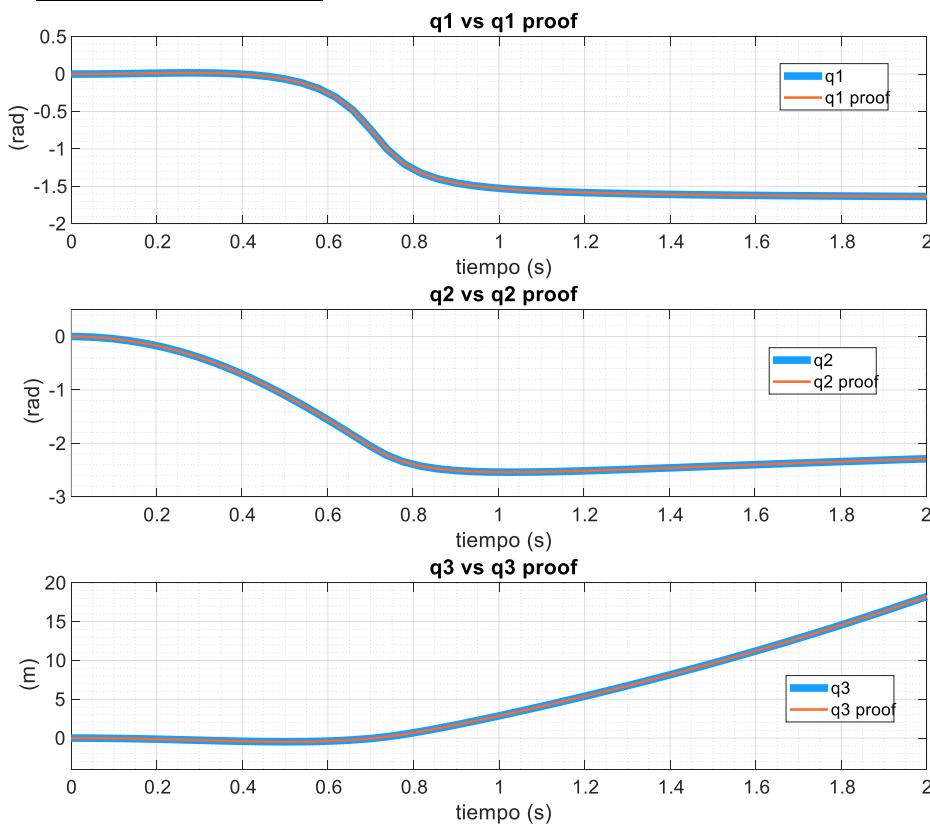




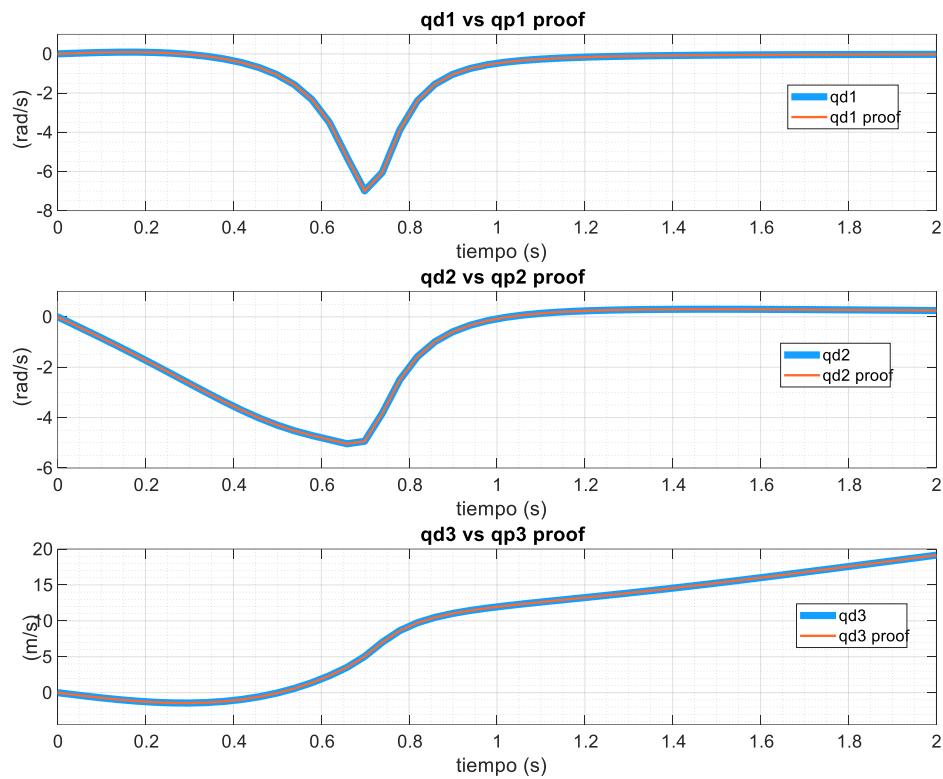
3. Comprobación dinámica

En el presente apartado corroboraremos el buen funcionamiento del simulador y veremos si el modelo dinámico desarrollado es equivalente al dado de prueba. Para ello simuiremos el modelo descrito en el anterior apartado y representaremos gráficamente las respuestas temporales tanto de nuestro modelo como del modelo de prueba proporcionado (que supondrá para nosotros lo equivalente al robot real) de manera conjunta.

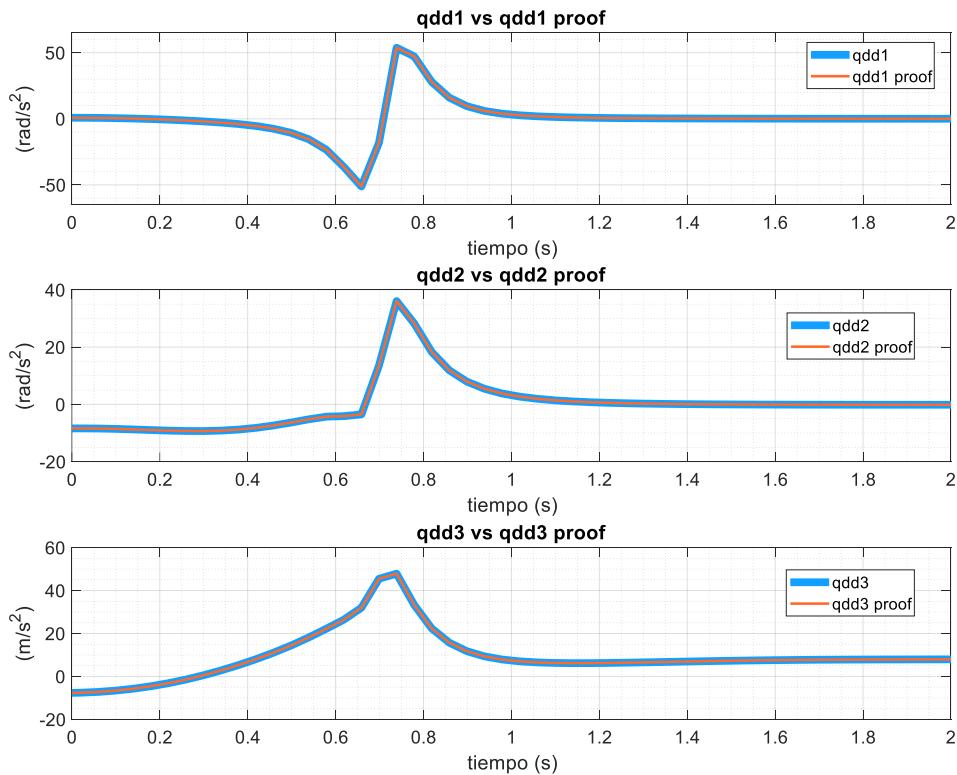
- Posiciones articulares:



- Velocidades articulares:



- Aceleraciones articulares:



También podemos realizar comprobaciones de forma numérica como sigue en este ejemplo:

```
>> ModeloDinamico_R3GDL(ones(1,9))
```

```
ans =
```

```
5.707469658918830
-4.334117669603020
-8.210393326842546
```

```
>> ModeloDinamico_R3GDL_Proof(ones(1,9))
```

```
ans =
```

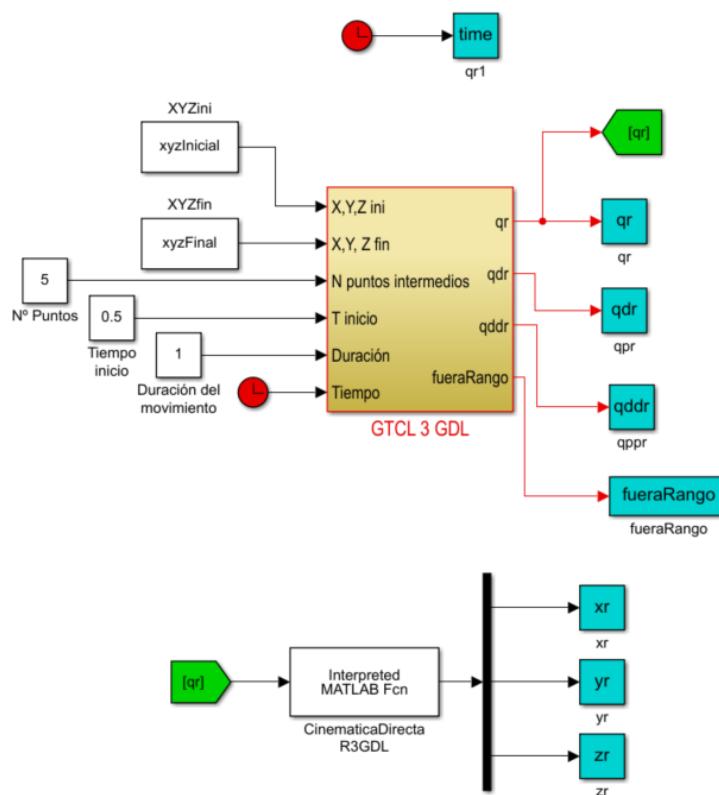
```
5.707465699429767
-4.334114990819662
-8.210387956101698
```

Analizando los resultados gráficos y numéricos anteriores podemos decir que el modelo dinámico desarrollado y el de verificación aportan prácticamente los mismos resultados, es decir, en otras palabras, nuestro modelo se asemeja bastante bien al robot real.

3) CONTROL CINEMÁTICO

1. Generador de trayectorias

Partiendo del generador de trayectorias proporcionado en clase y usando los modelos cinemáticos directo e inverso desarrollados en los apartados anteriores representaremos gráficamente las trayectorias generadas.

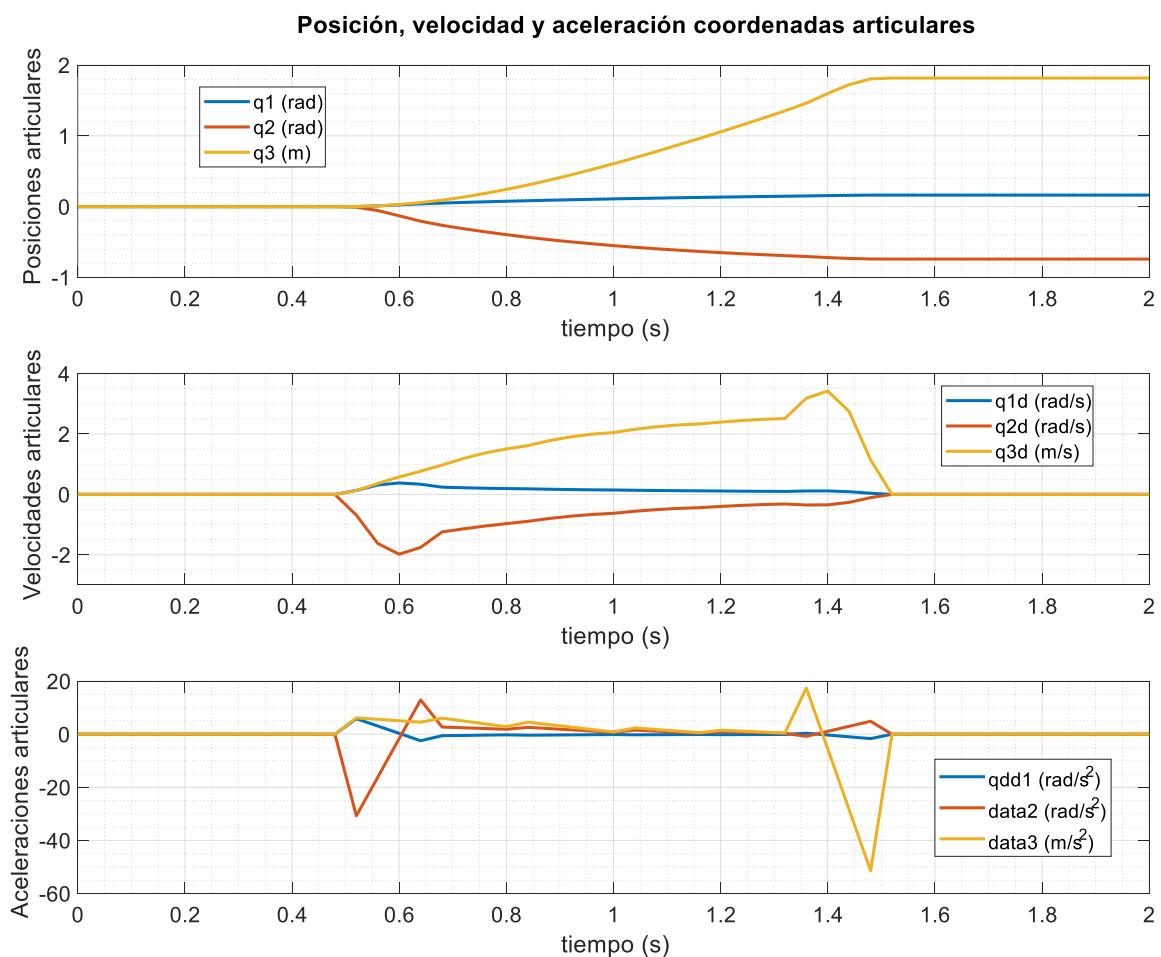


En el esquema anterior podemos ver como a partir de una posición inicial y final, un número de puntos intermedios, el tiempo de inicio, la duración del movimiento y el tiempo transcurrido en la simulación, empleando el fichero GTCL_R3GDL_v2.m implementado dentro del bloque GTCL 3 GDL, podemos obtener la posición, velocidad y aceleración en coordenadas articulares tales que consiguen describir la trayectoria en linea recta entre los puntos inicial y final deseados. Realizando una simulación con los parámetros que aparecen en el esquema de bloques. Donde xyzInicial y xyzFinal corresponden con:

$$\text{xyzInicial} = \begin{bmatrix} 1.5 \\ 0.5 \\ 2.5 \end{bmatrix}; \text{xyzFinal} = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$$

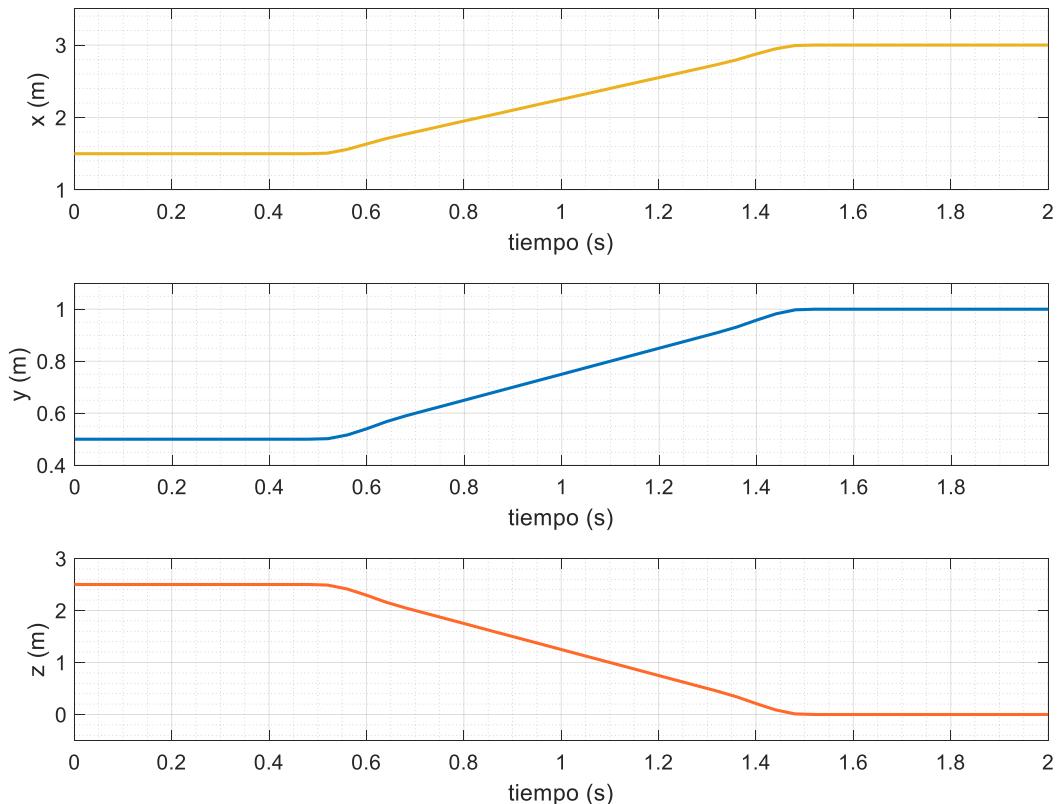
Destacamos que la posición inicial se corresponde con la correspondiente a la que tiene el robot en la posición de HOME.

Pasamos a ver las gráficas de las coordenadas articulares obtenidas de la simulación:

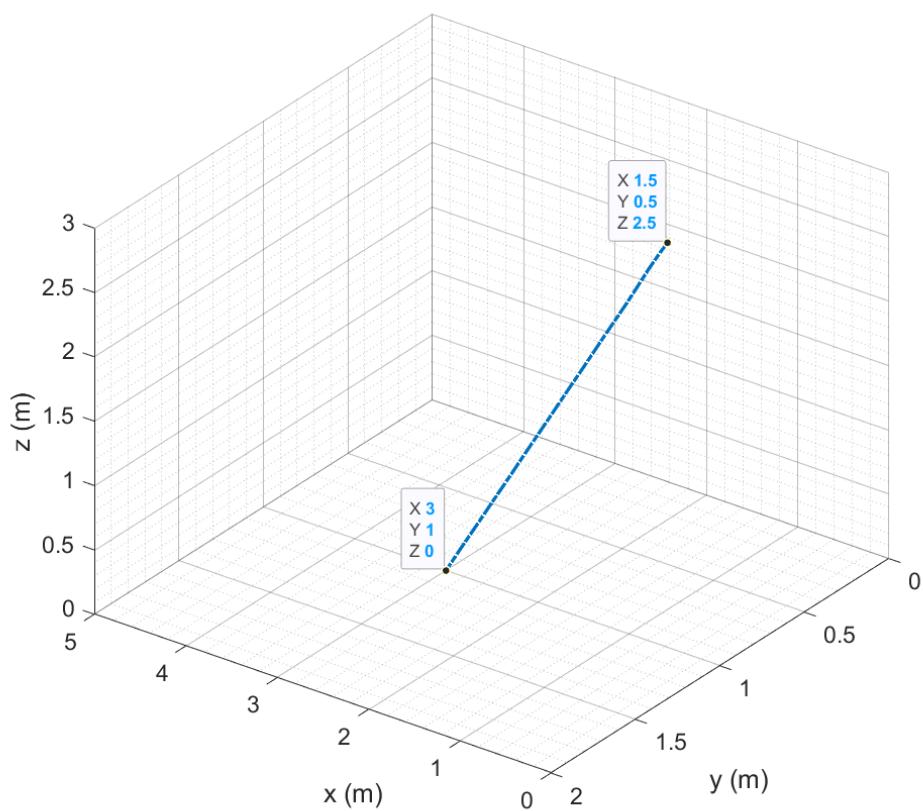


Aplicando la Cinemática Directa a las posiciones articulares obtenidas podemos ver la equivalencia en coordenadas cartesianas y verificar si efectivamente la trayectoria generada es la deseada.

Evolución coordenadas cartesianas



Trayectoria en espacio



4) CONTROL DINÁMICO

En primer lugar, debemos obtener el modelo dinámico linealizado de nuestro robot. Para ello haremos uso de las matrices que conforman nuestro modelo dinámico halladas por medio del método de Newton-Euler.

Aplicando la teoría vista en la asignatura, de forma resumida, gracias al efecto desacoplador de reductoras elevadas en los motores, entre otros, y realizando ciertas simplificaciones podemos hallar las funciones de transferencia que modelan cada uno de los grados de libertad de nuestro robot de la siguiente forma:

$$G_{ii}(s) = \frac{1}{(a_i s + b_i)s} \quad \text{donde } i = 1, 2, 3.$$

Los términos a_i se obtienen considerando el caso más desfavorable, es decir, maximizando los términos de la diagonal de la matriz de inercias M_a del modelo dinámico del robot.

Por otro lado, los términos b_i se consiguen, suponiendo que los términos de Coriolis centrífugos son despreciables, de aquellos términos de las distintas posiciones de la matriz V_a que no contienen productos cruzados de variables articulares, es decir del término que proviene de la parte en la que influyen los parámetros de los motores R y B_m .

Considerando que las articulaciones de rotación tienen un rango de movimiento entre $-\pi$ y π y las de translación entre 0 y 1.5 m, se han obtenido los siguientes valores:

$$\begin{aligned} a_1 &= 381.6195 ; b_1 = 0.08435 \\ a_2 &= 393.0195 ; b_2 = 0.1038 \\ a_3 &= 34.218 ; b_3 = 0.048931 \end{aligned}$$

Una vez obtenido el modelo linealizado se diseñarán distintos controladores aplicando las técnicas de control estudiadas en la asignatura con el fin de poder controlar nuestro robot por medio del modelo dinámico completo. Para el diseño de estos se tomará como especificación un tiempo de subida deseado del bucle cerrado de 0.1 segundos.

1. Diseño controlador PD

$$PD : C_i(s) = Kp_i + Kd_i s$$

$$G_{BAi}(s) = C_i(s) \cdot G_{ii}(s) = \frac{Kp_i + Kd_i s}{(a_i s + b_i)s}$$

$$G_{BCi}(s) = \frac{G_{BAi}(s)}{1 + G_{BAi}(s)} = \frac{Kp_i + Kd_i s}{a_i s^2 + (b_i + Kd_i)s + Kp_i} = \frac{\frac{1}{a_i} \cdot (Kp_i + Kd_i s)}{s^2 + \frac{(b_i + Kd_i)}{a_i}s + \frac{Kp_i}{a_i}}$$

Comparando denominador con la estructura genérica de un sistema de 2º orden : $s^2 + 2\delta w_n \cdot s + w_n^2$

$$s^2 + \frac{(b_i + Kd_i)}{a_i}s + \frac{Kp_i}{a_i} = s^2 + 2\delta w_n \cdot s + w_n^2 \rightarrow \frac{(b_i + Kd_i)}{a_i} = 2\delta w_n ; \frac{Kp_i}{a_i} = w_n^2$$

Tomando para el diseño los valores : $\delta = 1 \rightarrow$ Sistema críticamente amortiguado

Teniendo en cuenta que : $w_n = \frac{6}{t_s}$; Donde t_s : tiempo de subida del bucle cerrado

Podemos finalmente obtener los parámetros del controlador con las siguientes expresiones :

$$Kp_i = \frac{36 \cdot a_i}{t_s^2} ; Kd_i = \frac{12 \cdot a_i}{t_s} - b_i$$

El cálculo de los parámetros de los controladores lo realizamos mediante el fichero creado con el nombre *calculo_controlador.m*. Se obtienen los siguientes parámetros para este primer controlador PD descentralizado:

$$K_{p1} = 1.3738 \cdot 10^6 ; \quad K_{d1} = 4.5794 \cdot 10^4 ; \quad K_{i1} = 0$$

$$K_{p2} = 1.4149 \cdot 10^6 ; \quad K_{d2} = 4.7162 \cdot 10^4 ; \quad K_{i2} = 0$$

$$K_{p3} = 1.2318 \cdot 10^5 ; \quad K_{d3} = 4.1061 \cdot 10^3 ; \quad K_{i3} = 0$$

2. Diseño controlador PID

$$\text{PID : } C_i(s) = K_{pi} + K_{di}s + K_{li}\frac{1}{s} = \frac{Kc^*(1 + T_1 s)(1 + T_2 s)}{s}$$

Considerando las siguientes relaciones :

$$K_{li} = Kc^*$$

$$K_{di} = K_{li} \cdot T_1 \cdot T_2$$

$$K_{pi} = K_{li} \cdot (T_1 + T_2)$$

Imponiendo cancelación de polos :

$$G_{BAi} = \frac{Kc^*}{b_i} \cdot \frac{(1 + T_1 s)(1 + T_2 s)}{\left(\frac{a_i}{b_i} s + 1\right)s^2} \text{ con } T_1 = \frac{a_i}{b_i}$$

$$G_{BCi} = \frac{G_{BAi}(s)}{1 + G_{BAi}(s)} = \frac{\frac{Kc^*}{b_i}(1 + T_2 s)}{\frac{Kc^*}{b_i}(1 + T_2 s) + s^2}$$

Comparando denominador con la estructura genérica de un sistema de 2º orden : $s^2 + 2\delta w_n \cdot s + w_n^2$

$$\frac{Kc^*}{b_i}(1 + T_2 s) + s^2 = s^2 + 2\delta w_n \cdot s + w_n^2$$

Tomando para el diseño los valores : $\delta = 1 \rightarrow$ Sistema críticamente amortiguado

$$\text{Teniendo en cuenta que : } w_n = \frac{6}{t_s} ; \text{ Donde } t_s : \text{tiempo de subida del bucle cerrado}$$

$$Kc^* = \frac{36 b_i}{t_s^2} ; \quad T_2 = \frac{12 b_i}{t_s Kc^*} = \frac{t_s}{3}$$

Obtenidos los parametros Kc^* , T_1 y T_2 podemos tener de forma inmediata K_{li} , K_{di} y K_{pi} mediante las relaciones anteriormente vistas.

Nuevamente, en el fichero *calculo_controlador.m* se han recogido las expresiones anteriores consiguiendo finalmente los valores de los parámetros del controlador PID descentralizado:

$$K_{p1} = 1.3738 \cdot 10^6 ; \quad K_{d1} = 4.5794 \cdot 10^4 ; \quad K_{i1} = 303.66$$

$$K_{p2} = 1.4149 \cdot 10^6 ; \quad K_{d2} = 4.7162 \cdot 10^4 ; \quad K_{i2} = 373.68$$

$$K_{p3} = 1.2319 \cdot 10^5 ; \quad K_{d3} = 4.1062 \cdot 10^3 ; \quad K_{i3} = 176.1516$$

3. Diseño controlador por Par Calculado

Basándonos en la ley de control vista en teoría:

$$\text{Modelo del sistema: } \tau = M_A(q)\ddot{q} + C_A(q, \dot{q})\dot{q} + G_A(q)$$

$$\text{Control: } \tau = M_A(q)(\ddot{q}_r + u) + C_A(q, \dot{q})\dot{q} + G_A(q)$$

$$\text{Bucle cerrado interno: } \begin{aligned} \ddot{\tilde{q}} &= -u && (\text{Dinámica del error:}) \\ \tilde{q} &= q_r - q && (\text{DOBLE INTEGRADOR DESACOPLADO}) \end{aligned}$$

$$\text{Control de bucle interno: } u = K_P \tilde{q} + K_D \dot{\tilde{q}} \quad (\text{Diseño de PD/PID para } \ddot{q}=u)$$

Como idea principal recordamos que se pretende realizar una linealización por realimentación. Esta técnica de control no hace uso del modelo linealizado obtenido, sino que directamente linealiza el modelo completo que tenemos a través del desacople de las interacciones del robot.

Diseño parámetros del control interno (sistema 2º orden) :

$$K_P = w_n^2 \rightarrow K_P = \frac{36}{t_s^2}$$

$$K_D = 2\delta w_n (\delta = 1) \rightarrow K_D = \frac{12}{t_s}$$

Hallando numéricamente dichos parámetros, resultan:

$$K_{P1} = 3.6 \cdot 10^3 ; \quad K_{D1} = 120 ; \quad K_{I1} = 0$$

$$K_{P2} = 3.6 \cdot 10^3 ; \quad K_{D2} = 120 ; \quad K_{I2} = 0$$

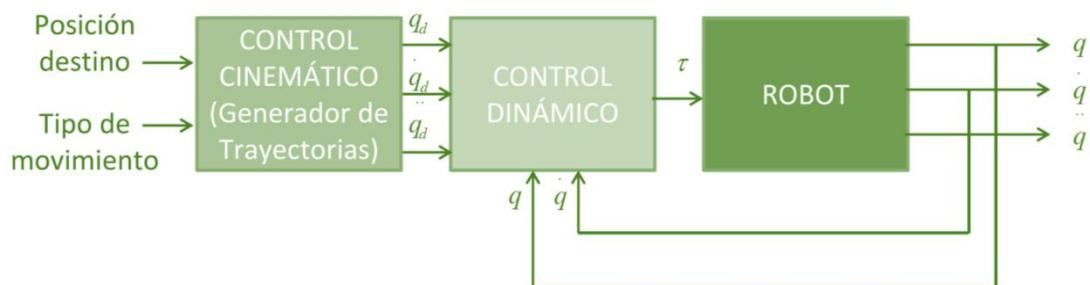
$$K_{P3} = 3.6 \cdot 10^3 ; \quad K_{D3} = 120 ; \quad K_{I3} = 0$$

Dichas expresiones se encuentran también recogidas en el fichero [calculo_controlador.m](#)

4. Resultados del control

Tras haber realizado el diseño de los distintos controladores pasamos a su implementación en bucle cerrado para tratar de controlar nuestro robot o, mejor dicho, el modelo dinámico completo que obtuvimos de éste.

Recordando el esquema general estudiado en la asignatura:

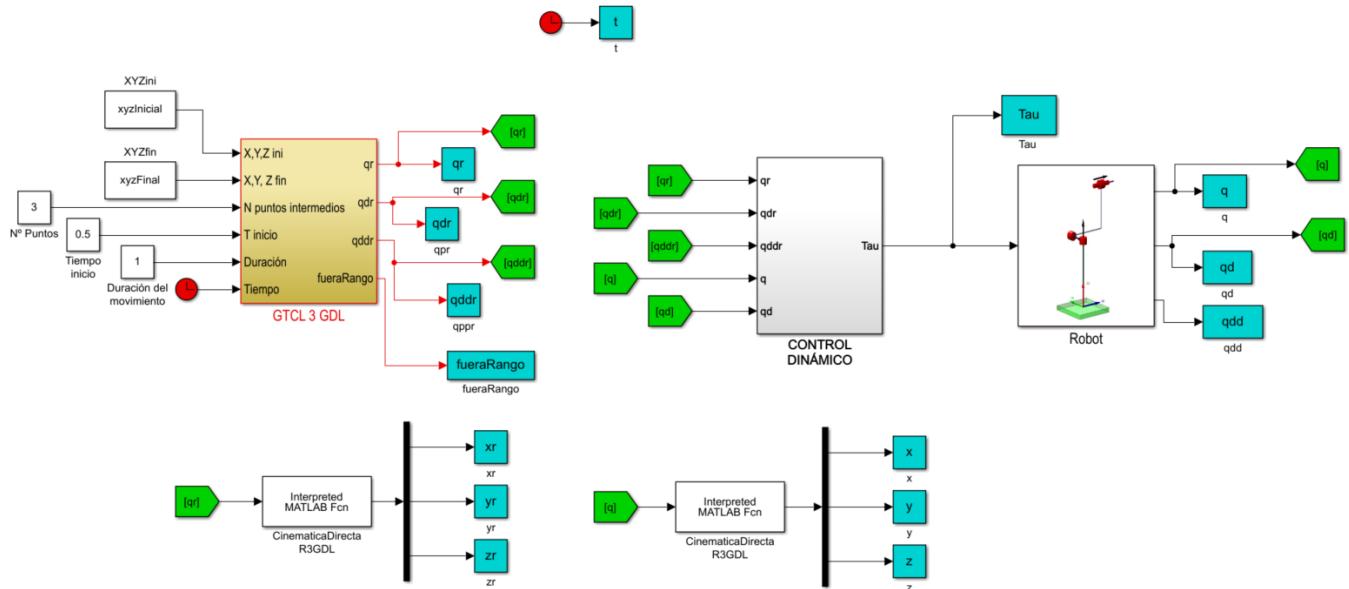


El bloque generador de trayectorias proporcionará al control las referencias deseadas en coordenadas articulares equivalente a la posición en cartesianas requerida en cada momento para que el extremo del robot se mueva desde la posición HOME hasta un punto situado en un incremento de coordenadas cartesianas $[\Delta X, \Delta Y, \Delta Z] = [-0.15, -0.15, 0.15]$ siguiendo una trayectoria en linea recta.

El bloque de control, a partir del error existente entre las referencias provenientes del bloque generador y la realimentación de la posición y velocidad en coordenadas articulares será capaz de generar el par necesario que se debe aplicar a las articulaciones del robot.

Como ya vimos anteriormente, el modelo dinámico hace uso de la señal de control para obtener a la salida la aceleración articular y mediante el uso de la estructura con integradores vista, obtener finalmente la posición y velocidad articular, las cuales a su vez también son entradas del propio modelo dinámico ya que son necesarias para el cálculo de las matrices M_a , V_a y G_a que lo describen.

El esquema que se empleará en Simulink para realizar todas las futuras simulaciones será el usado en clase adaptado para nuestro robot particular:



Para dotar al montaje de los valores necesarios para la simulación es preciso hacer uso del fichero [inicializacion.m](#).

Todas las gráficas que se mostrarán en los sucesivos apartados se han generado a partir de los resultados de las simulaciones mediante el fichero [graficas.m](#)

En todas las simulaciones se tomará un total de 3 puntos intermedios en la generación de la trayectoria.

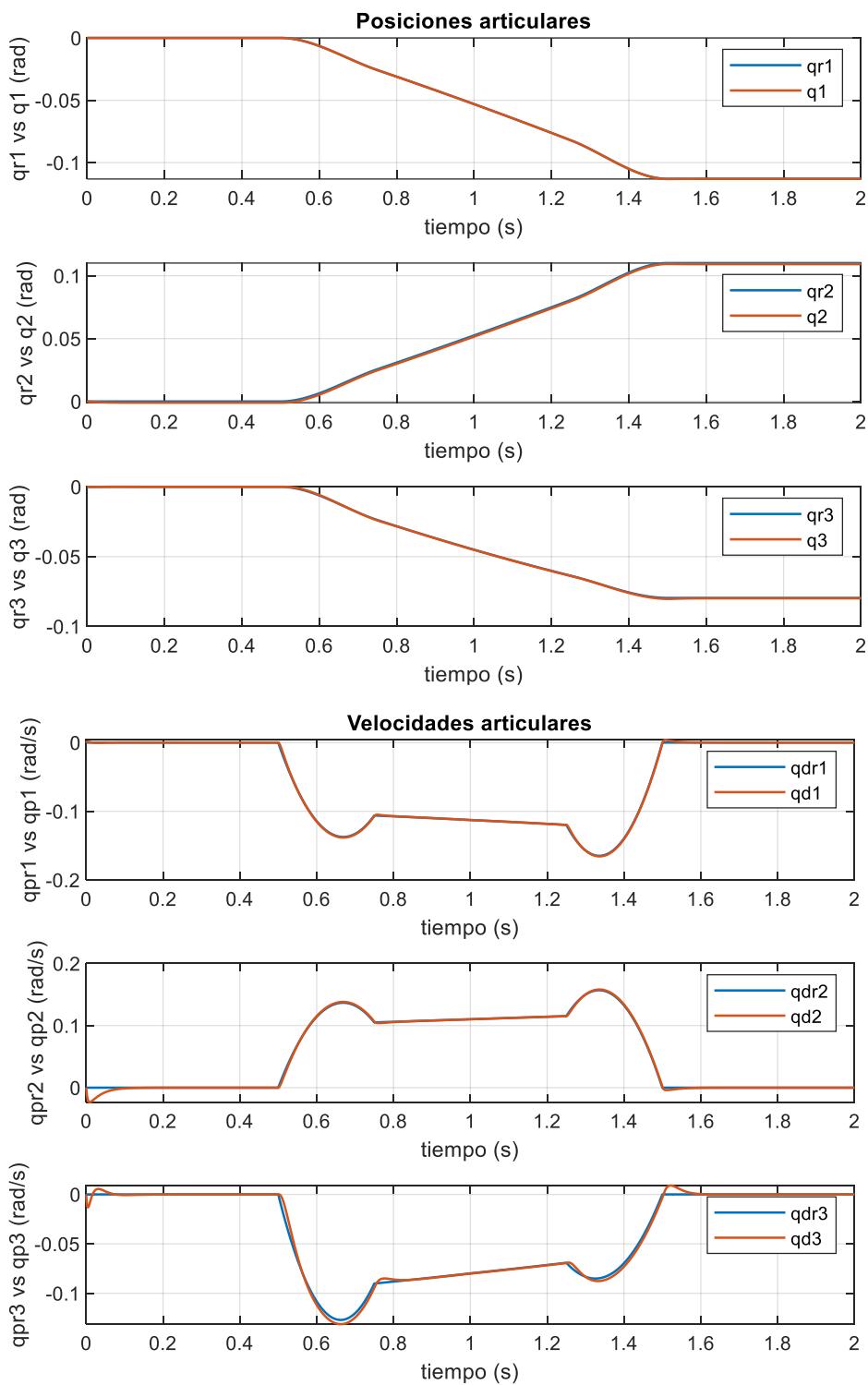
La duración del movimiento podrá variar para conseguir distintas velocidades de movimiento. Ese efecto será estudiado en un apartado dedicado exclusivamente a ello. En los previos se tomará por defecto una duración de movimiento de 1 segundo.

4.1. Controlador PD

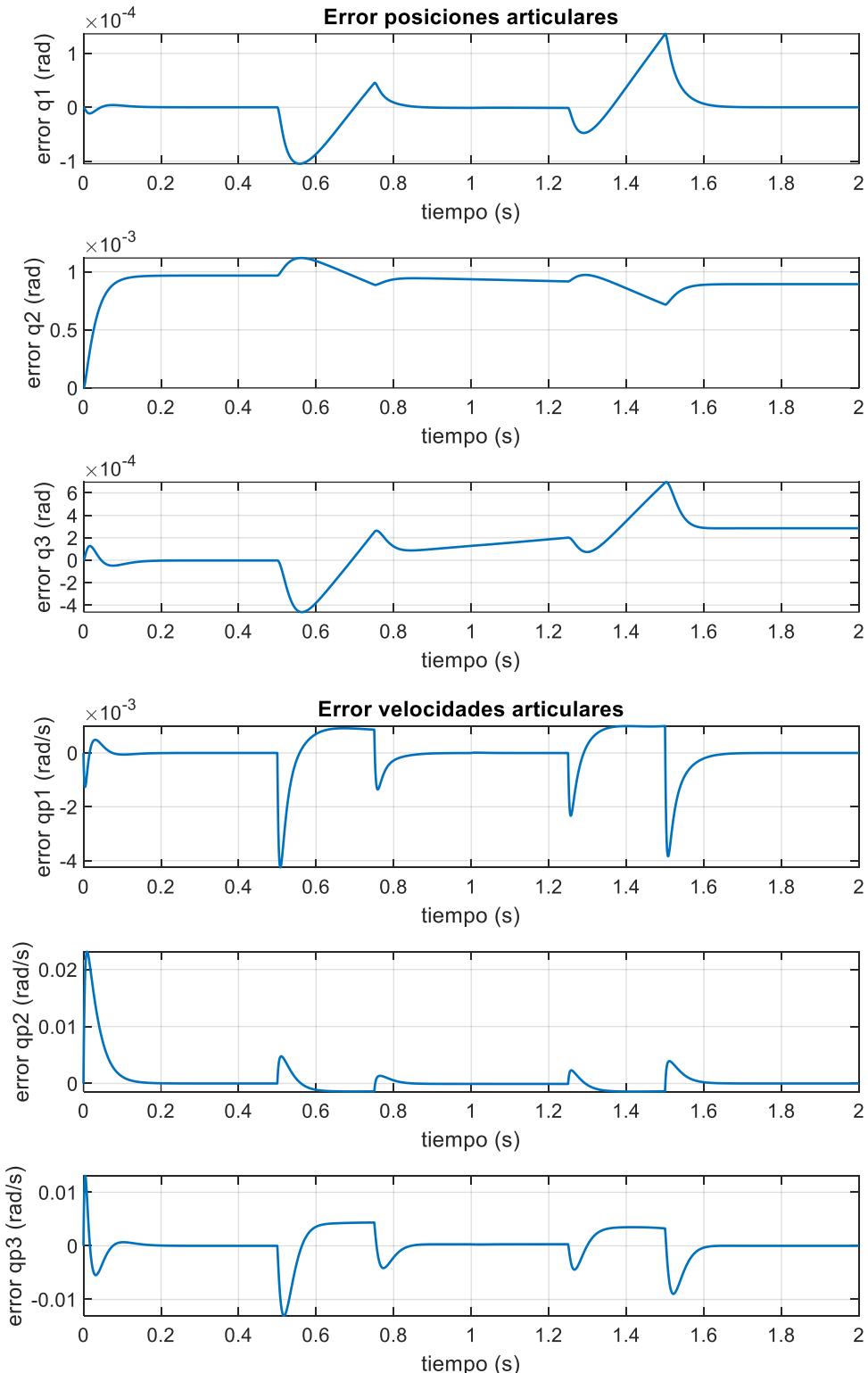
Se partirá del controlador PD descentralizado básico diseñado sin aplicar ninguna compensación. Una vez obtenido sus resultados se tratará de mejorar el control del sistema haciendo uso de algunas de las técnicas de compensación estudiadas. De esta manera se pretende demostrar la ventaja que supone su uso, siempre que se disponga del modelo necesario para ello.

- **Controlador PD sin compensación**

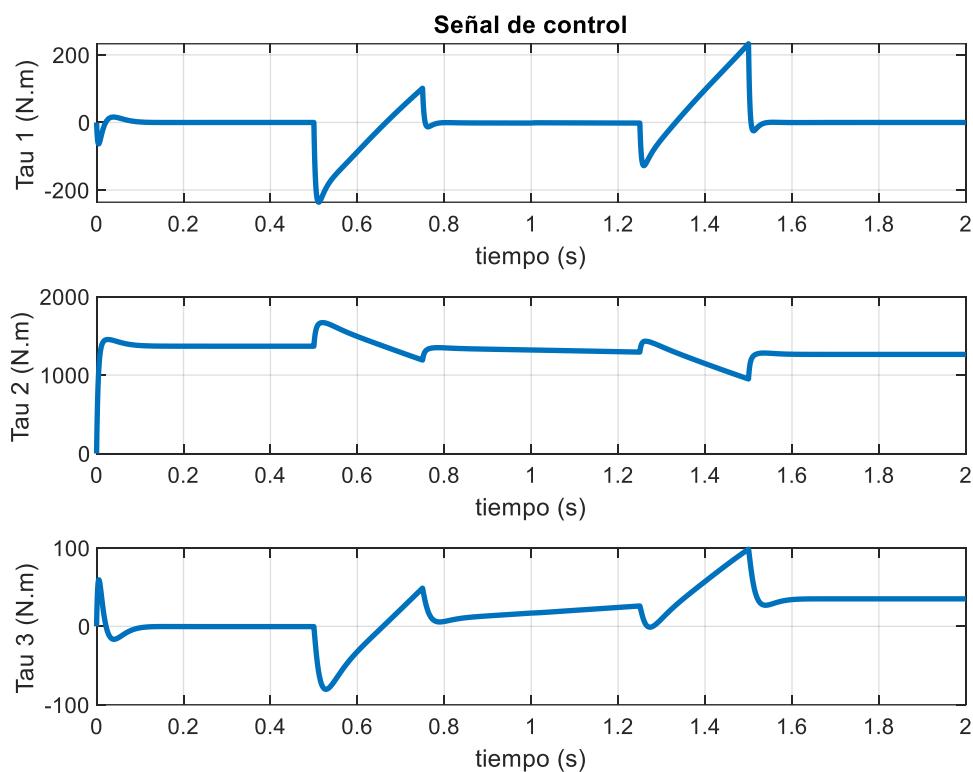
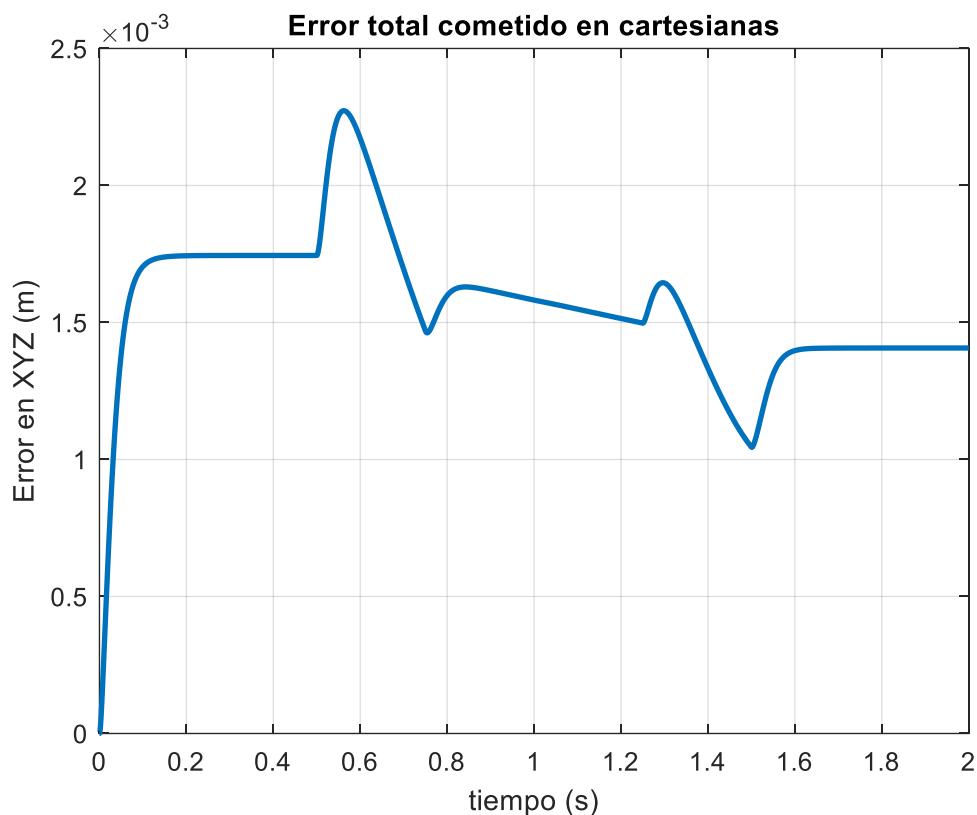
Los resultados obtenidos de la simulación son los siguientes:



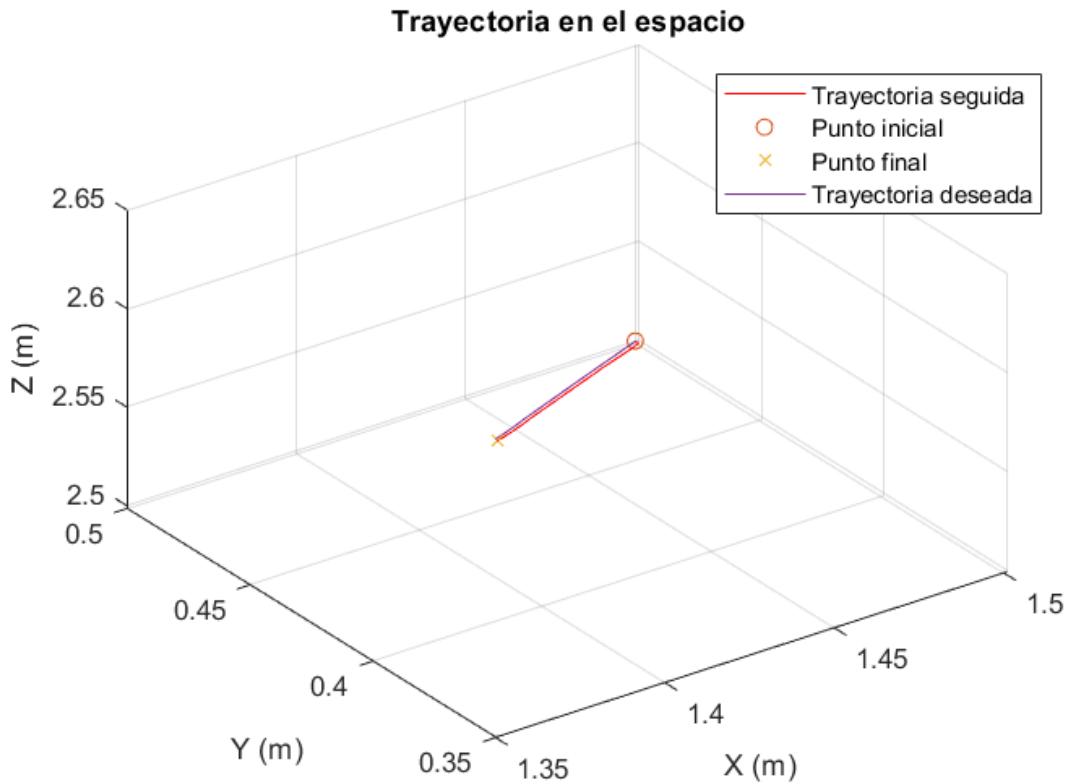
En cuanto a la posición y velocidad en coordenadas articulares se observa una pequeña discrepancia entre la referencia y lo obtenido en la salida.



El error obtenido es lo previsible al tratarse de un PD. Se acumula un error durante el tiempo por la propia estructura que al no incorporar un integrador no tiene capacidad de compensar las perturbaciones permanentes provenientes del efecto gravitatorio. De ahí que se observe error residual en régimen permanente en posición. Se corregirá en próximos controladores.



En la gráfica anterior puede verse la evolución del par de salida del controlador el cual se aplica sobre el robot.

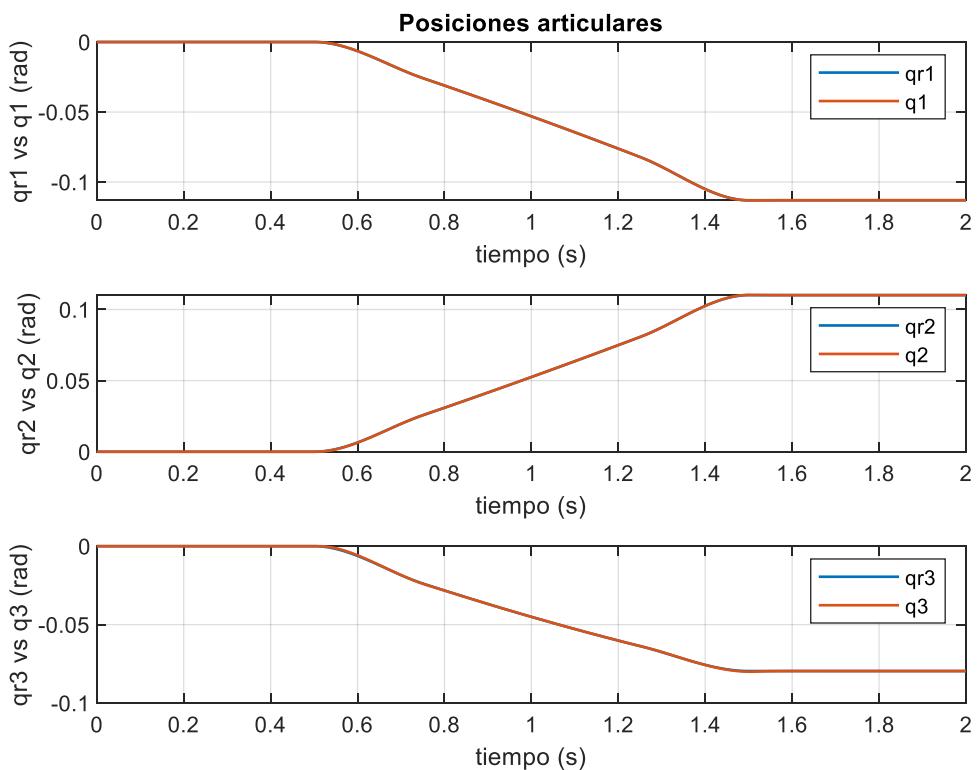


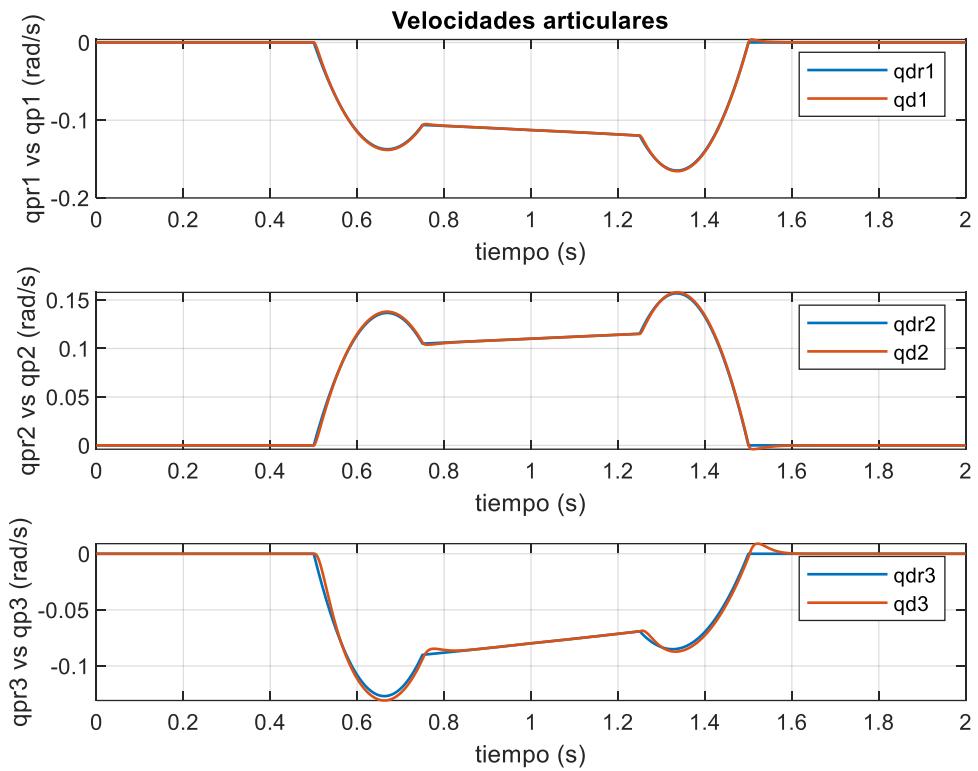
Finalmente, podemos decir que, a pesar de ser la técnica de control más sencilla de las estudiadas, el comportamiento es bastante bueno, cumpliendo las especificaciones requeridas. Sin embargo, siendo posible, conviene poder mejorarlo.

- **Controlador PD + Compensación de gravedad**

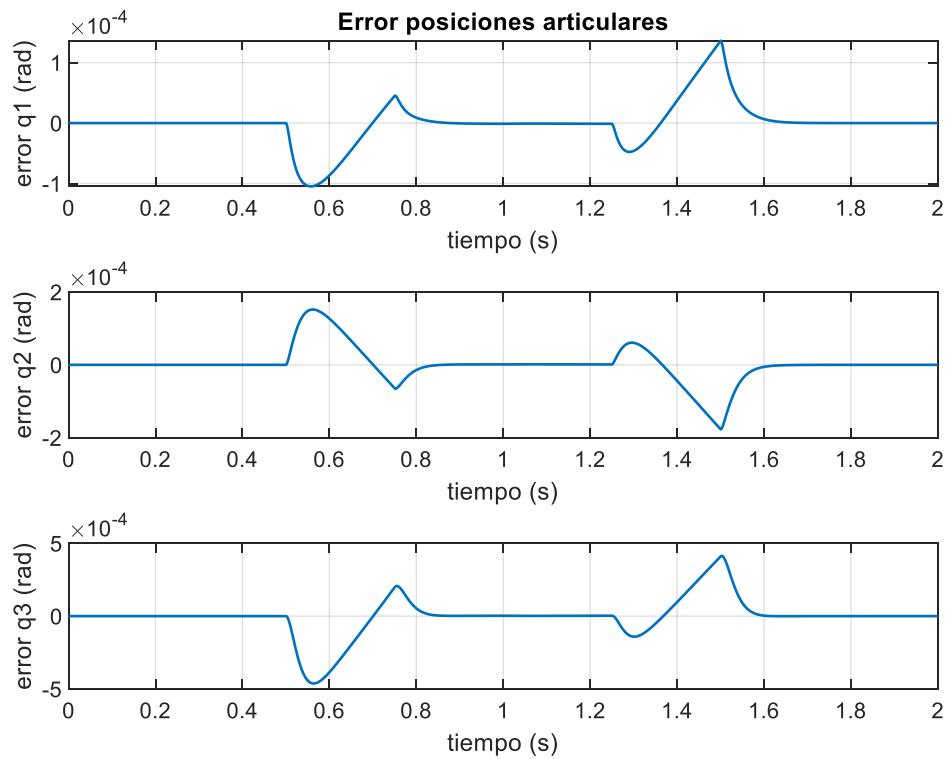
Aprovechándonos del conocimiento del modelo dinámico que tenemos, incorporamos al control PD puro un término adicional que hace cuenta del efecto de la gravedad logrando cancelarlo. Esto sería equivalente a controlar un robot que no ve el efecto de la gravedad.

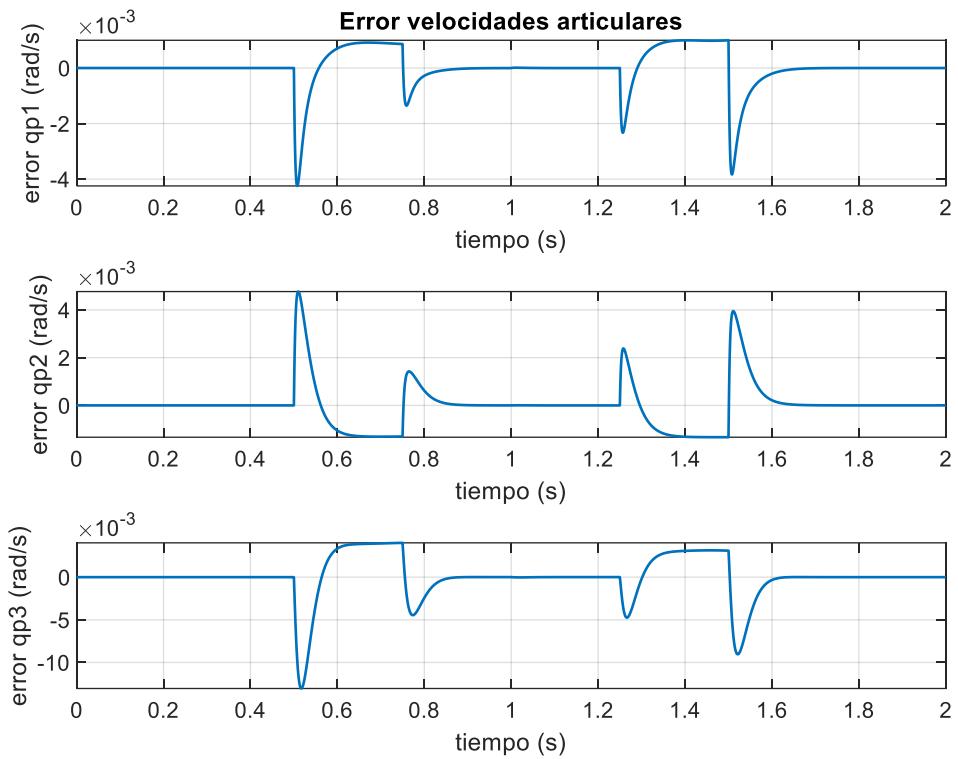
Para ello hacemos uso del fichero [Compensador.m](#) el cual se encuentra implementado dentro del bloque del controlador, añadiendo ahí ese par adicional.



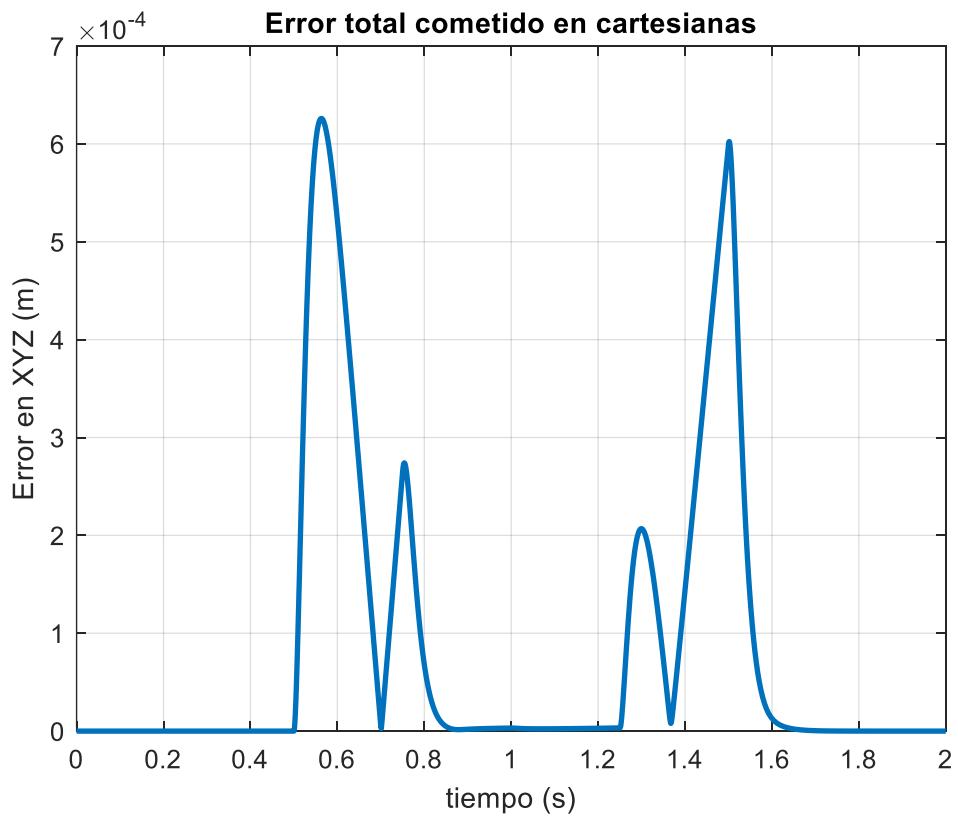


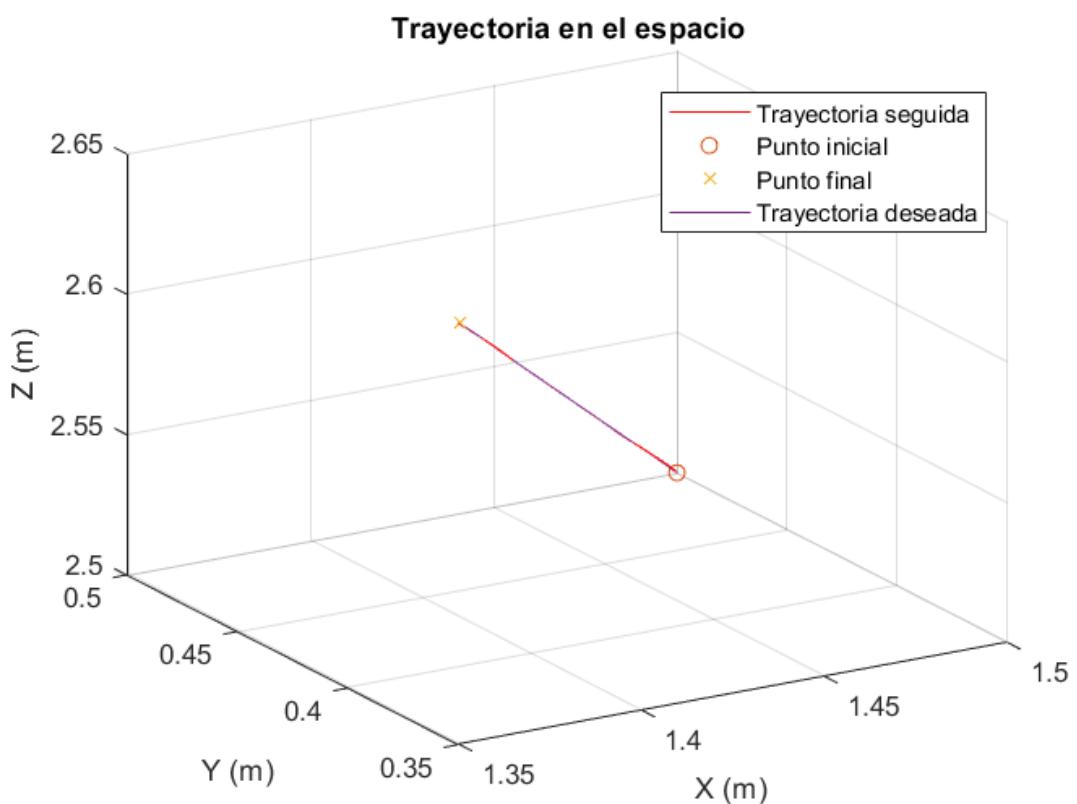
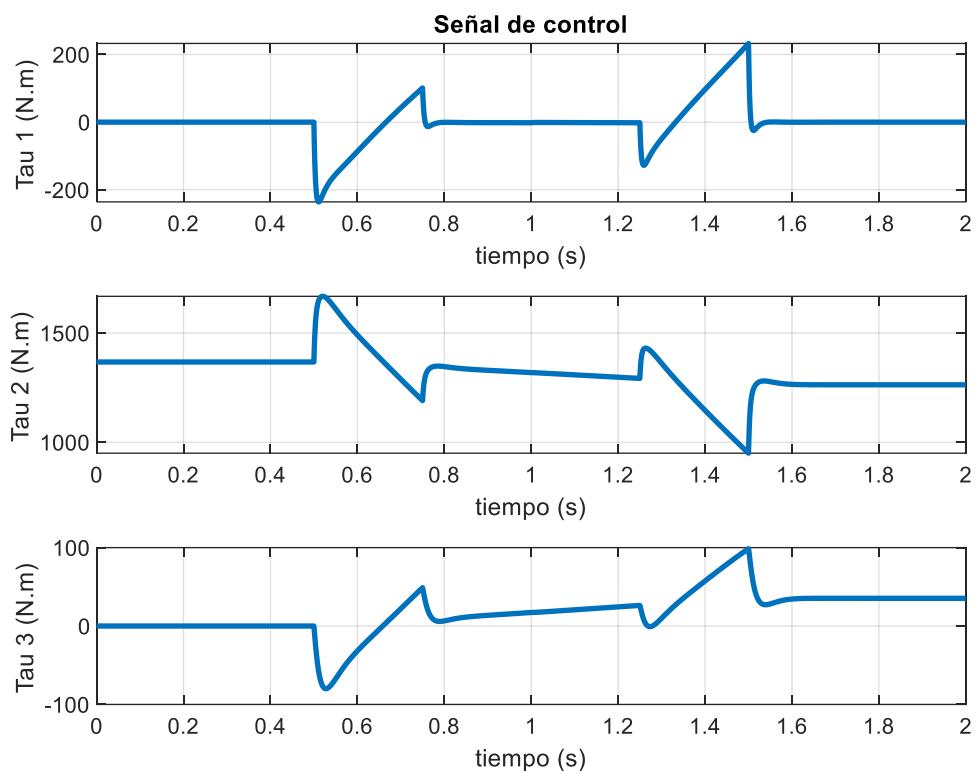
Vemos una mejora notoria acercándose mucho más a la referencia, esto puede verse mejor en las gráficas correspondientes a los errores:





Ahora si el error residual se va a cero a diferencia de antes, de esta forma incluso el PD es capaz de alcanzar error nulo sin ser PID.



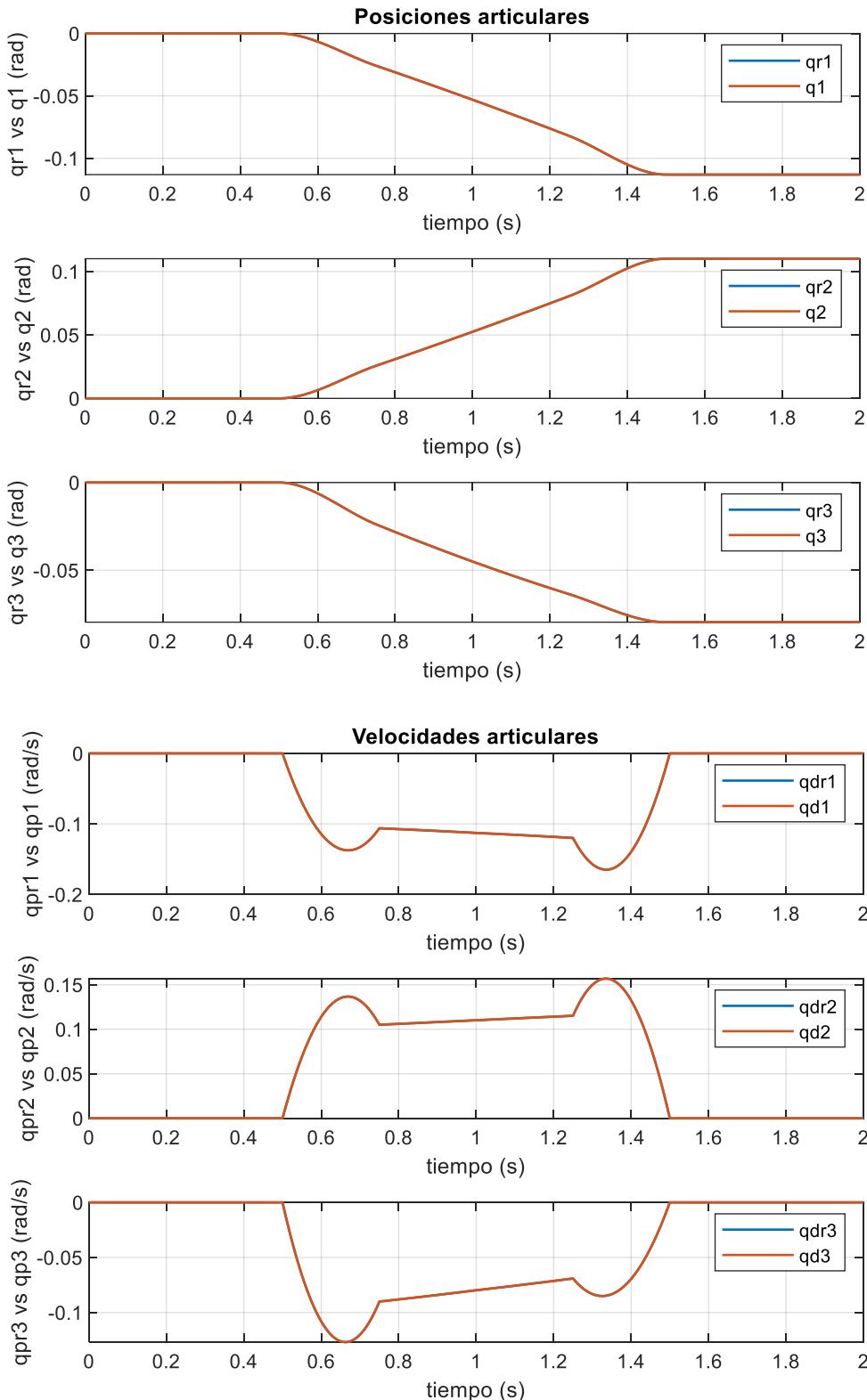


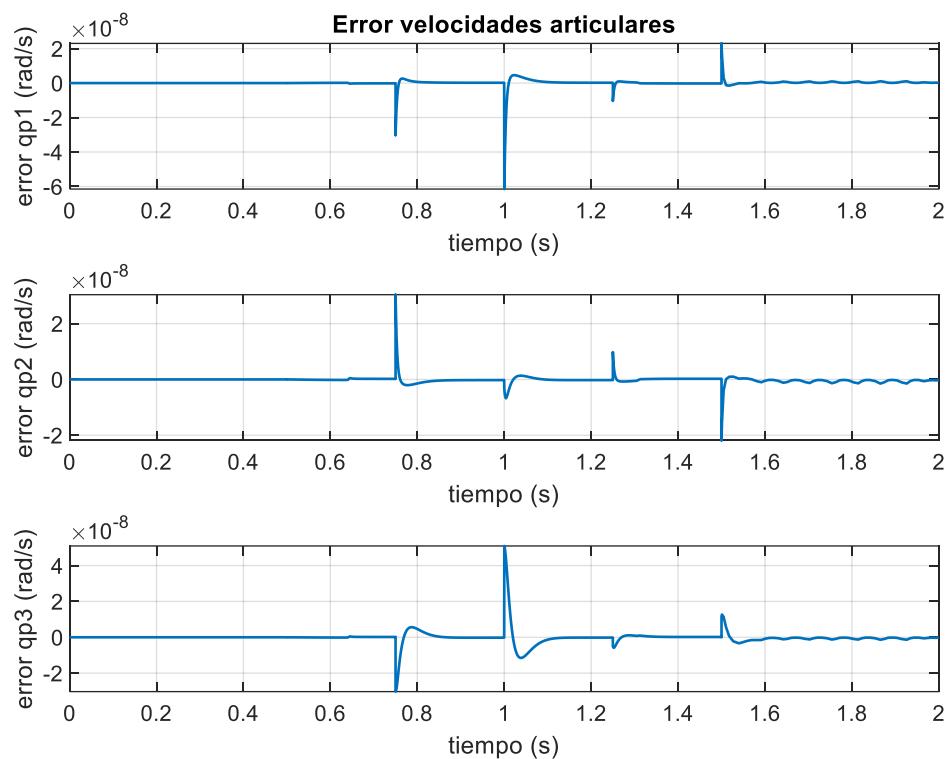
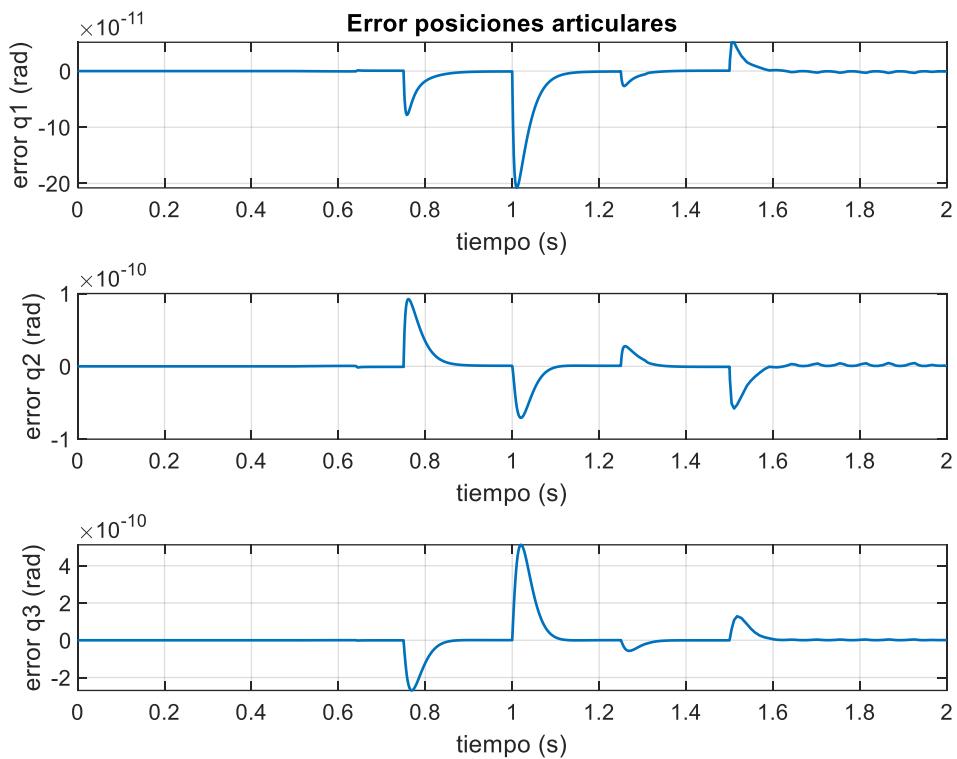
- **Controlador PD + precompensación dinámica**

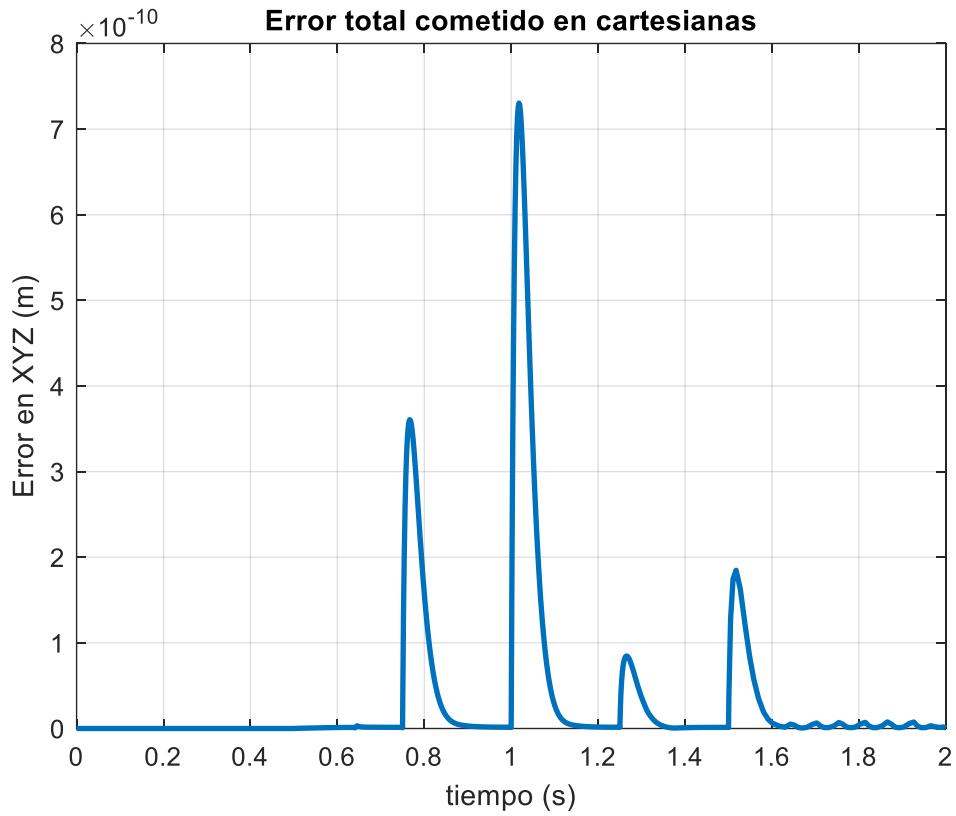
Podemos mejorar aún más el control añadiendo la precompensación de dinámica. En este caso, los parámetros de control, más concretamente K_{Di} , ya no depende de los términos b_i del modelo linealizado.

$$Kp_i = \frac{36 \cdot a_i}{t_s^2} ; Kd_i = \frac{12 \cdot a_i}{t_s}$$

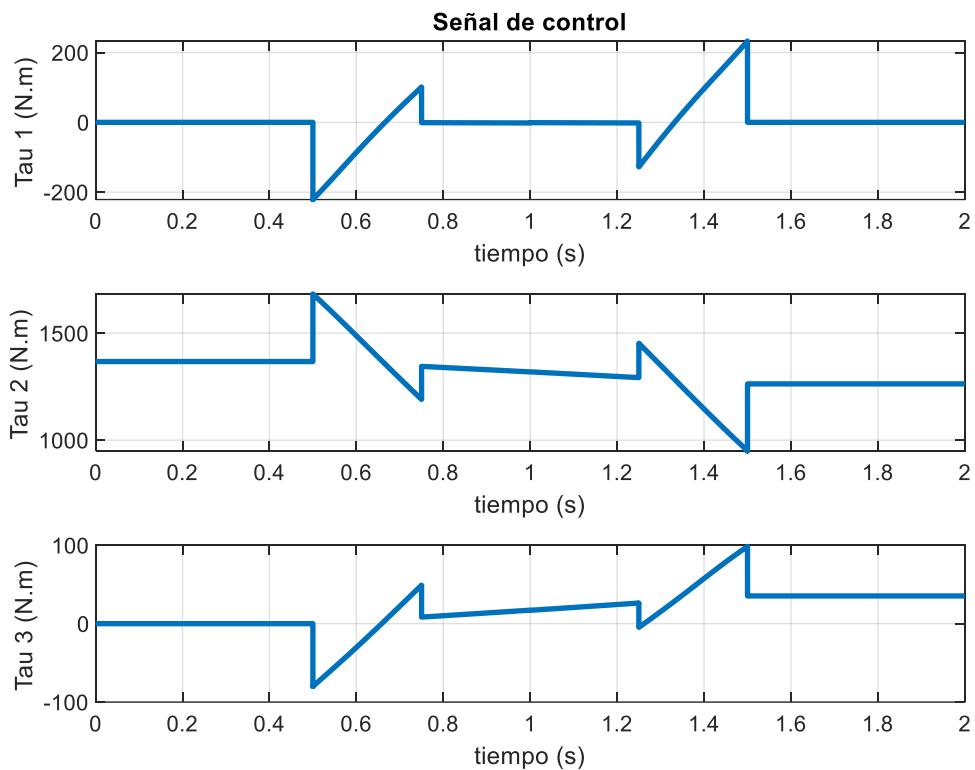
Podemos apreciar mejoras debido a que no solo se compensa el efecto de la gravedad, sino que también, así como Coriolis y parte de las inercias.



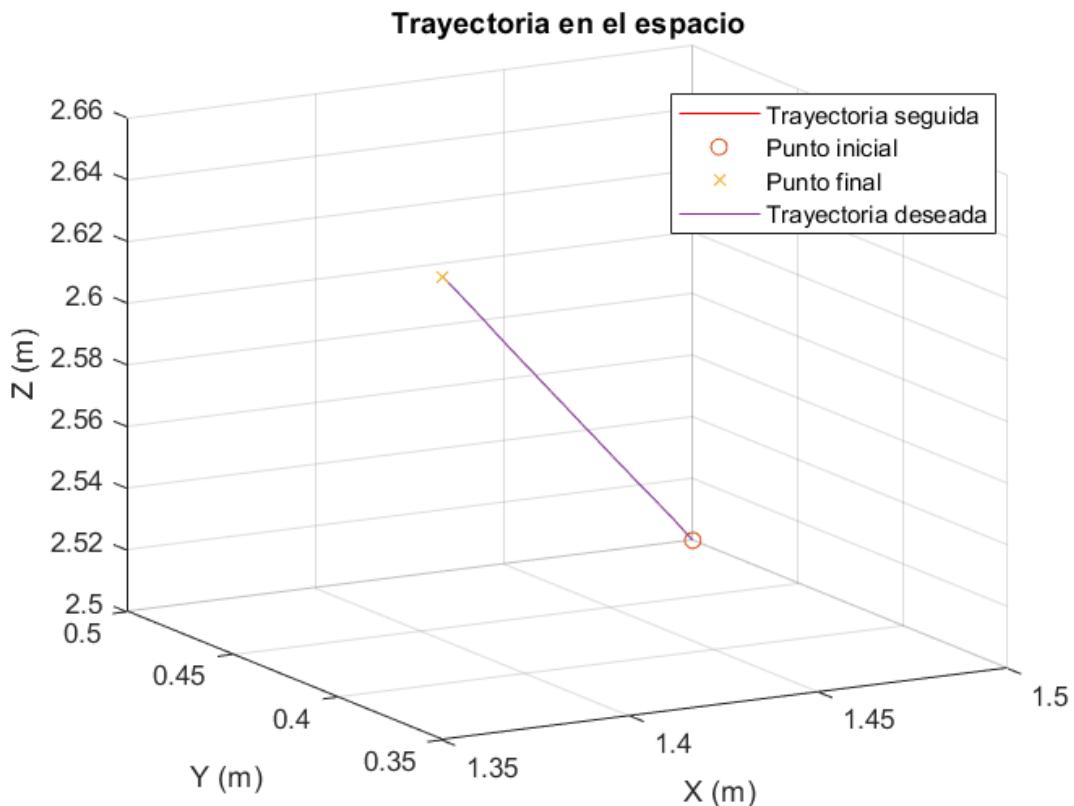




Vemos como el error cometido se hace muy pequeño comparado con los controles anteriores, mejorando unos 6 órdenes de magnitud.



Observamos transiciones más abruptas de la señal de control, en otras palabras, el control es bastante más agresivo.



La trayectoria seguida y deseada prácticamente coinciden.

Como hemos podido comprobar, siempre es mejor compensar lo que conocemos frente a no hacerlo, debido a las mejoras en precisión en el seguimiento de la trayectoria y disminuciones de los errores que se logran por medio de la compensación.

Sin embargo, para un robot real los resultados difícilmente serían tan ideales como los obtenidos ya que no suele tenerse un modelo tan preciso del robot real como para que los efectos de las compensaciones mejoren tanto el seguimiento de la trayectoria. En nuestro caso es así puesto que usamos como robot real el propio modelo del mismo.

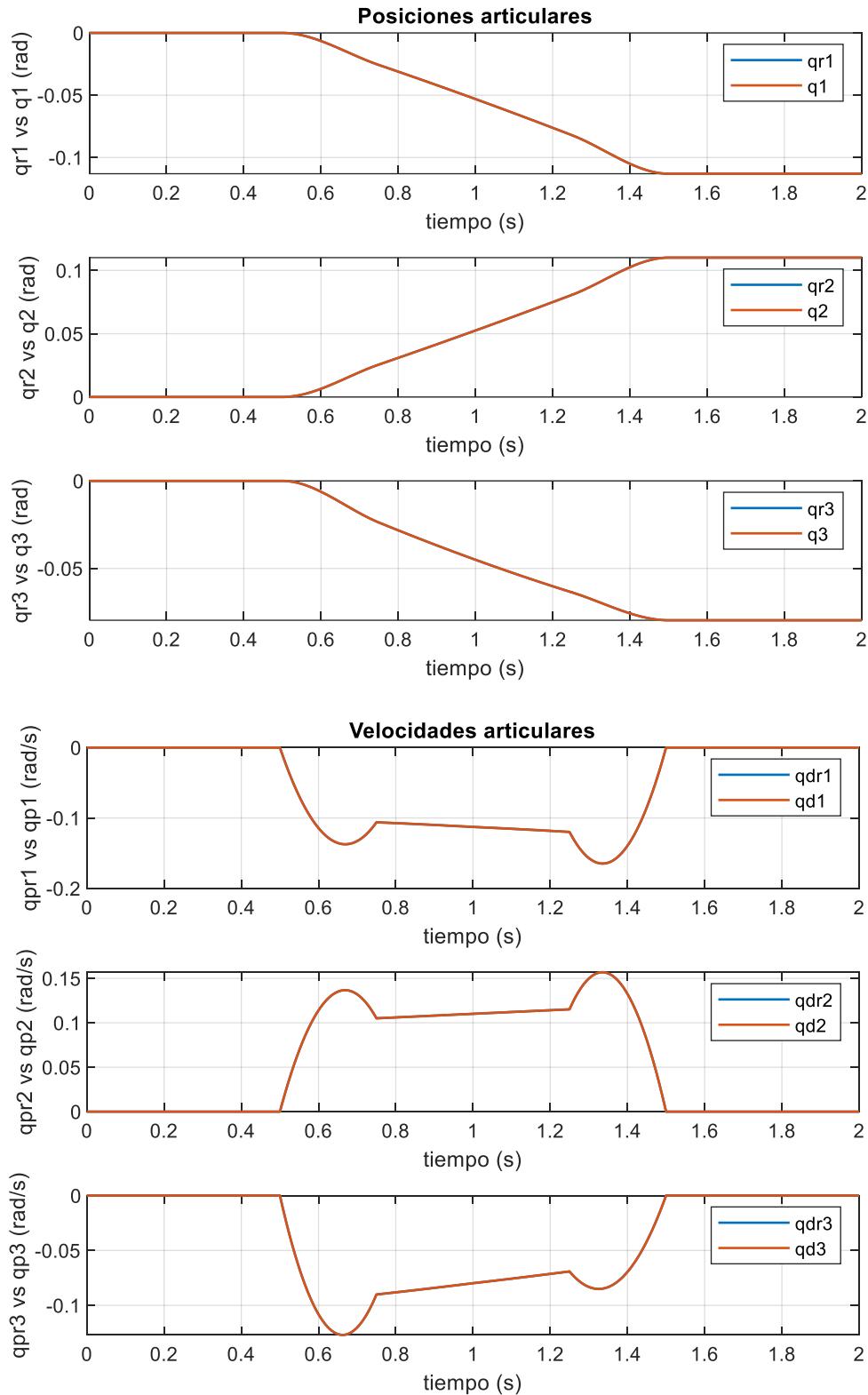
4.2. Controlador PID

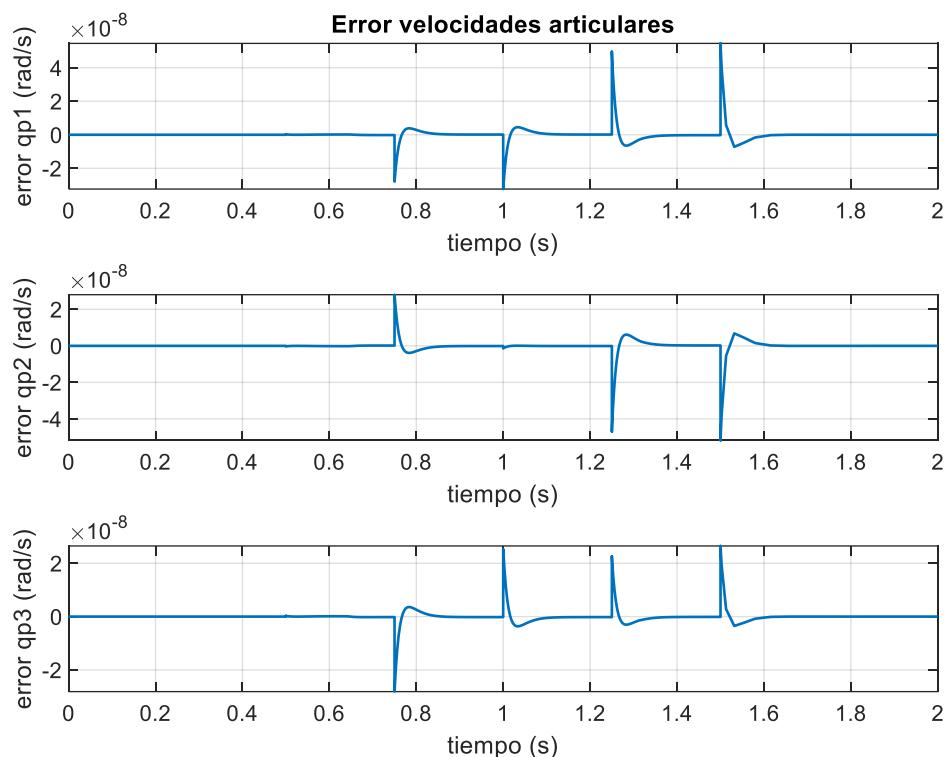
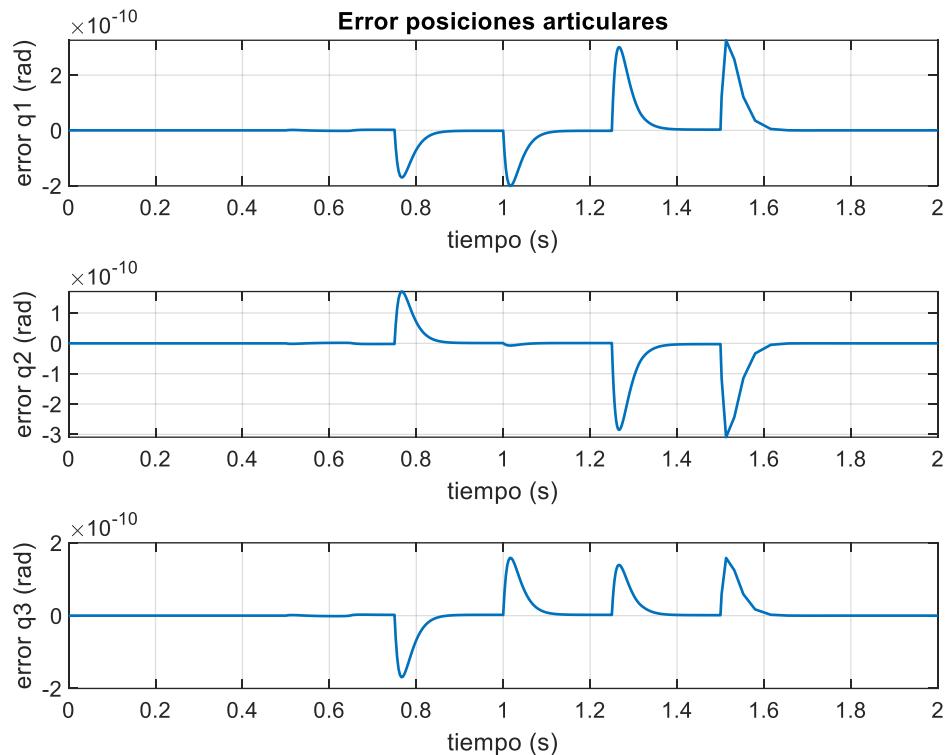
Continuamos con las simulaciones asociadas esta vez al controlador PID. Se espera obtener resultados similares al PD, con una mejora del error, que debería tender a cero en el régimen permanente. Este efecto equivale, por ejemplo, a realizar una compensación de gravedad sobre el controlador PD descentralizado.

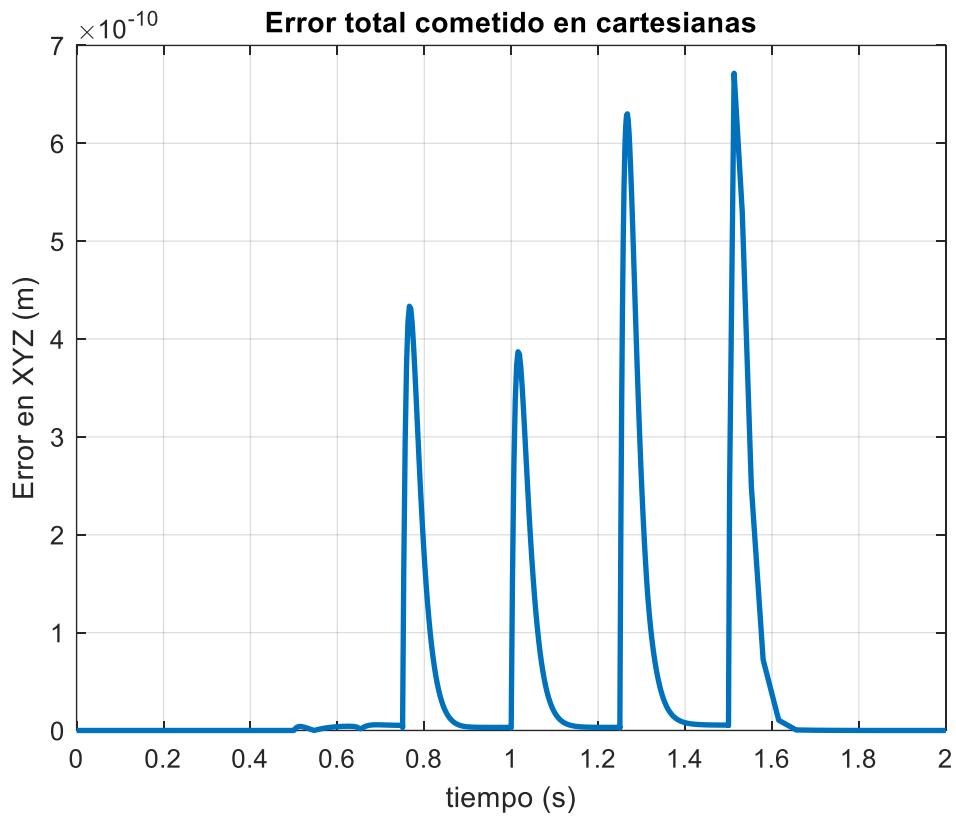
En contraposición, realizando simulaciones nos topamos con que implementando el control PID con los parámetros hallados anteriormente no supone ninguna mejora notable, y a priori, parece que el error no tiende a ser nulo. Ante este problema, tras analizar las constantes calculadas, deducimos que este resultado pueda deberse a un término integral demasiado pequeño. Si la constante correspondiente al término integral es muy reducida probablemente no apreciemos que el error tienda a anularse.

Consideramos en este caso conveniente modificar la constante correspondiente con el propósito de conseguir esa mejora buscada. Decidimos por tanto aumentar de forma considerable K_i . Observamos, mediante sucesivas simulaciones, que a medida que aumentamos dicho parámetro se consigue alcanzar la anulación del error más rápidamente.

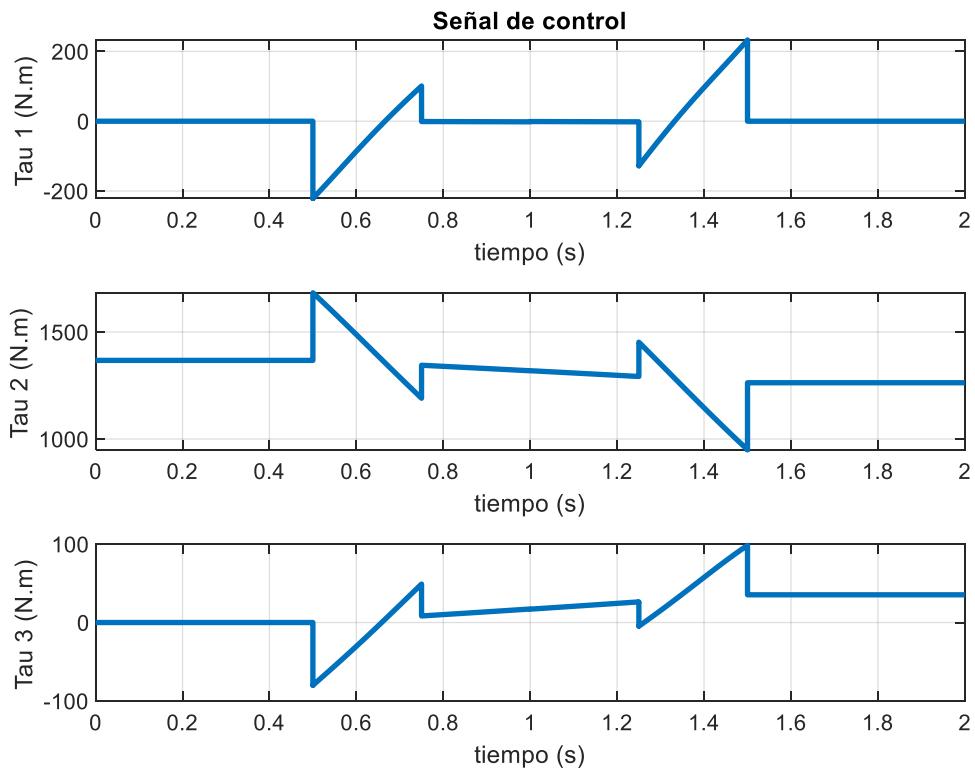
A continuación, se muestran las gráficas obtenidas usando un valor del término integral suficientemente alto, cuidando de no hacer inestable el sistema:

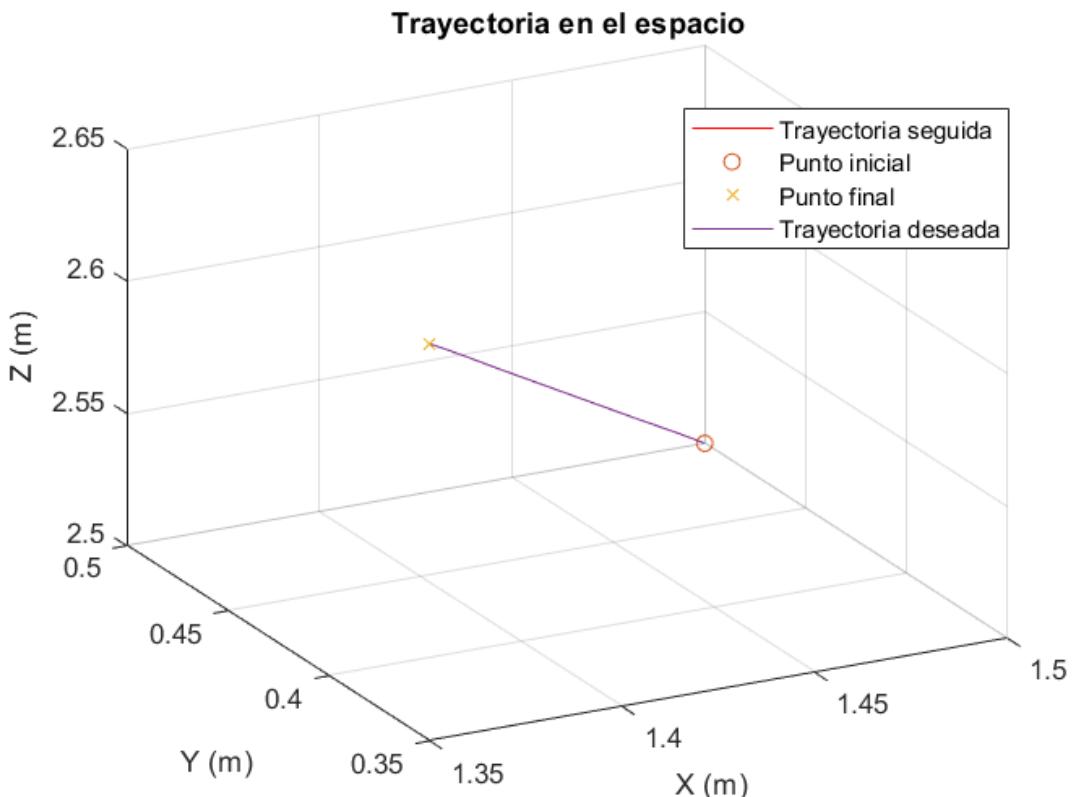






Vemos como efectivamente se consigue error en régimen permanente nulo.





Finalmente hay que decir que, al igual que se ha realizado con en el PD, podríamos hacer uso de técnicas de control que incluyan compensaciones con lo cual se mejoraría aún más el comportamiento en bucle cerrado.

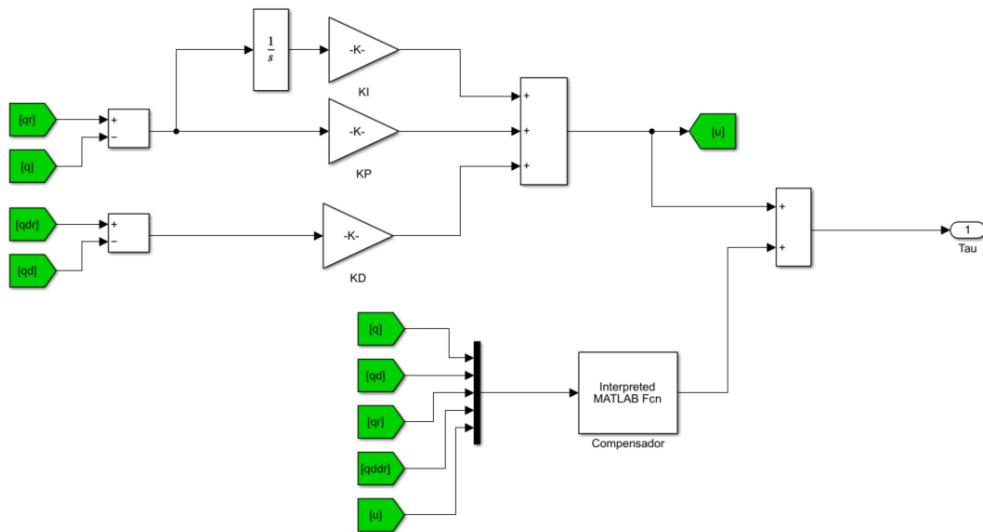
4.3. Controlador por Par Calculado

Pasemos por último al análisis de los resultados obtenidos con la técnica de control más sofisticada vista, es decir, el control por par calculado.

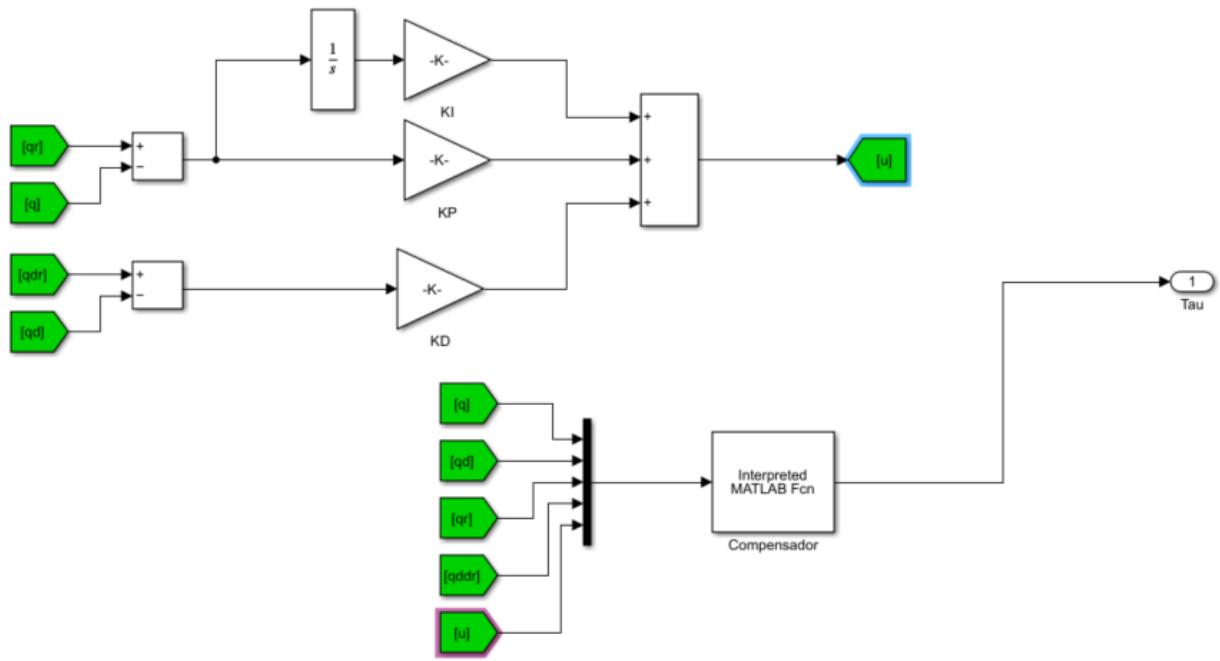
A diferencia de los anteriores es el único que no es un PD o PID al que se le suma un término de compensación. La salida del controlador no es la suma del PID y la inyección de par por parte del compensador, sino que el compensador usa la información del PID para hacer una operación dentro más compleja. Por ese motivo se ha añadido una entrada adicional al bloque compensador que precisamente es la salida del PID.

Podemos ver como se ha implementado en Simulink.

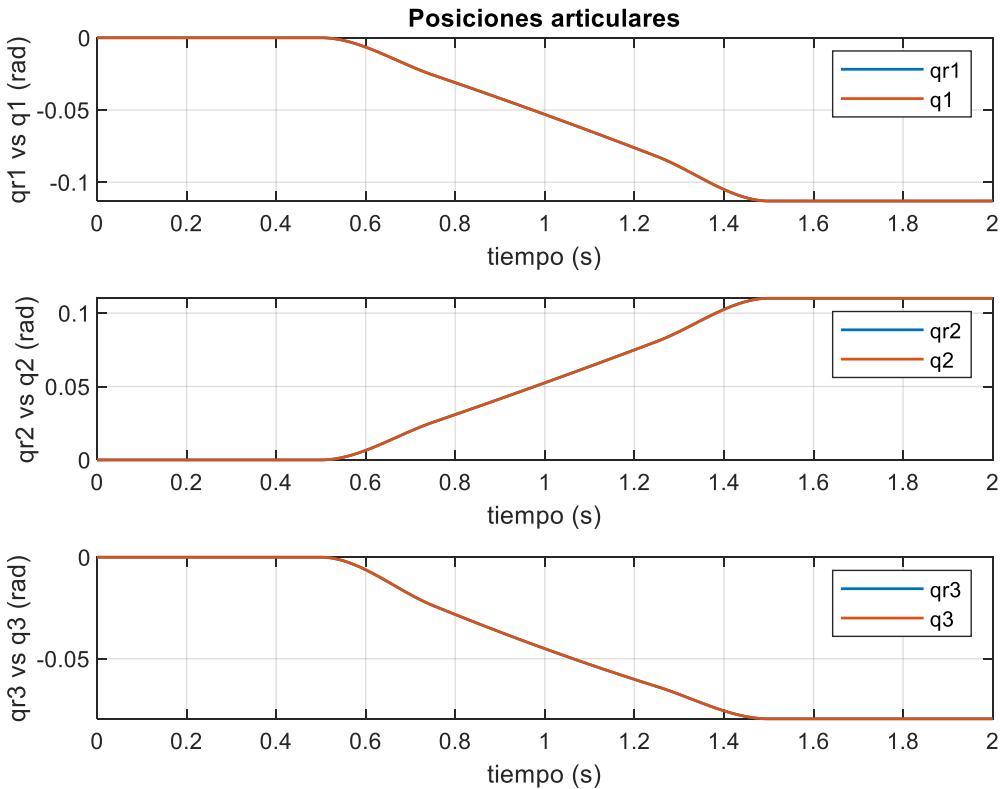
- Esquema empleado para todas las simulaciones anteriores:

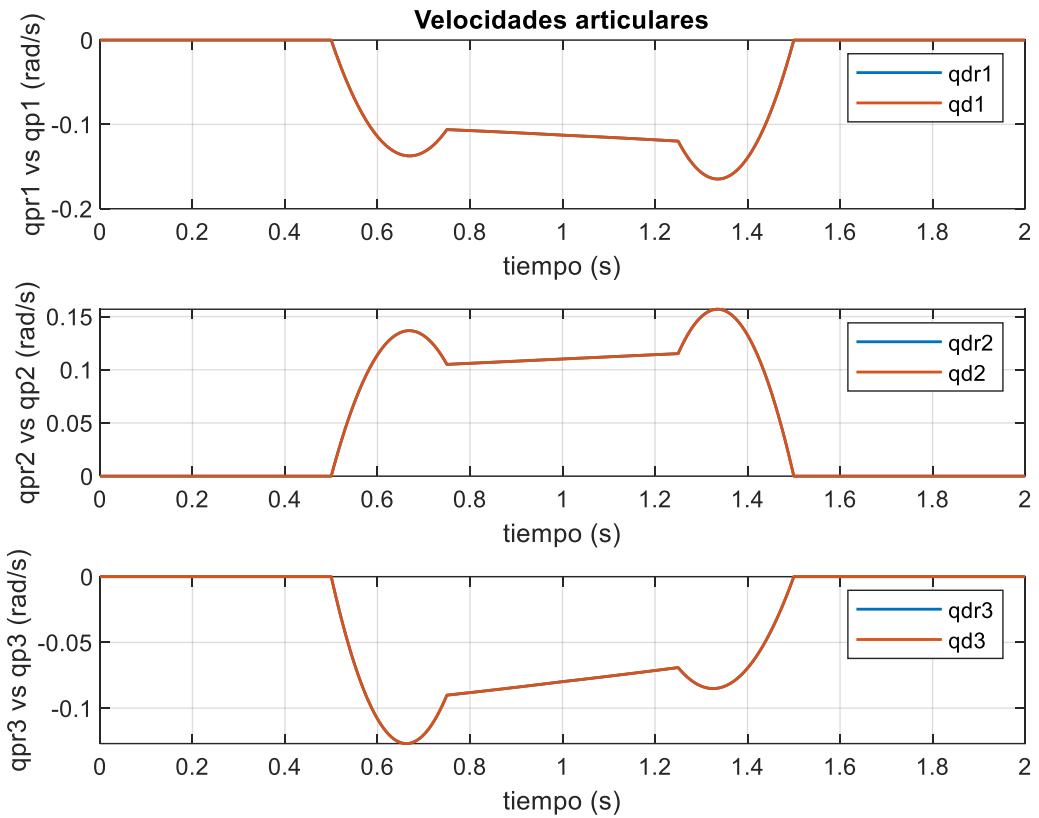


- Esquema empleado para el controlador por Par Calculado:



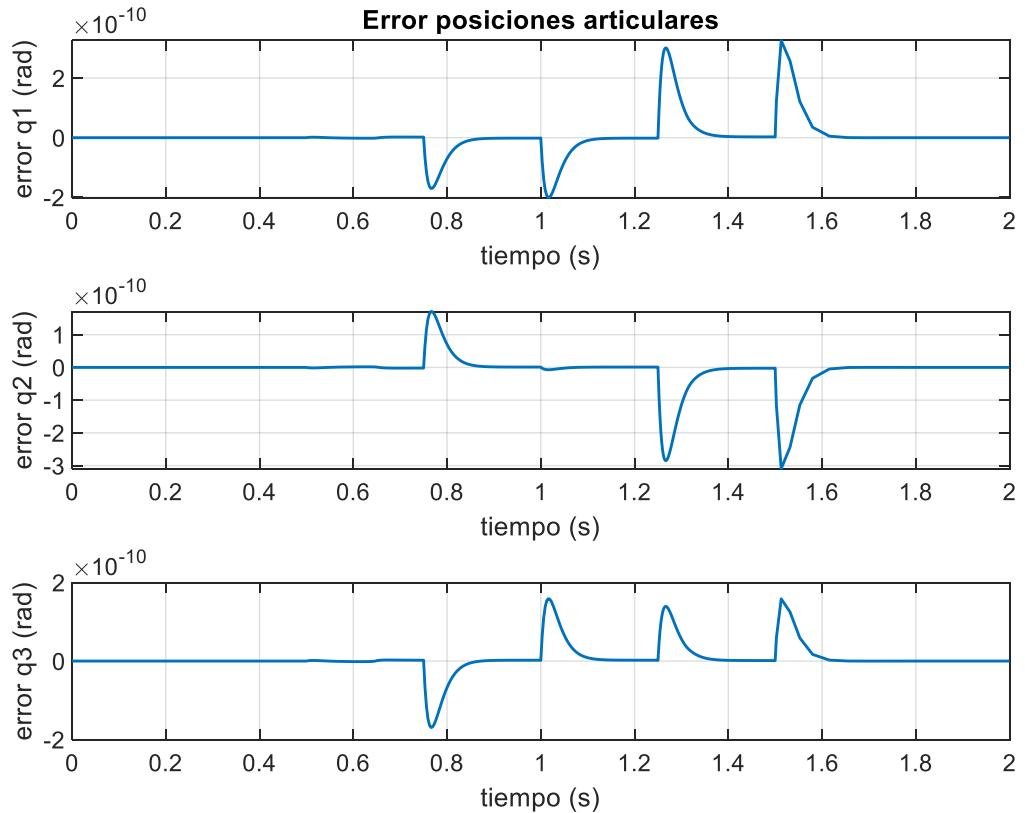
Pasemos a analizar las representaciones propias a esta última técnica que veremos:

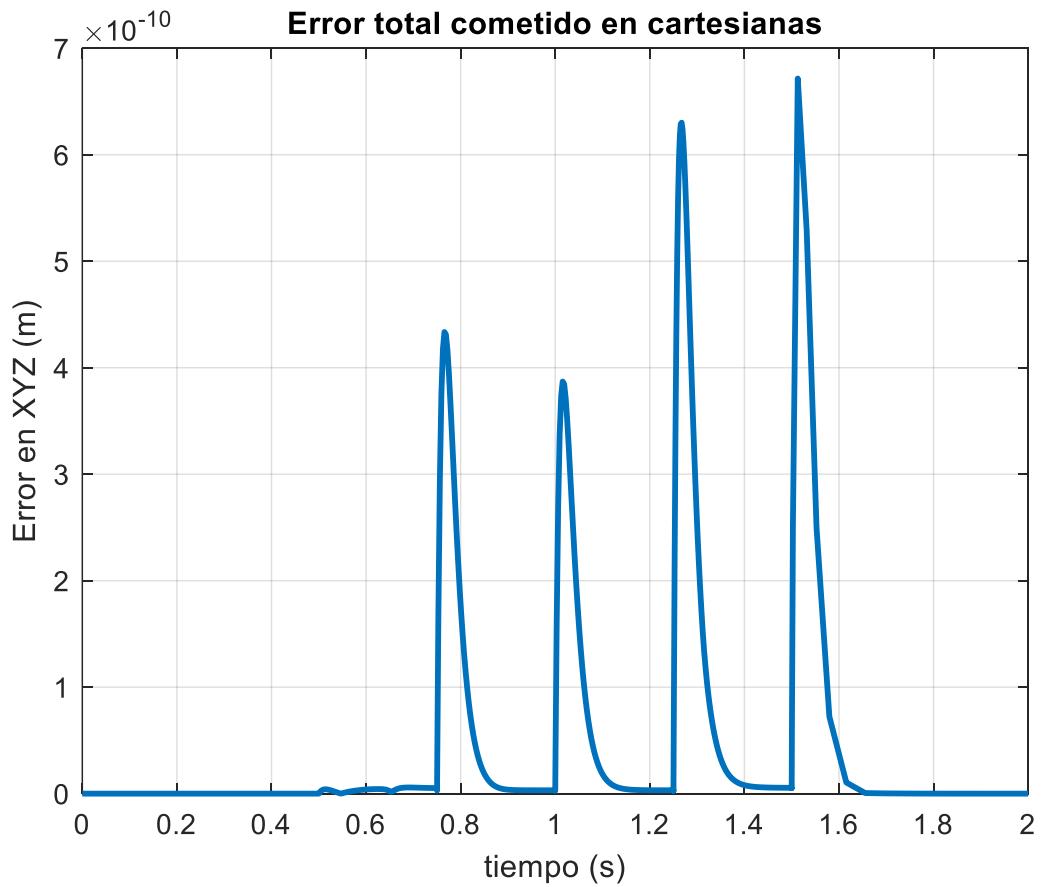
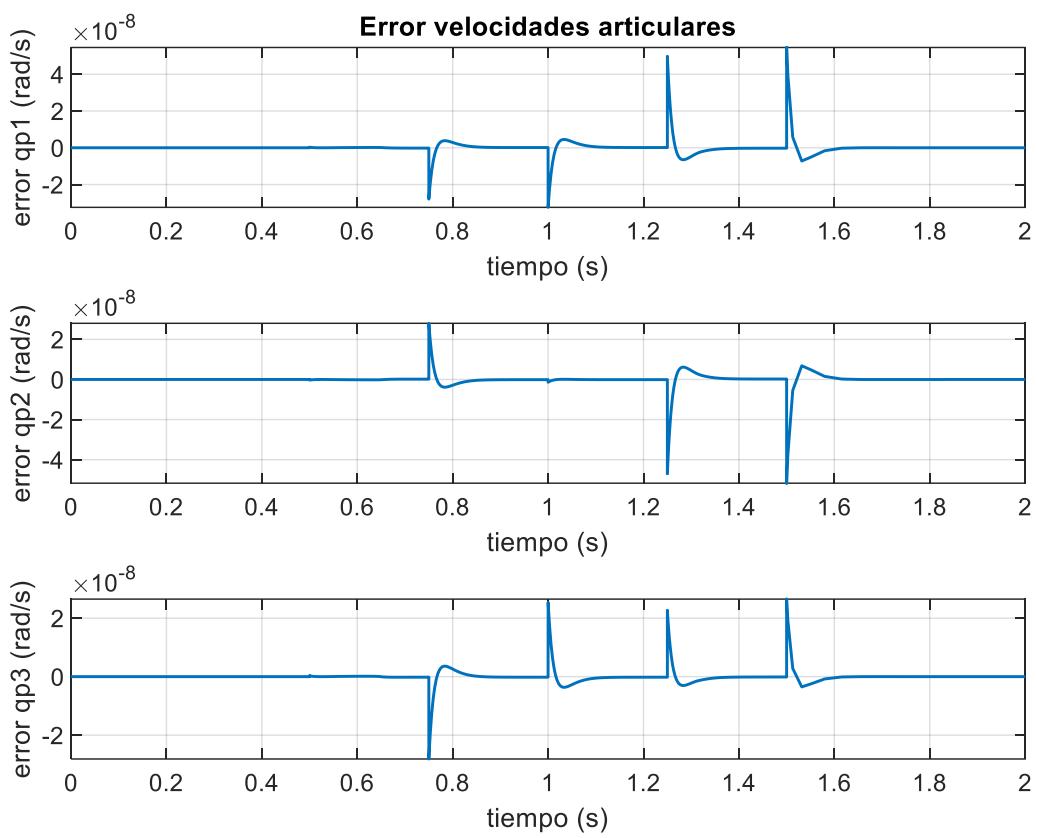


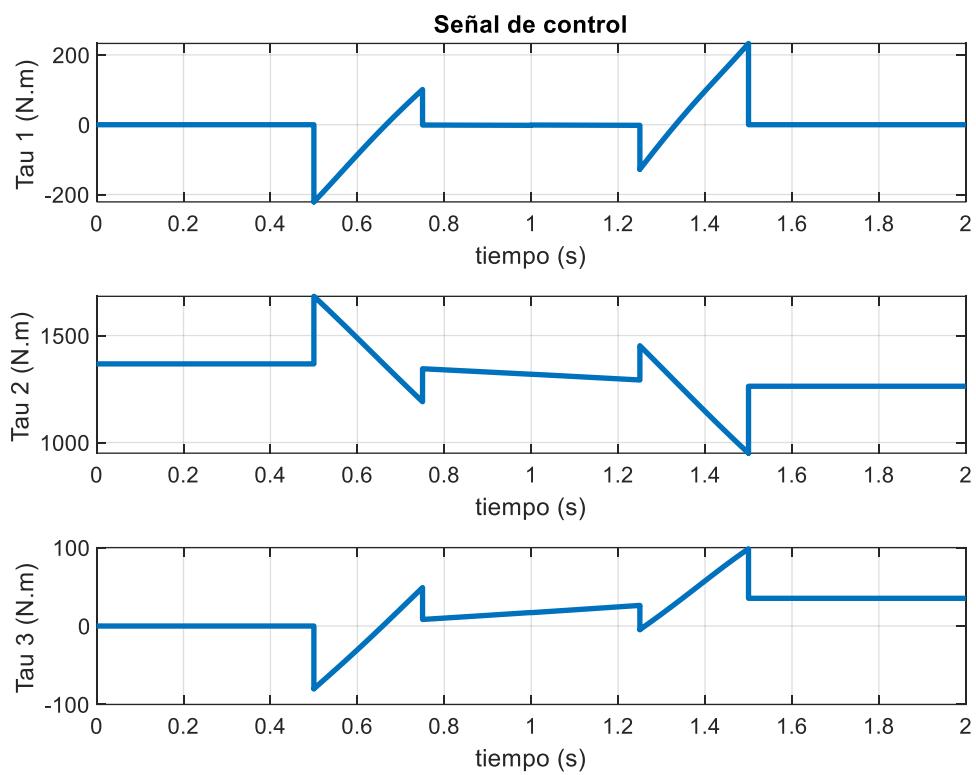


Se consigue un seguimiento de la referencia casi a la perfección, además de darse un buen rechazo a perturbaciones como se conseguía con la compensación en el PD.

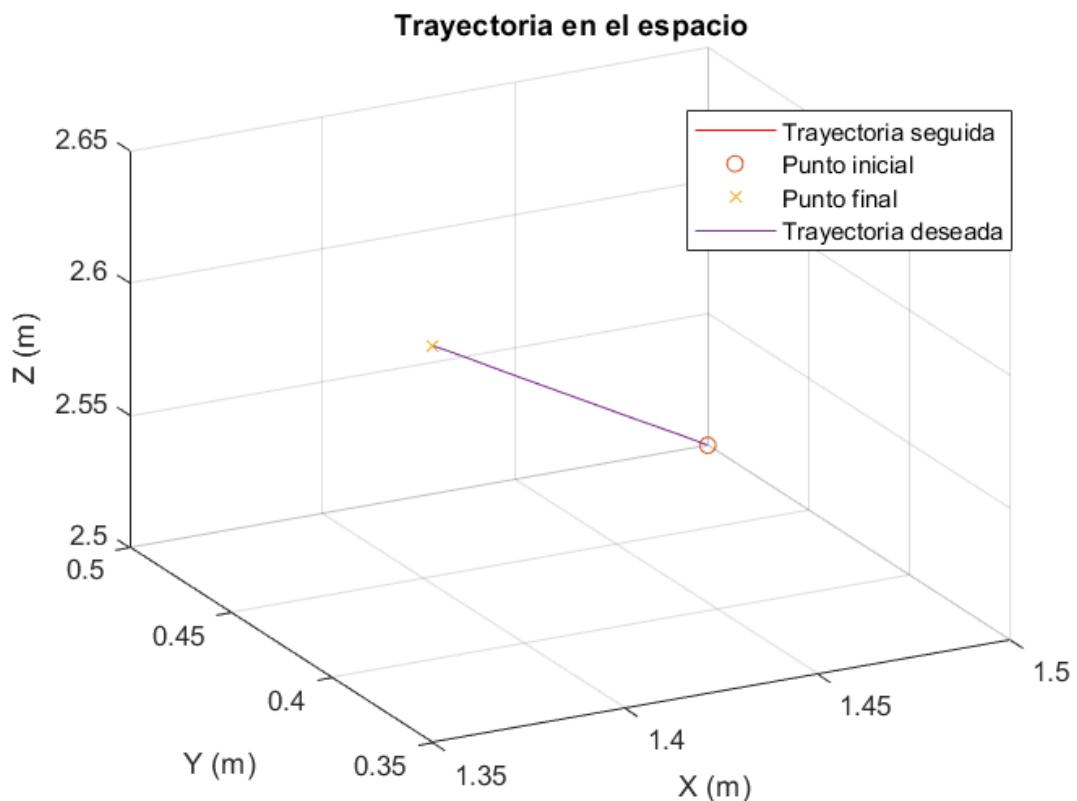
En el error cometido se puede ver la variación que existe:







Observamos transiciones abruptas en señal de control.



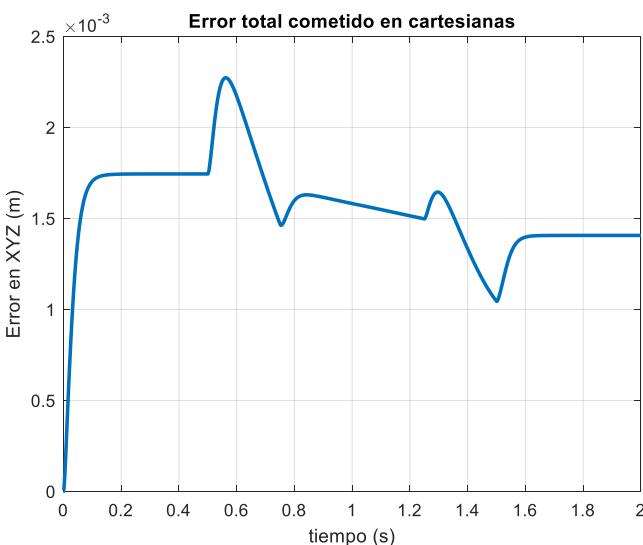
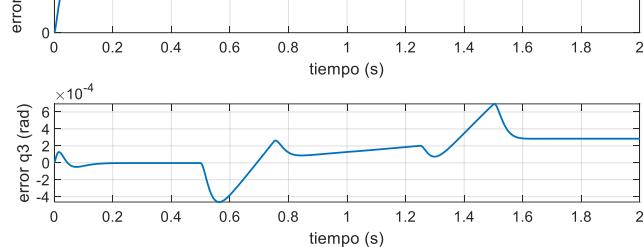
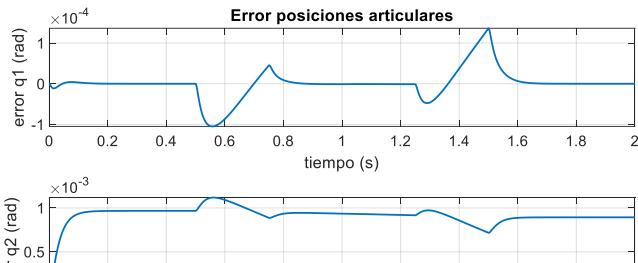
4.4. Velocidades de movimiento

En las simulaciones realizadas hasta este punto se ha tomado una duración del movimiento de 1 segundo. El objetivo de este apartado es el de ver como afecta en el control del sistema el hecho de requerir distintas velocidades para describir la trayectoria. En líneas generales, los errores obtenidos se agravan a medida que el aumenta la rapidez en el cambio de posición, sin alterar el controlador. Además, este efecto es más notorio conforme menor es la complejidad del controlador. Por tanto, en el PD se verá una mayor variación en el error cometido que un controlador con compensación de gravedad o par calculado.

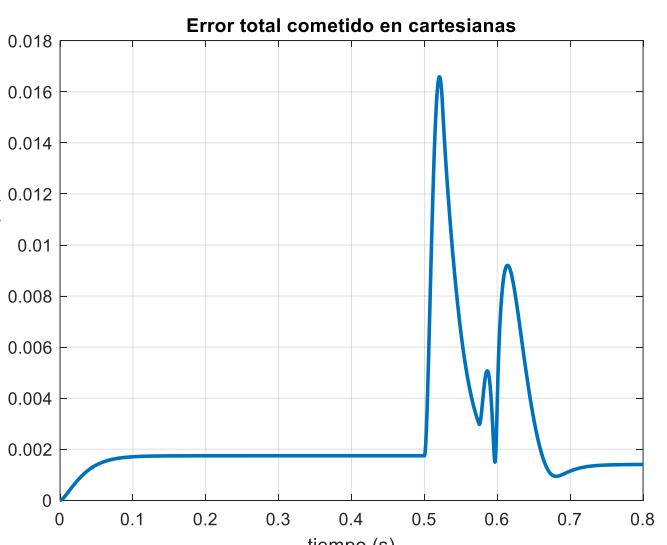
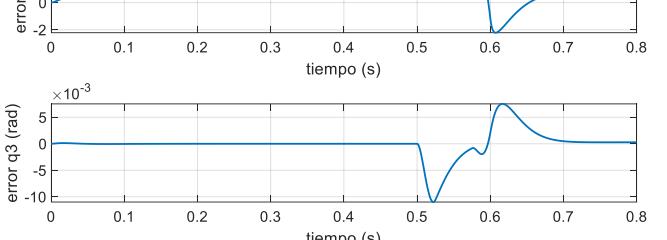
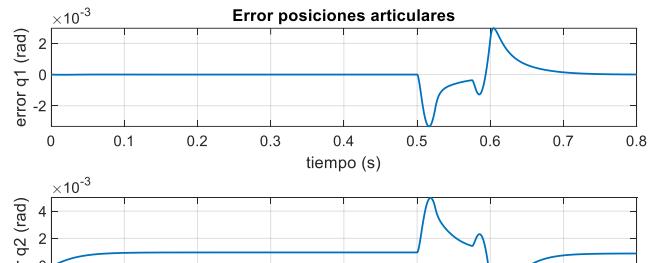
Veamos algunas comparativas para ejemplificar lo comentado.

A la izquierda podemos ver los errores y trayectoria seguida por el robot cuando usamos el PD con duración de movimiento de 1 segundo. A la derecha cuando se reduce ese tiempo a 0.1 segundos, es decir, el movimiento se realiza 10 veces más rápido.

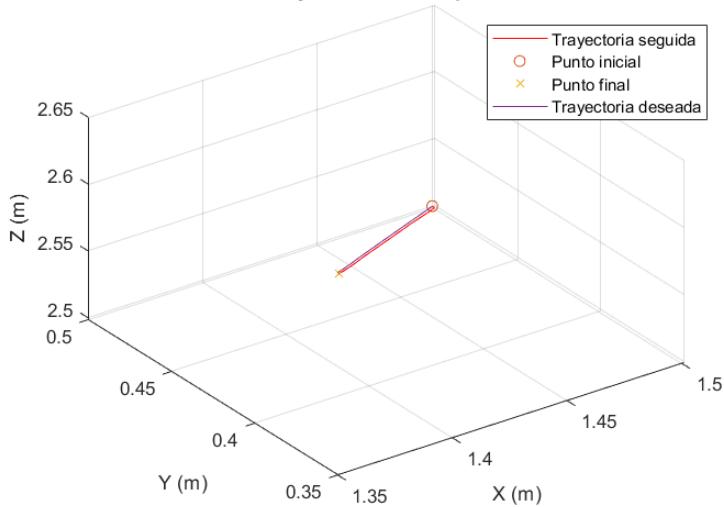
Velocidad lenta PD:



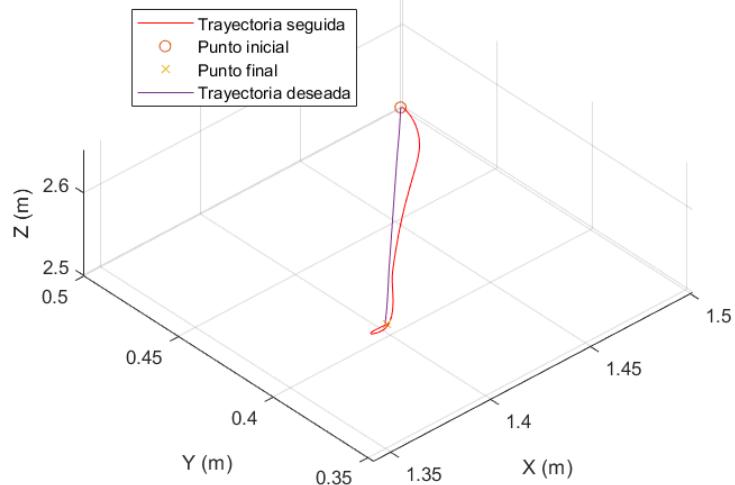
Velocidad rápida PD:



Trayectoria en el espacio



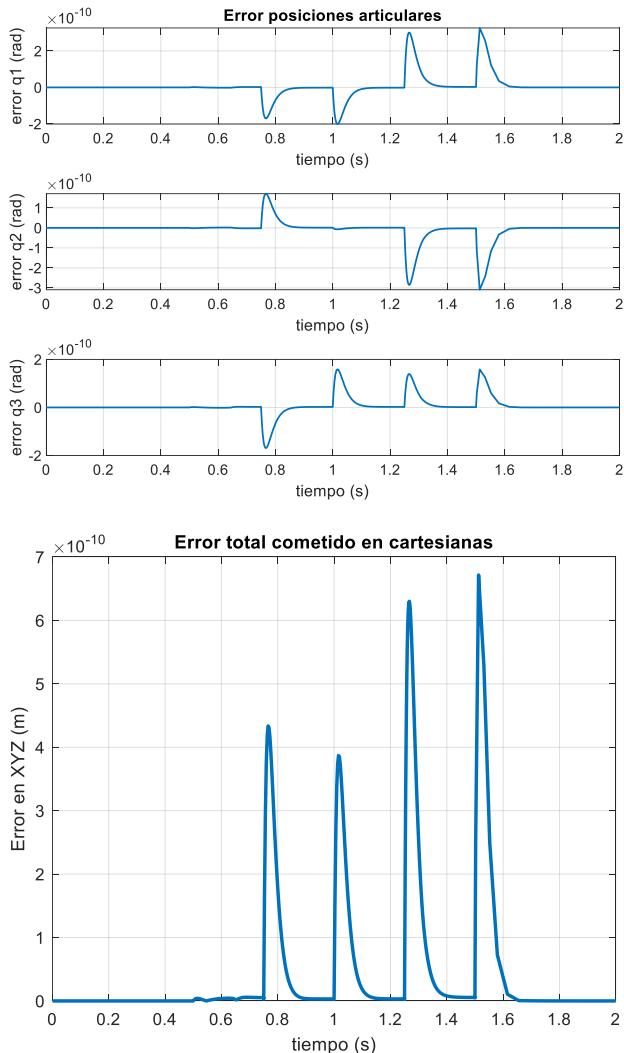
Trayectoria en el espacio



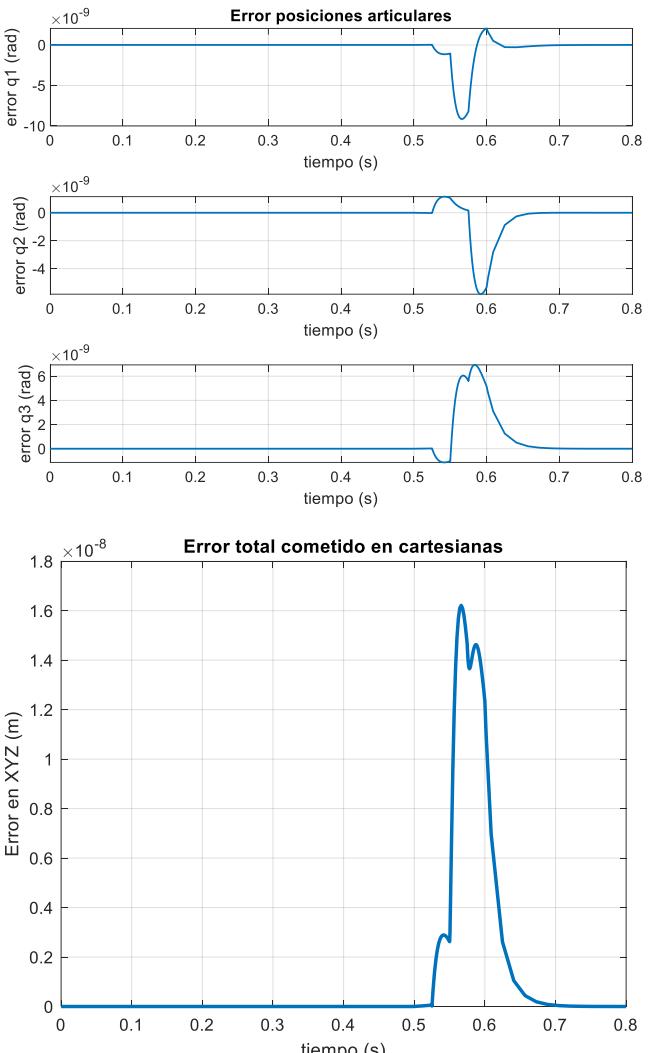
Corroboramos de esta forma los resultados comentados.

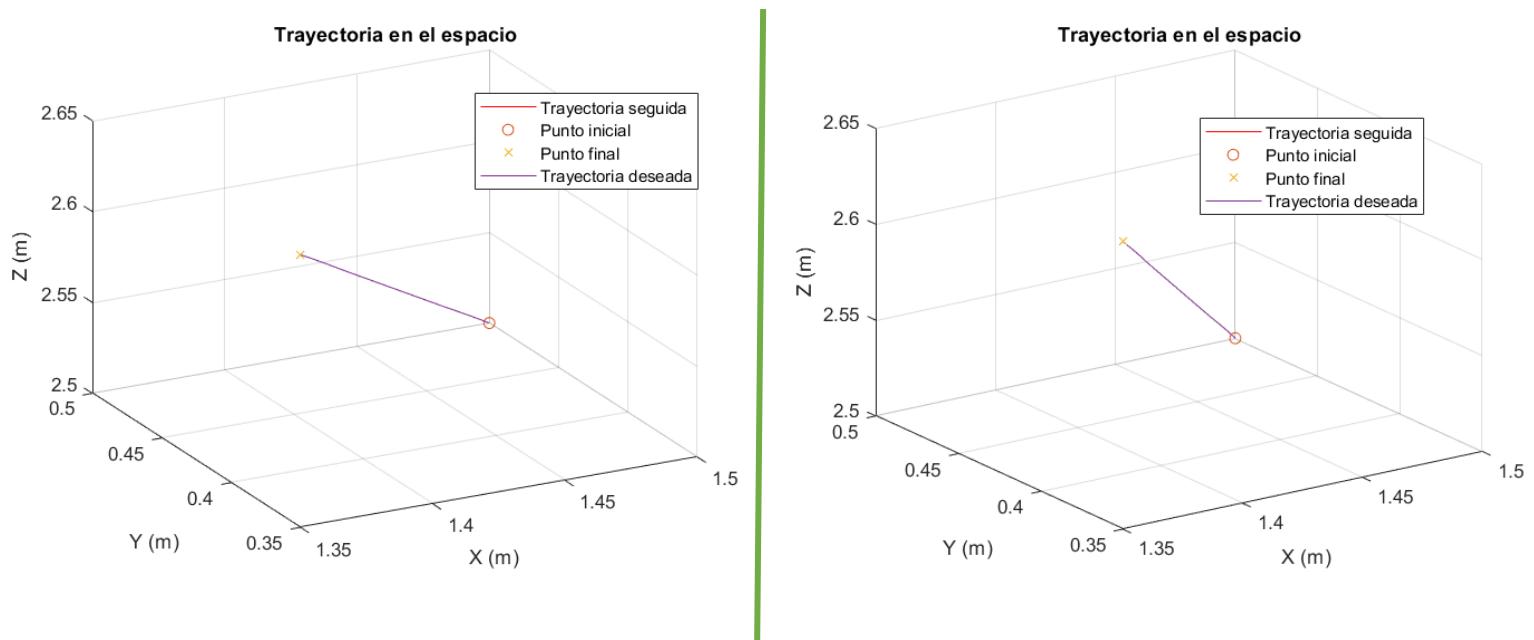
Veamos ahora un ejemplo para el caso de usar el controlador por par calculado. Nuevamente a la izquierda se representan los resultados obtenidos con 1 segundo de duración de movimiento mientras que a la derecha el equivalente a una velocidad 10 veces superior:

Velocidad lenta Par Calculado:



Velocidad rápida Par Calculado:





Podemos comprobar que el efecto es bastante menos apreciable en el caso de esta técnica de control más compleja, pero de igual forma, al aumentar la velocidad, aumenta también el error cometido.

Al igual que se ha hecho con estos controladores y velocidades de movimiento particulares se podría simular cualquier otra deseada. Se han mostrado los casos anteriores para exemplificar y demostrar experimentalmente los resultados teóricos genéricos esperados.

4.5. Conclusiones

En último lugar trataremos de resumir los principales resultados obtenidos de las simulaciones realizadas en este apartado final.

- Las prestaciones que nos ofrece el controlador más simple de todos, refiriéndonos al PD, son aceptables, sin embargo, puede mejorarse mediante la adición de un término integral, es decir, usando un PID. En dicho caso, seleccionando correctamente las constantes se puede conseguir un error nulo en el régimen permanente.
- Incorporando una inyección extra de par a la propia señal de control que aporta el PD/PID mediante técnica tales como la compensación de gravedad o precompensación de dinámica se logran reducir errores cometidos acercándonos con una mayor exactitud en el seguimiento de la trayectoria de referencia deseada.
- La técnica de control por par calculado es la más sofisticada de las vistas, la cual permite lograr un seguimiento de referencia casi perfecto en simulación.
- La velocidad con la que se realiza el movimiento, o lo que es lo mismo, la duración del cambio de posición tiene un efecto importante en los resultados obtenidos. A mayor velocidad pedida, peores resultados obtendremos. La elección de aumentar velocidad de movimiento o conseguir mayor precisión es, por tanto, dependiente de la aplicación particular, siendo en muchos casos necesario alcanzar una solución de compromiso. El efecto es más notorio conforme disminuye la complejidad del control.
- Los resultados obtenidos son bastante ideales al tratarse de simulaciones donde el suponemos que el sistema real es el propio modelo desarrollado. En la práctica real siempre existe discrepancia entre modelo y robot por lo que es de esperar resultados buenos, pero no tan ideales.