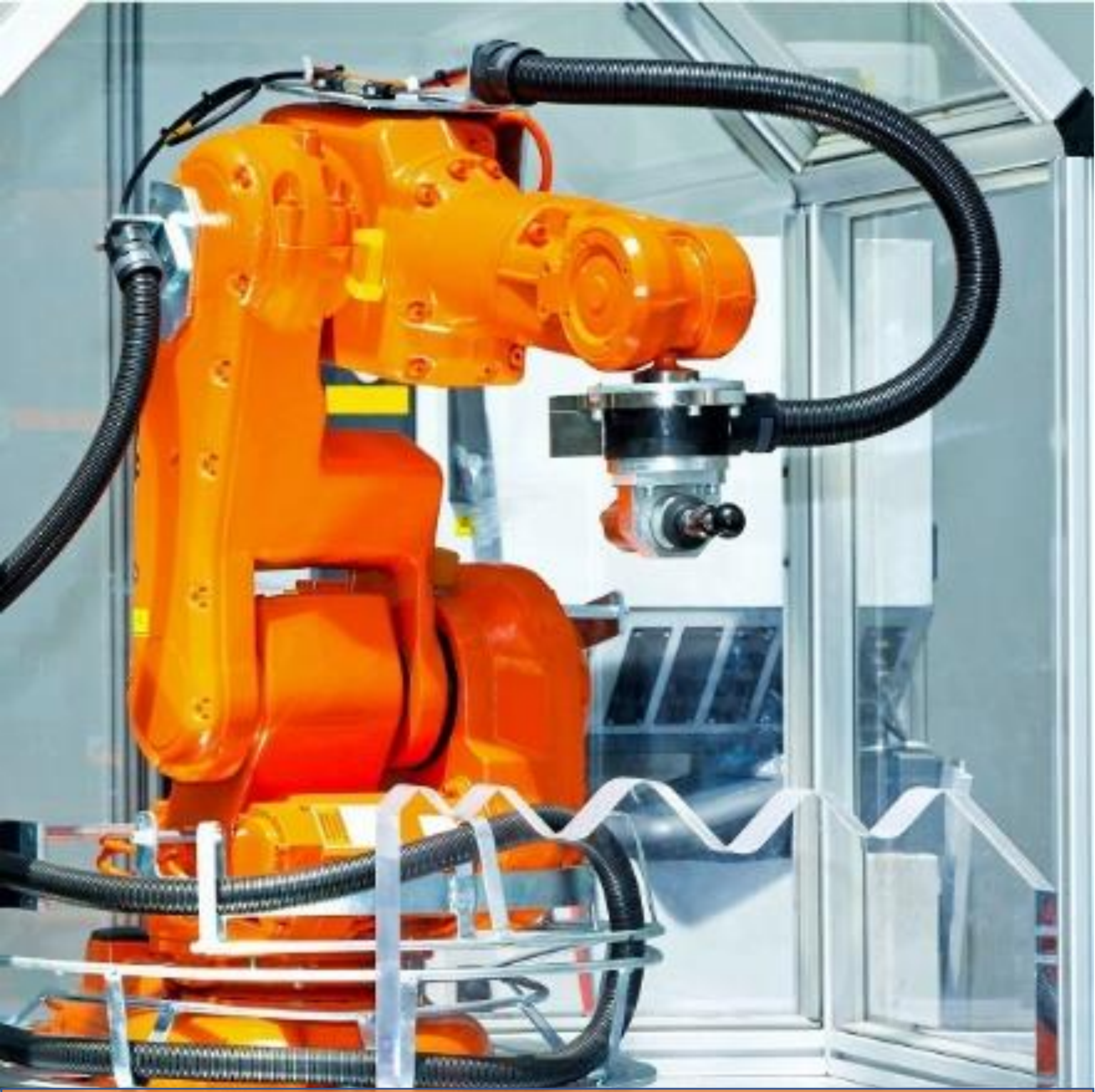


PROYECTO FUNDAMENTOS DE ROBÓTICA

Francisco Javier Román Cortés 3º GIERM

INDICE

1.- Introducción

Se plantea como objetivo del proyecto modelar y trabajar con un brazo robótico manipulador, de manera que se apliquen sobre este trabajo los conocimientos adquiridos a lo largo de la asignatura.

El orden lógico de tareas a resolver viene marcado por el índice de la página anterior, pero básicamente es resolver la cinemática del brazo robótico, tanto directa como inversa, para posteriormente, calcular y resolver también la dinámica, entre otras tareas a resolver.

Con esto, finalmente se diseñarán y aplicarán ciertos controladores para cumplir determinados requisitos impuestos al brazo robótico.

A partir de mi DNI: 54179754-B, donde D1 = 5 y D2 = 4, podemos ver que brazo robótico es el que se me asigna:

- Se designa el brazo tal que:

$\mathrm{D1} * {}^{5}/_{9} = 5 * {}^{5}/_{9} = 2.78;\ \text{Tomando la parte entera:}\ \mathrm{E1} = 2,$
me corresponde el brazo c).

- Se designa la muñeca tal que:

$\mathrm{D2} * {}^{3}/_{9} = 4 * {}^{3}/_{9} = 1.33;\ \text{Tomando la parte entera:}\ \mathrm{E2} = 1,$
me corresponde la muñeca 2.

Con esto **mi robot** en cuestión es el **C2**:

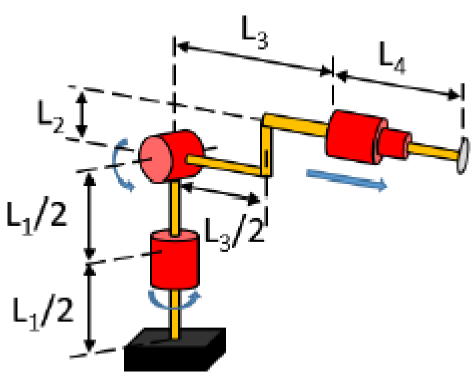


Figura 1: Brazo del Robot

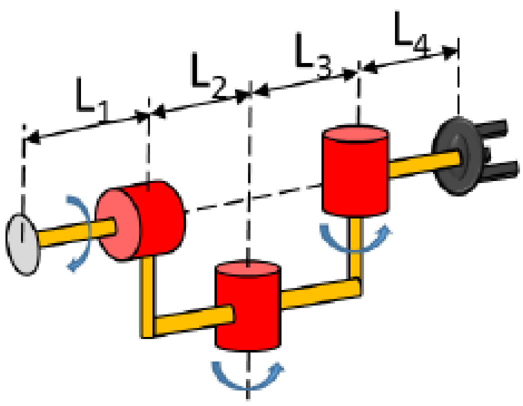


Figura 2: Muñeca del Robot

Parámetros Geométricos Brazos	L_1 (m)	L_2 (m)	L_3 (m)	L_4 (m)	L_5 (m)	L_6 (m)
c	0.8	0.4	0.6	0.4	-	-

Tabla 1: Parámetros Geométricos del brazo

Parámetros Geométricos Muñecas	L_1 (m)	L_2 (m)	L_3 (m)	L_4 (m)
2	0	0.2	0.1	0.1

Tabla 2: Parámetros Geométricos de la muñeca

2.- Análisis Cinemático

Este análisis cinemático consiste en el estudio de los movimientos del robot, pero sin tener en cuenta los efectos de las masas ni las inercias, ya que esto se estudia en el apartado del análisis dinámico. En este caso el objetivo es encontrar una serie de matrices o ecuaciones que relacionen los sistemas de referencia de las articulaciones y lo que es de mayor interés, una relación entre el sistema de referencia de la base y el sistema de referencia del efector final (herramienta).

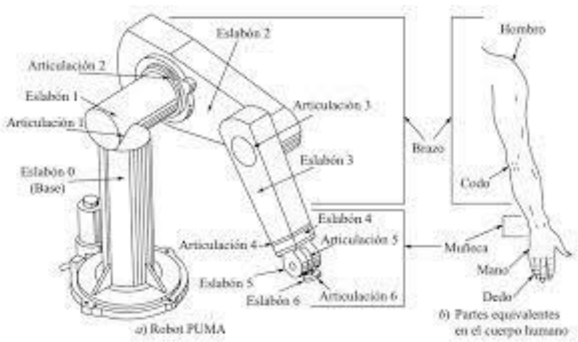


Figura 3: Símil articulaciones robot-brazo humano

Con el estudio de la cinemática, podemos relacionar los movimientos de las articulaciones entre sí, el de la herramienta o efector final con la base, etc. Es decir, podemos conocer cómo afectan los movimientos de una de las articulaciones al resto.

2.1.- Representación gráfica y obtención de parámetros de Denavit-Hartenberg

Se propone el siguiente modelo en 3D para una mejor visualización del robot en cuestión y facilitar así el procedimiento de Denavit-Hartenberg y la obtención de los parámetros de dicho procedimiento.

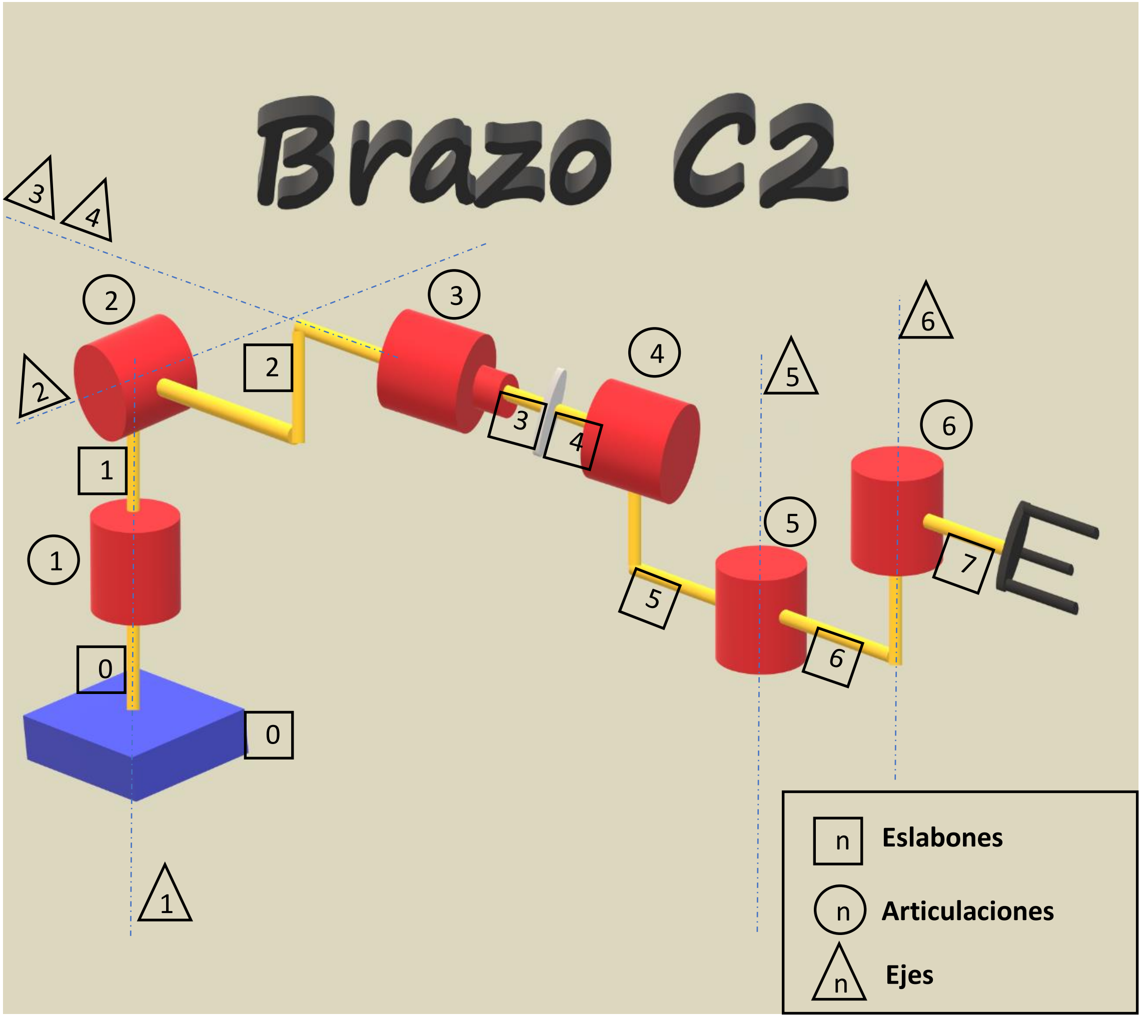


Figura 4: Modelo 3D del robot completo para mejor visualización con eslabones, articulaciones y ejes

Se ha decidido poner la numeración de eslabones, articulaciones y ejes en esta figura, para no sobrecargar de información la inferior (contiene ejes planteados para resolución)

A continuación, se adjunta la resolución sobre el modelo 3D del robot planteado anteriormente siguiendo el método de Denavit-Hartenberg, así como la tabla de parámetros resultante de dicha resolución.

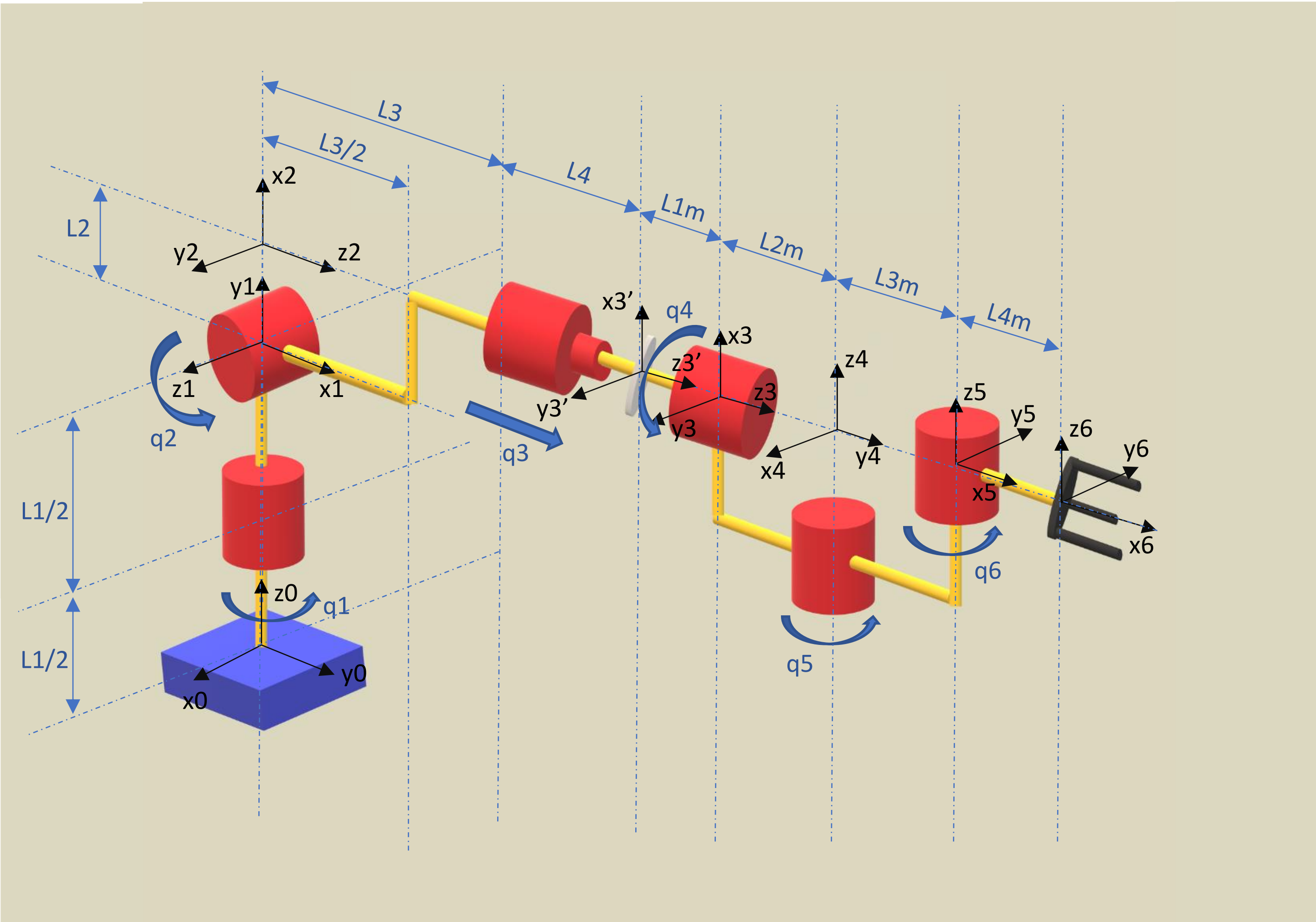


Figura 5: Modelo 3D del robot con los ejes considerados al resolver D-H (Resolución Parámetros DH)

Notar que el sistema añadido {3'} correspondería al efector final si se desacopla el sistema de la muñeca, esto es importante tenerlo en cuenta ya que, más adelante se desacopla dicho sistema de la muñeca para trabajar sólo con el brazo.

De esta figura, se ha llevado a cabo el método de Denavit-Hartenberg y por lo tanto podemos rellenar la tabla de parámetros:

Articulación	θ	d	a	α
1	q1 (+π/2)	L1	0	π/2
2	q2 (+π/2)	0	L2	π/2
3'	0	q3 (L3 + L4)	0	0
3	0	L1m	0	0
4	q4 (+π/2)	L2m	0	π/2
5	q5 (+π/2)	0	L3m	0
6	q6 (0)	0	L4m	0

Tabla 3: Parámetros de Denavit-Hartenberg del robot

2.2.- Obtención de las matrices de transformación homogéneas articulares

Para la obtención de estas MTH articulares, se ha hecho uso de la función MDH proporcionada en el software de robótica para *MATLAB* de Peter Corke. Se referencia al código de *MATLAB* utilizado para esto: Código_MTH_articulares.

- MTH de {0} a {1} (A01):

```
A01 =
[cos(q1), 0,  sin(q1),  0]
[sin(q1), 0, -cos(q1),  0]
[      0, 1,        0, L1]
[      0, 0,        0,  1]
```

- MTH de {1} a {2} (A12)

```
A12 =
[cos(q2), 0,  sin(q2), L2*cos(q2)]
[sin(q2), 0, -cos(q2), L2*sin(q2)]
[      0, 1,        0,          0]
[      0, 0,        0,          1]
```

- MTH de {2} a {3'} (A23prima) (Corresponde a la transformación desde el sistema {2} al sistema {3'}, correspondiente este último al punto de enlace entre el extremo del brazo/inicio de la muñeca.

```
A23prima =
[1, 0, 0,  0]
[0, 1, 0,  0]
[0, 0, 1, q3]
[0, 0, 0,  1]
```

- MTH de {3'} a {3} (A3prima3)

```
A3prima3 =
[1, 0, 0,   0]
[0, 1, 0,   0]
[0, 0, 1, L1m]
[0, 0, 0,   1]
```

- MTH de {3} a {4} (A34)

```
A34 =
[cos(q4), 0,  sin(q4),   0]
[sin(q4), 0, -cos(q4),   0]
[      0, 1,        0, L2m]
[      0, 0,        0,   1]
```

- MTH de {4} a {5} (A45)

```
A45 =
[cos(q5), -sin(q5), 0, L3m*cos(q5)]
[sin(q5),  cos(q5), 0, L3m*sin(q5)]
[       0,        0, 1,          0]
[       0,        0, 0,          1]
```

- MTH de {5} a {6} (A56)

```
A56 =
[cos(q6), -sin(q6), 0, L4m*cos(q6)]
[sin(q6),  cos(q6), 0, L4m*sin(q6)]
[       0,        0, 1,          0]
[       0,        0, 0,          1]
```

2.3.- Obtención del modelo cinemático directo (ecuaciones simbólicas)

A partir de este apartado, para el desarrollo del proyecto sólo se tienen en cuenta las tres primeras articulaciones del robot, es decir, se desacopla el sistema de la muñeca.

Para hallar la MTH que relaciona la base {0} con el extremo del brazo (con la muñeca desacoplada), {3'} se realiza la multiplicación: A01*A12*A23prima:

```
>> %MTH base(0)->desacople_muñeca(3')(brazo)
T03prima = simplify(A01*A12*A23prima)
T03prima =
[cos(q1)*cos(q2),  sin(q1), cos(q1)*sin(q2), cos(q1)*(L2*cos(q2) + q3*sin(q2))]
[cos(q2)*sin(q1), -cos(q1), sin(q1)*sin(q2), sin(q1)*(L2*cos(q2) + q3*sin(q2))]
[        sin(q2),        0,        -cos(q2),      L1 + L2*sin(q2) - q3*cos(q2)]
[              0,        0,               0,                                 1]
```

Se puede identificar la última columna como la información que detalla la posición del sistema 3', es decir del extremo del robot sin muñeca (sería el "nuevo efector final"):

$$\begin{cases} x = \cos(q1) * (L2 * \cos(q2) + q3 * \sin(q2)) \\ y = \sin(q1) * (L2 * \cos(q2) + q3 * \sin(q2)) \\ z = L1 + L2 * \sin(q2) - q3 * \cos(q2) \end{cases}$$

Se puede reescribir la MTH anterior con la notación reducida tal que:

$$T03' = \begin{bmatrix} C_1C_2 & S_1 & C_1S_2 & C_1(L_2C_2 + q_3S_2) \\ C_2S_1 & -C_1 & S_1S_2 & S_1(L_2C_2 + q_3S_2) \\ S_2 & 0 & -C_2 & L_1 + L_2S_2 - q_3C_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Se anexa el código utilizado para obtener estas figuras [Representacion_Toolbox]:

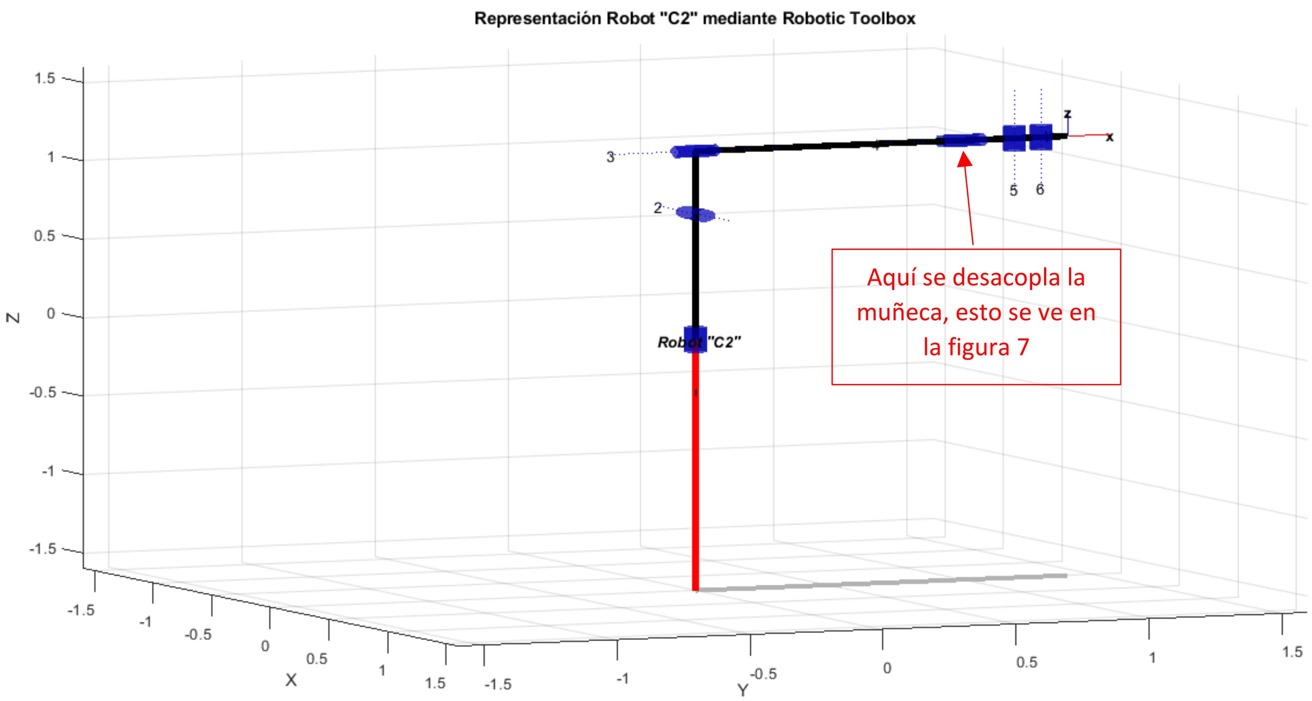


Figura 6: Representación del robot con sus 6 gdl mediante Robotic Toolbox en su "postura de dibujo"

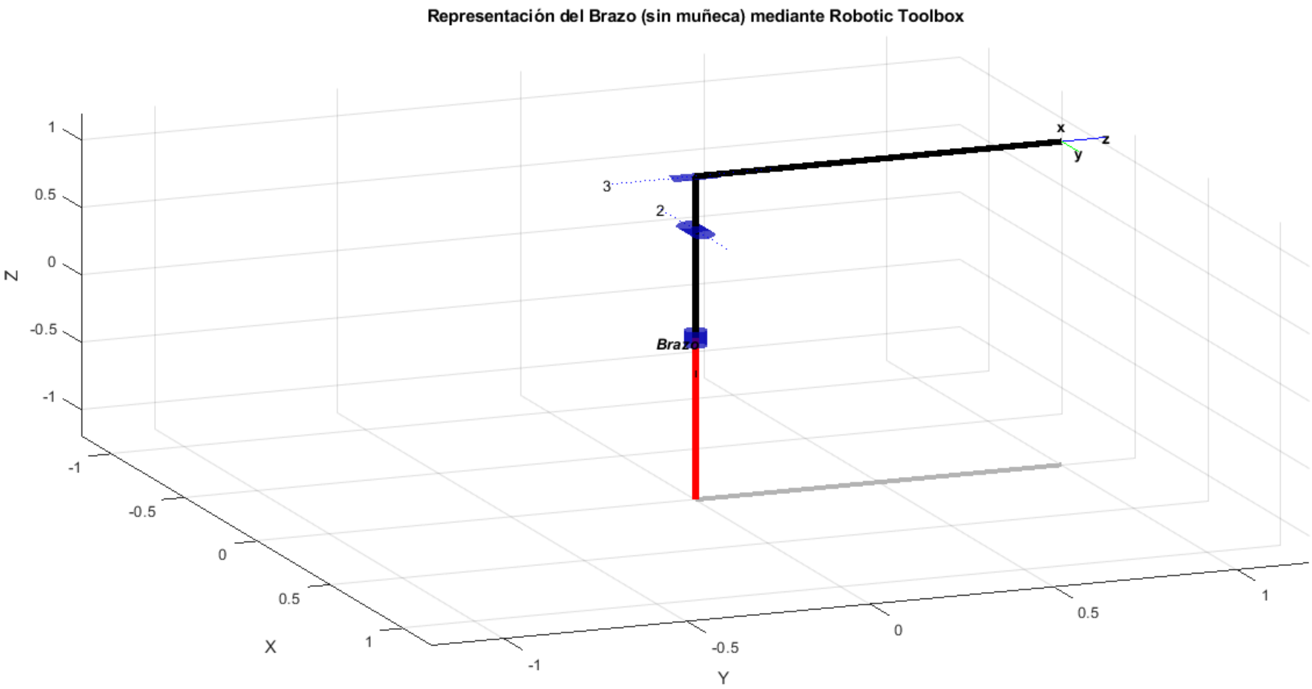


Figura 7: Representación del brazo (sin muñeca) con sus 3 gdl mediante Robotic Toolbox en su "postura de dibujo"

Comprobación del Modelo Cinemático Directo:

Es interesante, tras obtener el modelo cinemático directo para el robot de 3 gdl, comprobar su corrección mediante la verificación de ciertas posiciones, por ejemplo, probamos 3 posiciones sencillas de verificar. Para las 3 posiciones se va a plantear como quedaría el modelo del robot, donde se identificará como debería salir esta matriz "T03prima" para estos valores de q1, q2, q3 y se comprobará si queda igual con la que se ha obtenido en MATLAB.

- **"Posición de dibujo (HOME)": q1 = π/2; q2 = π/2; q3 = L3+L4**

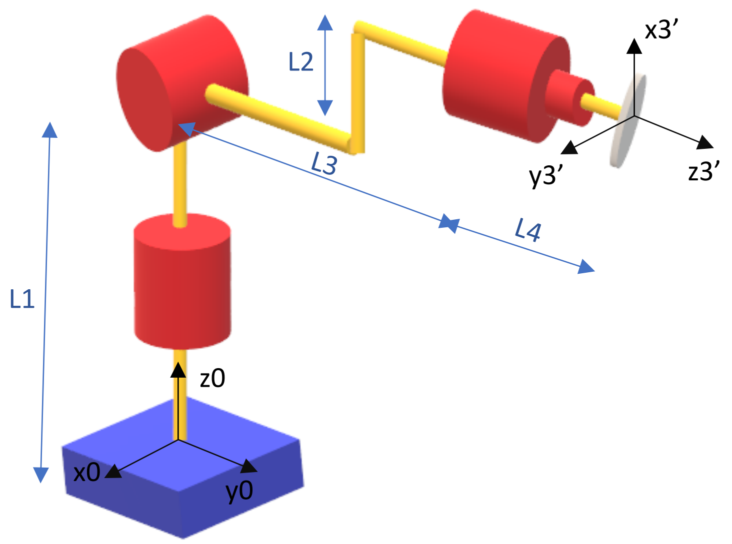


Figura 8: Representación del brazo en "posición de dibujo"

Se puede ver que viendo el sistema {3'} respecto del {0}, está desplazado una distancia L1+L2 respecto del eje z0 y una distancia L3+L4 respecto del eje y0, con lo cual:

Cuarta columna de T03prima = vector [0 (L3+L4) (L1+L2)]'

junto a la orientación que tenemos:

x3' sigue la dirección de z0 -> primera columna de T03prima = vector [0 0 1]'

y3' sigue la dirección de x0 -> segunda columna de T03prima = vector [1 0 0]'

z3' sigue la dirección de y0 -> tercera columna de T03prima = vector [0 1 0]'

Se añade una cuarta fila: [0 0 0 1], para tener la matriz 4x4 deseada.

Evaluando esto sobre la T03prima obtenida, se comprueba que ocurre lo esperado:

```
>> q1 = pi1/2;q2 = pi1/2;q3 = L3+L4;
>> eval(T03prima)
ans =
[0, 1, 0,       0]
[0, 0, 1, L3 + L4]
[1, 0, 0, L1 + L2]
[0, 0, 0,       1]
```

Al hacer estas comprobaciones, debemos tener en cuenta ciertos aspectos, como que por la estructura física del robot q3 debe estar en el siguiente rango: [L3/2; L3+L4], así como los valores de q1 y q2 que provocasen colisiones entre eslabones.

- **"Posición hacia arriba": q1 = π/2; q2 = π; q3 = L3+L4**

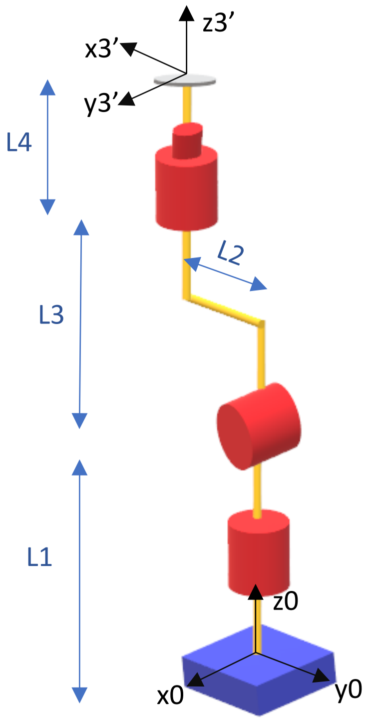


Figura 9: Representación del brazo en "posición hacia arriba"

Se puede ver que viendo el sistema {3'} respecto del {0}, está desplazado una distancia L1+L3+L4 respecto del eje z0 y una distancia -L2 respecto del eje y0, con lo cual:

Cuarta columna de T03prima = vector [0 (-L2) (L1+L3+L4)]'

junto a la orientación que tenemos:

x3' sigue la dirección de – y0 -> primera columna de T03prima = vector [0 -1 0]'

y3' sigue la dirección de x0 -> segunda columna de T03prima = vector [1 0 0]'

z3' sigue la dirección de z0 -> tercera columna de T03prima = vector [0 0 1]'

Se añade una cuarta fila: [0 0 0 1], para tener la matriz 4x4 deseada.

Evaluando esto sobre la T03prima obtenida, se comprueba que ocurre lo esperado:

```
>> q1 = pi1/2;q2 = pi1;q3 = L3+L4;
>> eval(T03prima)
ans =
[ 0, 1, 0,            0]
[-1, 0, 0,          -L2]
[ 0, 0, 1, L1 + L3 + L4]
[ 0, 0, 0,            1]
```

- **“Posición hacia dentro”: q1 = π; q2 = π/2; q3 = L3+L4**

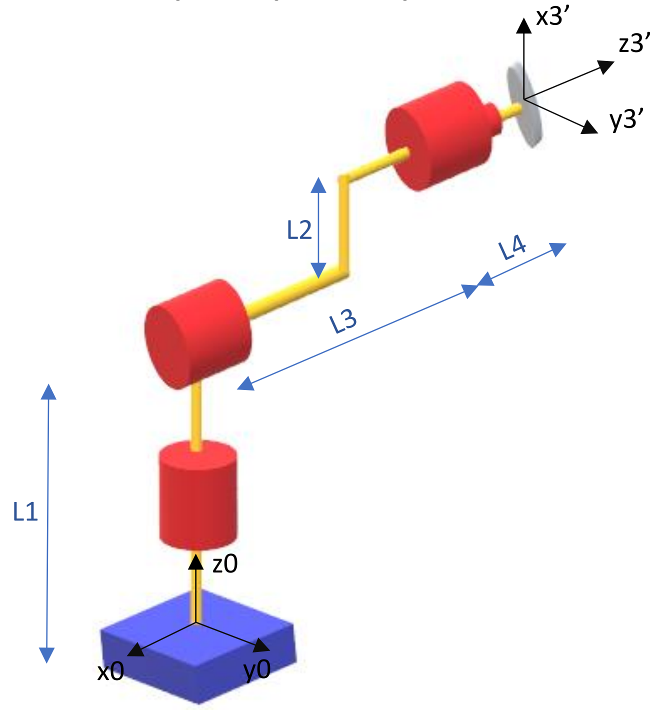


Figura 10: Representación del brazo en “posición hacia adentro”

Se puede ver que viendo el sistema {3’} respecto del {0}, está desplazado una distancia L1+L2 respecto del eje z0 y una distancia –(L3+L4) respecto del eje x0, con lo cual:

Cuarta columna de T03prima = vector [(-L3-L4) 0 (L1+L2)]’

junto a la orientación que tenemos:

x3’ sigue la dirección de z0 -> primera columna de T03prima = vector [0 0 1]’

y3’ sigue la dirección de y0 -> segunda columna de T03prima = vector [0 1 0]’

z3’ sigue la dirección de – x0 -> tercera columna de T03prima = vector [-1 0 0]’

Se añade una cuarta fila: [0 0 0 1], para tener la matriz 4x4 deseada.

Evaluando esto sobre la T03prima obtenida, se comprueba que ocurre lo esperado:

```
>> q1 = pi1;q2 = pi1/2;q3 = L3+L4;
>> eval(T03prima)
ans =
[0, 0, -1, - L3 - L4]
[0, 1,  0,         0]
[1, 0,  0,   L1 + L2]
[0, 0,  0,         1]
```

2.4.- Obtención del modelo cinemático inverso (expresiones analíticas)

Sabemos que existe un método para resolver el problema cinemático inverso a partir de las MTH. Hay que tener en cuenta que, en el problema inverso, los datos que tenemos son $p_x, p_y\ y\ p_z$, la posición del efector final {3'} y se busca determinar las incógnitas articulares: $q_1, q_2\ y\ q_3$. Para ello se siguen los pasos necesarios:

1.- Obtener el MCD (Modelo Cinemático Directo):

Se ha hecho justo en el apartado anterior.

2.- Aislar q1:

$$(\mathbf{A01})^{-1} * \mathbf{T03prima} = \mathbf{A12} * \mathbf{A23prima}$$

```
[cos(q1),   sin(q1), 0,    0]        [nx, ox, ax, px]
[       0,        0, 1, -L1]    *    [ny, oy, ay, py]    =
[sin(q1), -cos(q1), 0,    0]        [nz, oz, az, pz]
[       0,        0, 0,    1]        [ 0,  0,  0,  1]
```

```
    [cos(q2), 0,  sin(q2), L2*cos(q2) + q3*sin(q2)]
=   [sin(q2), 0, -cos(q2), L2*sin(q2) - q3*cos(q2)]
    [      0, 1,        0,                       0]
    [      0, 0,        0,                       1]
```

Despejando de la ecuación señalada:

$$\mathrm{p_x S_1 - p_y C_1 = 0}\ ;\ \mathrm{p_x} S_1 = \mathrm{p_y} C_1\ ;\ \frac{\mathrm{p_y}}{\mathrm{p_x}} = \tan(\mathrm{q_1})\ ;\ \boxed{\mathbf{q_1 = atan2\,(p_y, p_x)}}$$

3.- Aislar ahora q2, además podemos sacar también q3 (depende de q1 y q2):

Se ha acudido para este apartado a las siguientes Ecuaciones_prototipo:

$$(\mathbf{A12})^{-1} * (\mathbf{A01})^{-1} * \mathbf{T03prima} = \mathbf{A23prima}$$

```
[cos(q1)*cos(q2), cos(q2)*sin(q1),  sin(q2), - L2 - L1*sin(q2)]        [nx, ox, ax, px]
[        sin(q1),        -cos(q1),        0,                 0]    *   [ny, oy, ay, py]    =
[cos(q1)*sin(q2), sin(q1)*sin(q2), -cos(q2),        L1*cos(q2)]        [nz, oz, az, pz]
[              0,               0,        0,                 1]        [ 0,  0,  0,  1]
```

```
    [1, 0, 0,  0]
=   [0, 1, 0,  0]
    [0, 0, 1, q3]
    [0, 0, 0,  1]
```

En primer lugar, si nos fijamos en la ecuación recuadrada en rojo:

$$\mathrm{p_x(C_1C_2) + p_y(C_2S_1) + p_zS_2 - L_2 - L_1S_2 = 0}\;;$$

$$\mathrm{S_2}\underbrace{(\mathrm{p_z - L_1})}_{b} + \mathrm{C_2}\underbrace{(\mathrm{p_xC_1 + p_yS_1})}_{a} = \underbrace{\mathrm{L_2}}_{c};$$

Identificando términos con la ecuación (A.5)

$$\boxed{\mathbf{q_2 = atan\,2\left(L_2, \pm\sqrt{a^2 + b^2 - c^2}\right) - atan2(a, b)}}$$

Siendo *"a, b y c"* los términos señalados anteriormente, que como vemos depende de q_1, ya calculado previamente.

Finalmente, si nos fijamos en la ecuación recuadrada en verde queda:

$$\mathrm{p_x(C_1S_2) + p_y(S_1S_2) - p_zC_2 + L_1C_2 = q_3}\;;$$

En este caso, no es necesario acudir a ecuaciones prototipo, sino despejar q_3:

$$\boxed{\mathbf{q_3 = S_2(p_yS_1 + p_xC_1) + C_2(L_1 - p_z)}}$$

Ahora q_3, depende tanto de q_1 como de q_2, estando ambas ya resueltas.

Con esto ya se tendrían determinadas q_1, q_2 y q_3 y por tanto está resuelto el problema cinemático inverso. Se anexa: Codigo_CInv para trabajar con la Cinemática Inversa.

Comprobación del Modelo Cinemático Inverso:

Para realizar comprobaciones que verifiquen en cierto modo si estas ecuaciones obtenidas son correctas se podría realizar el siguiente procedimiento:

1) A partir del MCD, escogiendo valores de q1, q2, q3, obtener la matriz T03prima y fijarnos en la última columna (posición).
2) Metiendo esa posición (px, py, pz) en las ecuaciones de la cinemática inversa, se consiguen ciertos valores de q1, q2, q3, habiendo 2 ramas de soluciones para estas 3 Qs.
3) Se evalúa la matriz T03prima con los valores de q1, q2, q3 obtenidos de las 2 ramas de soluciones de la Cinemática Inversa y se comprueba si obtenemos la misma posición (cuarta columna) que en 1). Para la submatriz de rotación, se tendrán varias diferentes que no tienen por qué coincidir con la original del MCD,

al representar esas 2 soluciones tendremos varias posiciones que puede tomar el robot para alcanzar esa posición en su extremo {3'}.

- **"Posición de dibujo": q1 = π/2; q2 = π/2; q3 = L3+L4**

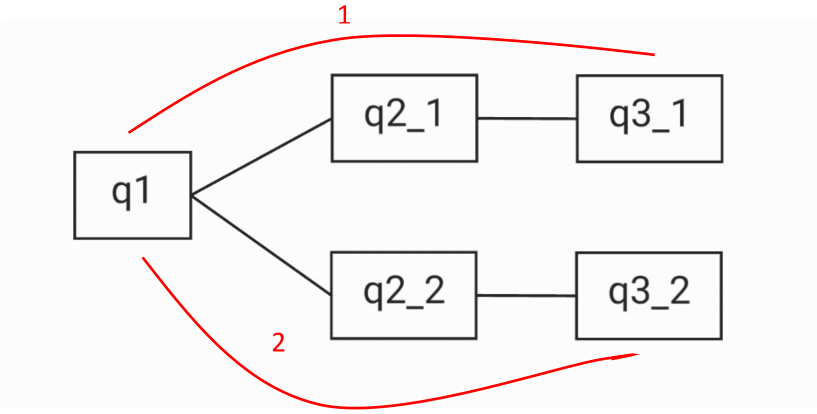
1)

```
>> q1 = pi1/2;q2 = pi1/2;q3 = L3+L4;
>> eval(T03prima)
ans =
[0, 1, 0,       0]
[0, 0, 1, L3 + L4]
[1, 0, 0, L1 + L2]
[0, 0, 0,       1]
```

```
>> L1 = 0.8; L2 = 0.4; L3 = 0.6; L4 = 0.4;
>> q1 = pi1/2;q2 = pi1/2;q3 = L3+L4;
>> eval(T03prima)
ans =
[0, 1, 0,   0]
[0, 0, 1,   1]
[1, 0, 0, 6/5]
[0, 0, 0,   1]
```

Posición que se espera numéricamente para la "Posición de dibujo"

2)




```
>> q1,q2_1,q2_2,q3_1,q3_2
q1 =
        1.5708
q2_1 =
       -0.80978
q2_2 =
        1.5708
q3_1 =
      -1
q3_2 =
       1
```

3) Se evalúan las 2 ramas de Qs obtenidas metiéndolas en el MCD:

1

```
>> q1=q1; q2=q2_1; q3=q3_1;
>> eval(T03prima)
ans =
    0.0000    1.0000   -0.0000    0.0000
    0.6897   -0.0000   -0.7241    1.0000
   -0.7241         0   -0.6897    1.2000
         0         0         0    1.0000
```

2

```
>> q1=q1; q2=q2_2; q3=q3_2;
>> eval(T03prima)
ans =
    0.0000    1.0000    0.0000    0.0000
    0.0000   -0.0000    1.0000    1.0000
    1.0000         0   -0.0000    1.2000
         0         0         0    1.0000
```

Como vemos, podemos comprobar que para las 2 ramas de soluciones de q1, q2, q3 obtenidas, conseguimos la posición esperada, analizada en 1). Sin embargo, aunque a priori puede parecer que las 2 ramas son válidas entonces, habría que pararse a ver si son físicamente posibles, ya que por ejemplo recordamos que q3 tenía que estar en el rango: [L3/2; L3+L4], por la estructura física de las barras y como se definieron los sistemas de referencia, es decir: [0.3; 1], luego las soluciones anteriores que implican q3 = -1 no serían válidas. Habría que analizar estas situaciones también con los ángulos que pueden girar q1 y q2.

Notar que, de las ramas de soluciones anteriores, la rama 2 nos proporciona las Qs que introdujimos en 1) y por tanto nos resulta en la misma T03prima de nuevo.

Esta comprobación se podría realizar con más posiciones distintas, pero como es relativamente extensa, sólo la desarrollo para la “posición de dibujo”.

2.5.- Dibujo de trayectoria circular en el plano cartesiano X-Y

Teniendo la Cinemática Inversa, se propone realizar una trayectoria circular por parte del brazo. Para ello, se elige una circunferencia que esté completamente dentro del espacio de trabajo del robot con: **centro en (0.2, 0.8, 1) [m] y radio = 0.2 [m].**
Enlace a animación de la trayectoria en gif:

https://drive.google.com/file/d/175dTSW0A2Pwo6-V7eVy-MQk3hKSLy4oo/view?usp=sharing

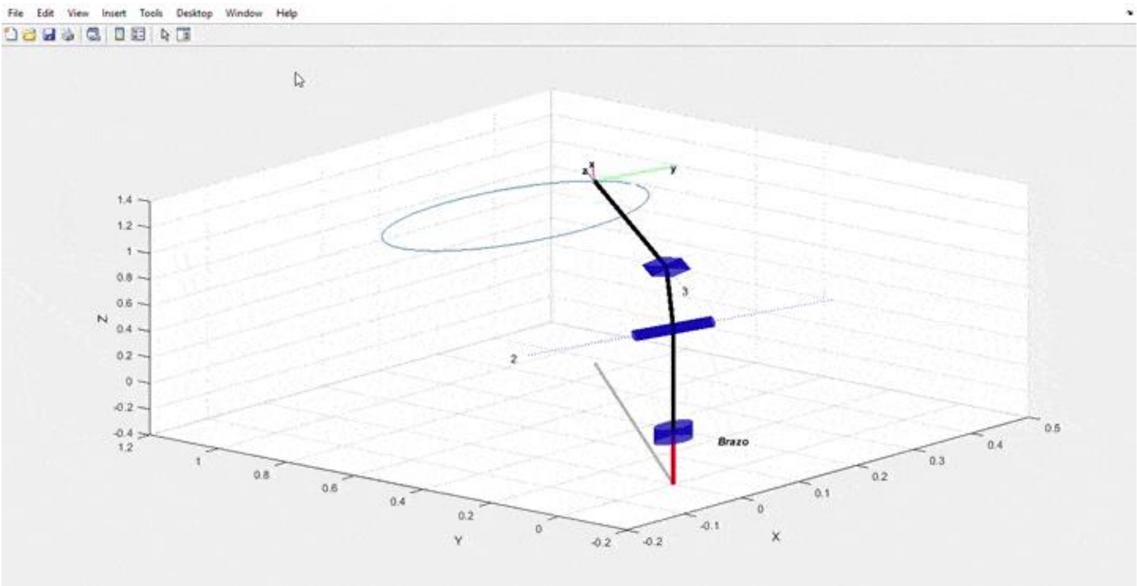


Figura 11: Captura del brazo durante la animación de la trayectoria circular + enlace a gif animado

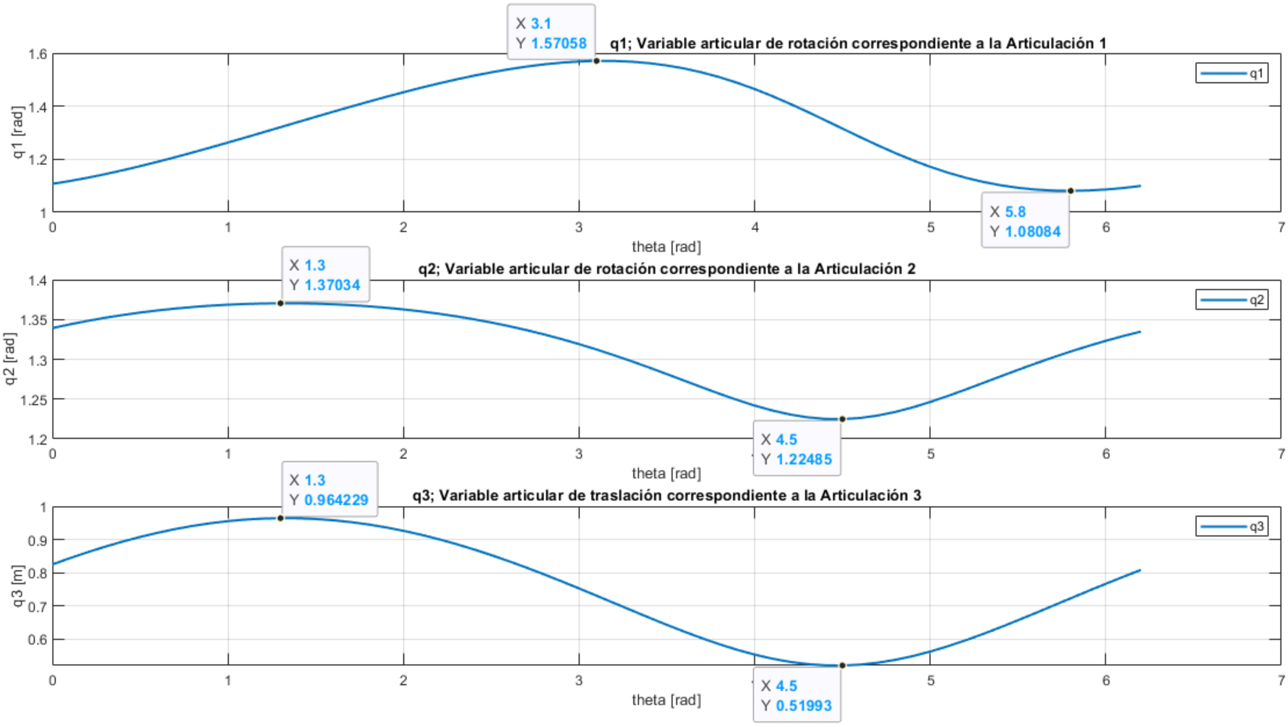


Figura 12: Gráficas mostrando la evolución de las variables articulares (q_i) para la rama de soluciones escogida (válida)

Como vemos en el Codigo_Trayectoria para conseguir estos resultados, no comentaré mucho el funcionamiento del código pues está comentado tanto en el .m, como en el anexo. Lo que si es merecedor de mención es que se ha usado una de las dos ramas, concretamente la siguiente: q1 ---> q2_2 ---> q3_2; Esto se ha escogido así puesto que como se ve en la siguiente gráfica q3 sale con valores negativos, y, aunque el programa haga funcionar la animación, físicamente no es posible por la estructura del robot que q3 tenga esos valores. Sin embargo, para la rama elegida (gráfica anterior a esta explicación), q3 si tiene unos valores aptos para nuestro robot y por tanto admisibles, por lo que nos quedamos con esta rama de soluciones de la Cinemática Inversa para que el robot realice la trayectoria circular.

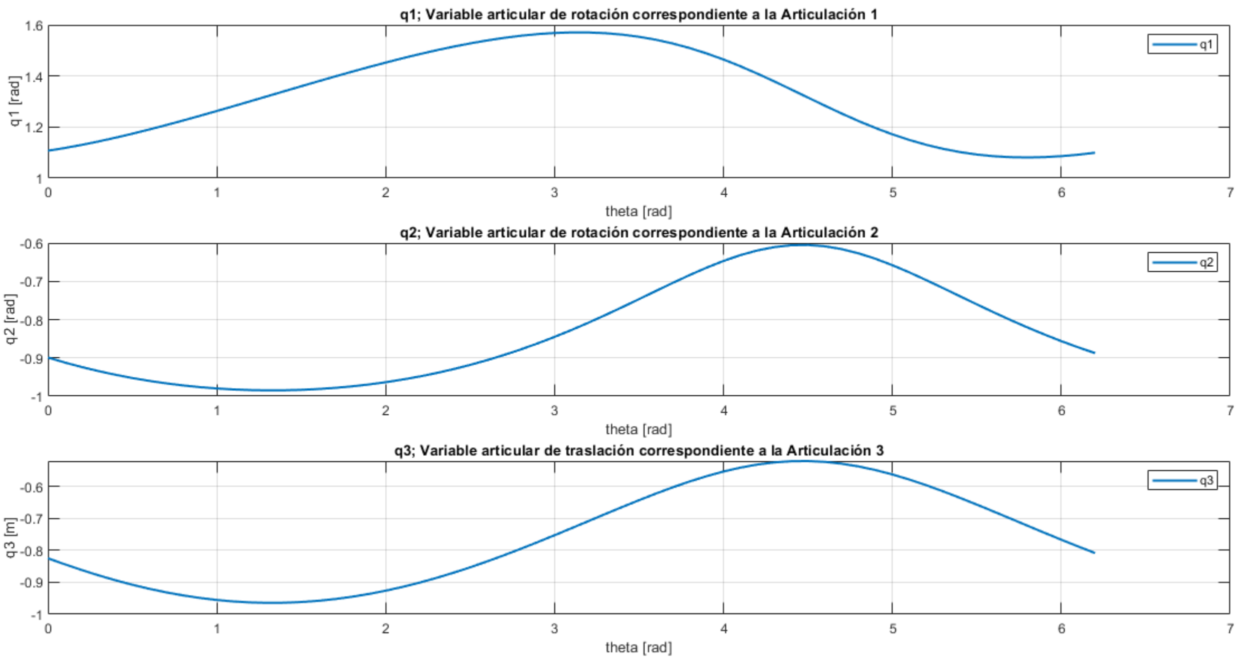


Figura 13: Gráficas mostrando la evolución de las variables articulares (q_i) para la rama de soluciones DESCARTADA (NO válida)

Como se mencionaba anteriormente, se comprueba en esta última gráfica que para la rama que se ha descartado, q3 toma valores negativos, que no son admisibles físicamente y por este mismo motivo se ha descartado.

2.6.- Cálculo de jacobianos directos e inversos y estudio de posibles singularidades

Se anexa el código para todos los cálculos posteriores Jacobianos;

Se busca primero hallar el Jacobiano directo analítico, para ello partimos del MCD, quedándonos con la posición (px, py, pz), una vez se han sustituido los valores numéricos de las longitudes:

```
px = cos(q1)*((2*cos(q2))/5 + q3*sin(q2));
py = sin(q1)*((2*cos(q2))/5 + q3*sin(q2));
pz = (2*sin(q2))/5 - q3*cos(q2) + 4/5;
```

Con esto y utilizando la expresión para calcular el Jacobiano directo, considerando sólo la posición (px, py, pz) al tener sólo 3 gdl:

$$J = \begin{pmatrix} \frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \frac{\partial p_x}{\partial q_3} \\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \frac{\partial p_y}{\partial q_3} \\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \frac{\partial p_z}{\partial q_3} \end{pmatrix}$$

se obtiene:

```
>> J = [diff(px,q1)     diff(px,q2)     diff(px,q3);
     diff(py,q1)     diff(py,q2)     diff(py,q3);
     diff(pz,q1)     diff(pz,q2)     diff(pz,q3)]
J =
[-sin(q1)*((2*cos(q2))/5 + q3*sin(q2)), -cos(q1)*((2*sin(q2))/5 - q3*cos(q2)), cos(q1)*sin(q2)]
[ cos(q1)*((2*cos(q2))/5 + q3*sin(q2)), -sin(q1)*((2*sin(q2))/5 - q3*cos(q2)), sin(q1)*sin(q2)]
[                                     0,             (2*cos(q2))/5 + q3*sin(q2),        -cos(q2)]
```

Mientras que para el Jacobiano Inverso simplemente se invierte la matriz J:

```
>> Jinv = simplify(inv(J))
Jinv =
[    -(5*sin(q1))/(2*cos(q2) + 5*q3*sin(q2)),       (5*cos(q1))/(2*cos(q2) + 5*q3*sin(q2)),                            0]
[                    (cos(q1)*cos(q2))/q3,                         (cos(q2)*sin(q1))/q3,                   sin(q2)/q3]
[(cos(q1)*(2*cos(q2) + 5*q3*sin(q2)))/(5*q3), (sin(q1)*(2*cos(q2) + 5*q3*sin(q2)))/(5*q3), (2*sin(q2) - 5*q3*cos(q2))/(5*q3)]
```

Tras esto se halla el determinante del jacobiano directo (J):

```
>> Determinante = simplify(det(J))
Determinante =
sin(q2)*q3^2 + (2*cos(q2)*q3)/5
```

$$|J| = \frac{q3 \cdot (q3 \cdot sen(q2) + 2 \cdot \cos(q2))}{5}$$

Finalmente, se busca hallar los puntos singulares, igualando este determinante a 0 y viendo que valores de q2 y q3 hacen 0 el determinante:

$$|J| = \frac{q3 \cdot (q3 \cdot sen(q2) + 2 \cdot \cos(q2))}{5} = 0$$

```
>> Singularidad = simplify(0 == Determinante)
Singularidad =
2*cos(q2) + 5*q3*sin(q2) == 0 | q3 == 0
```

Vemos que una condición clara es q3 = 0, que lo anula siempre. Para la de la izquierda, podemos buscar una serie de parejas de q2 y q3 que cumplan esa ecuación y por tanto sean configuraciones singulares, tales como:

```
q3 = 2/5; q2 = 3*pi/4 + n*pi
q3 = -2/5; q2 = pi/4 + n*pi
con n=[0,1,2,3,…]
```

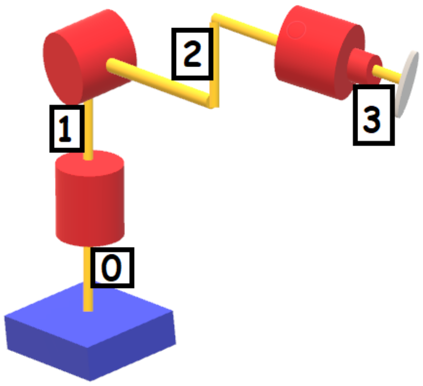
Es posible que haya más parejas que cumplan la ecuación, no estaba seguro de si había más soluciones

Recordar que q3 = -2/5 no era posible físicamente por la disposición de los eslabones

3.- Análisis Dinámico

3.1.- Cálculo de Parámetros Dinámicos

En primer lugar, para esta parte del trabajo, es necesario calcular los parámetros dinámicos del robot de 3 GDL. Para ellos rescatamos el modelo de dicho robot junto a la descripción ofrecida de las características dinámicas correspondientes:

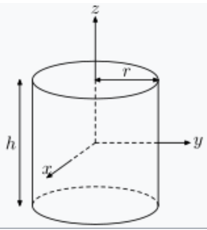


Parámetros Dinámicos Brazos	Enlace	m (kg)	Centros de masas e inercias	K_r
a	1	5	Se considerará que todos los eslabones son macizos y de densidad constante. La sección de todos los eslabones es circular, y es la misma para un mismo robot. El valor del radio de la sección será igual a L1/20..	25
	2	5.5		20
	3	3		25
b	1	5.5		25
	2	4		20
	3	3.5		25
c	1	4.5		25
	2	4		20
	3	3.2		25
	1	4		25

Partiendo de esto, se puede plantear la resolución de parámetros dinámicos de cada eslabón. Como se menciona que los eslabones son macizos y de sección circular tenemos que sus tensores de inercia responden a la siguiente expresión:

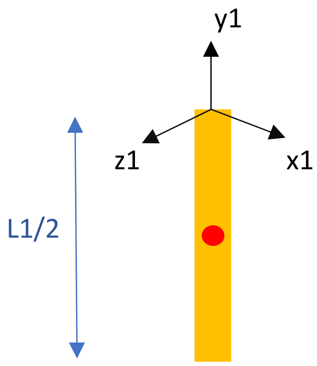
Código para cálculo de parámetros dinámicos: Cálculo_Parámetros_Dinámicos

Cilindro rígido de radio r, altura h y masa m



$$I = \begin{bmatrix} \frac{1}{12}m(3r^2+h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2+h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix}$$

- **<u>Eslabón 1:</u>**



Se plantea la resolución de los parámetros de este eslabón con el siguiente bloque de código:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ESLABÓN 1 (Sin desdoble, 1 barra (longitud L1/2)) alineado con DH
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M1= 4.5; %kg
X1cdm= 0;
Y1cdm= -L1/4;
Z1cdm= 0;
Pcdm1 = [X1cdm;Y1cdm;Z1cdm];

I1xx=(1/12)*M1*(3*(R^2)+(L1/2)^2);
I1zz=I1xx;
I1yy=0.5*M1*R^2;
I1=[I1xx, 0, 0; 0, I1yy, 0; 0, 0, I1zz];

Jm1=min([I1(1,1),I1(2,2),I1(3,3)]);
```

El término (1/2)·m·r^2, según el esquema anterior, se corresponde con la componente del "eje y" del tensor, que corresponde a "I1yy" en este código

Ejecutando el código obtenemos los siguientes parámetros para este eslabón:

```
>> I1
I1 =
    0.0618         0         0
         0    0.0036         0
         0         0    0.0618
>> Jm1
Jm1 =
    0.0036
```

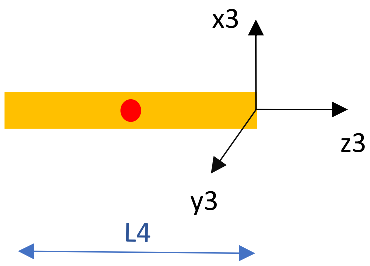
Expresamos estos resultados:

$$I1 = \begin{bmatrix} 0.0618 & 0 & 0 \\ 0 & 0.0036 & 0 \\ 0 & 0 & 0.0036 \end{bmatrix} \mathrm{kg \cdot m^2}$$

$$\mathrm{Jm1} = 0.0036\ \mathrm{kg} \cdot m^2$$

A continuación, se plantea el eslabón 3, que al ser una única barra también queda bastante simple su análisis:

- **Eslabón 3:**



Se plantea la resolución de los parámetros de este eslabón con el siguiente bloque de código:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ESLABÓN 3 (Sin desdoble, 1 barra (longitud L4)) alineado con DH
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M3= 3.2; %kg
X3cdm=0;
Y3cdm=0;
Z3cdm= -L4/2;
Pcdm3 = [X3cdm;Y3cdm;Z3cdm];

I3xx=(1/12)*M3*(3*(R^2)+(L4)^2);
I3yy=I3xx;
I3zz=0.5*M3*R^2;
I3=[I3xx, 0, 0; 0, I3yy, 0; 0, 0, I3zz];

Jm3=min([I3(1,1),I3(2,2),I3(3,3)]);
```

El término (1/2)·m·r^2, según el esquema anterior, se corresponde con la componente del "eje z" del tensor, que corresponde a "I3zz" en este código

Ejecutando el código obtenemos los siguientes parámetros para este eslabón:

```
>> I3
I3 =
    0.0439         0         0
         0    0.0439         0
         0         0    0.0026
>> Jm3
Jm3 =
    0.0026
```

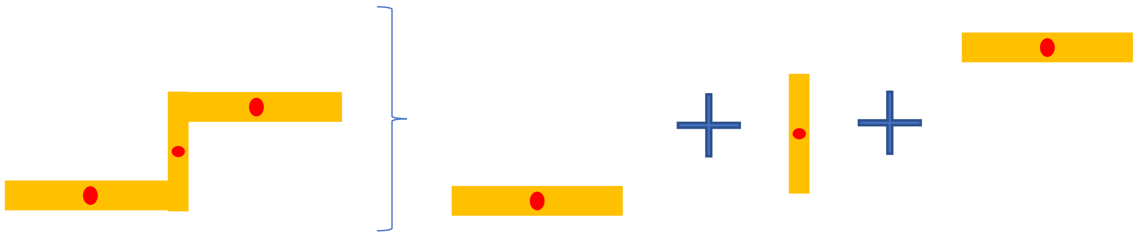
Expresamos estos resultados:

$$I3 = \begin{bmatrix} 0.0439 & 0 & 0 \\ 0 & 0.0439 & 0 \\ 0 & 0 & 0.0026 \end{bmatrix} \mathrm{kg \cdot m^2}$$

$$\mathrm{Jm3} = 0.0026\ \mathrm{kg} \cdot m^2$$

- **<u>Eslabón 2:</u>**

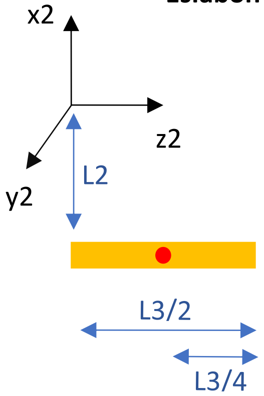
Finalmente, para plantear el eslabón 2, hay que considerar que consiste en una barra articulada, la cual debe ser descompuesta en 3 barras para su análisis:



Eslabón 2 articulado (completo) | Eslabón 2A | Eslabón 2B | Eslabón 2C

Tras ver el planteamiento para este eslabón articulado, podemos ir resolviendo cada uno de los "sub-eslabones":

- **Eslabón 2A:**



Podemos ver que la posición del centro de masas de este eslabón, respecto de estos ejes queda:

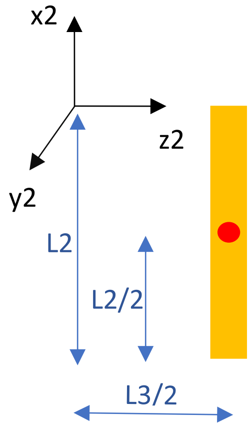
$$Pcdm2A = \begin{bmatrix} -L2 \\ 0 \\ \frac{L3}{4} \end{bmatrix}$$

Usando un código equivalente a los eslabones individuales anteriores se resuelve este sub-eslabón:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ESLABÓN 2 (Con desdoble, 3 barras (longitudes L3/2; L2; L3/2)) alineado
% con DH
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M2 = 4; %kg
L2A = L3/2;
L2B = L2;
L2C = L3/2;

%  ESLABÓN 2A
M2A= M2*(L2A/(L2A+L2B+L2C));
I2Axx=(1/12)*M2A*(3*(R^2)+(L2A)^2);
I2Ayy=I2Axx;
I2Azz=0.5*M2A*R^2;
I2A=[I2Axx, 0, 0; 0, I2Ayy, 0; 0, 0, I2Azz];
```

- **Eslabón 2B:**

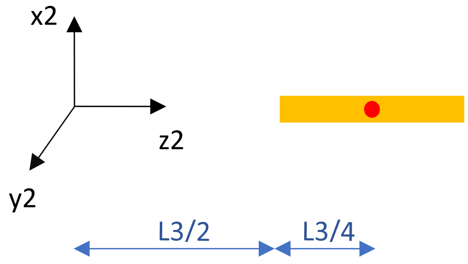


Podemos ver que la posición del centro de masas de este eslabón, respecto de estos ejes queda:

$$Pcdm2B = \begin{bmatrix} -\frac{L2}{2} \\ 0 \\ \frac{L3}{2} \end{bmatrix}$$

```
%  ESLABÓN 2B
M2B= M2*(L2B/(L2A+L2B+L2C));
I2Bxx=0.5*M2B*R^2;
I2Byy=(1/12)*M2B*(3*(R^2)+(L2B)^2);
I2Bzz=I2Byy;
I2B=[I2Bxx, 0, 0; 0, I2Byy, 0; 0, 0, I2Bzz];
```

- **Eslabón 2C:**



Podemos ver que la posición del centro de masas de este eslabón, respecto de estos ejes queda:

$$Pcdm2C = \begin{bmatrix} 0 \\ 0 \\ L3 \cdot \left(\frac{3}{4}\right) \end{bmatrix}$$

```
%  ESLABÓN 2C
M2C= M2*(L2C/(L2A+L2B+L2C));
I2Cxx=(1/12)*M2C*(3*(R^2)+(L2C)^2);
I2Cyy=I2Cxx;
I2Czz=0.5*M2C*R^2;
I2C=[I2Cxx, 0, 0; 0, I2Cyy, 0; 0, 0, I2Czz];
```

Tras haber planteado cada sub-eslabón, hay que comenzar con la superposición de los 3, para conseguir obtener el tensor de inercia correspondiente al eslabón articulado completo. No muestro los resultados de I2A, I2B e I2C, ya que son pasos intermedios y no tienen tanto interés.

Para ello, en primer lugar, hallamos el centro de masas del eslabón conjunto:

```
% Centro de masas del eslabón
M2=M2A+M2B+M2C;
Pcdm2A=[-L2; 0; L3/4];
Pcdm2B=[-L2/2; 0; L3/2];
Pcdm2C = [0; 0; L3*(3/4)];
Pcdm2=(M2A*Pcdm2A+M2B*Pcdm2B+M2C*Pcdm2C)/(M2A+M2B+M2C);
```

De donde conseguimos las coordenadas del centro de masas "Pcdm2":

```
>> Pcdm2
Pcdm2 =
   -0.2000
         0
    0.3000
```

$$P_{cdm2} = \begin{bmatrix} -0.2 \\ 0 \\ 0.3 \end{bmatrix} m$$

Ahora, utilizando el Teorema de Steiner, mediante el siguiente bloque de código hallamos a partir de esos tensores anteriores I2A, I2B e I2C, los tensores de inercia de cada sub-eslabón respecto de este centro de masas "conjunto" (Pcdm2). Tras, esto teniendo los tensores de inercia de cada contribución individual respecto de cdm2, basta con sumarlos para tener el tensor de inercias del eslabón 2 conjunto.

```
% Vector desde cdm2A a cdm2
Pcdm2A_cdm2=Pcdm2-Pcdm2A;
% Inercia de parte 2A respecto a cdm2
%  rx=Pcdm2A_cdm2(1); ry=Pcdm2A_cdm2(2); rz=Pcdm2A_cdm2(3);
%  RSteiner2A=[ry^2+rz^2, -rx*ry, -rx*rz; -rx*ry, rx^2+rz^2, -ry*rz; -rx*rz, -ry*rz, rx^2+ry^2];
% Pero más compacto
DespSteiner2A=norm(Pcdm2A_cdm2)^2*eye(3)-Pcdm2A_cdm2*Pcdm2A_cdm2';
I2Acdm2=I2A+M2A*DespSteiner2A;

% Vector desde cdm2B a cdm2
Pcdm2B_cdm2=Pcdm2-Pcdm2B;
% Inercia de parte 2B respecto a cdm2
%  rx=Pcdm2B_cdm2(1); ry=Pcdm2B_cdm2(2); rz=Pcdm2B_cdm2(3);
%  RSteiner2B=[ry^2+rz^2, -rx*ry, -rx*rz; -rx*ry, rx^2+rz^2, -ry*rz; -rx*rz, -ry*rz, rx^2+ry^2];
% Pero más compacto
DespSteiner2B=norm(Pcdm2B_cdm2)^2*eye(3)-Pcdm2B_cdm2*Pcdm2B_cdm2';
I2Bcdm2=I2B+M2B*DespSteiner2B;

% Vector desde cdm2C a cdm2
Pcdm2C_cdm2=Pcdm2-Pcdm2C;
% Inercia de parte 2C respecto a cdm2
%  rx=Pcdm2C_cdm2(1); ry=Pcdm2C_cdm2(2); rz=Pcdm2C_cdm2(3);
%  RSteiner2C=[ry^2+rz^2, -rx*ry, -rx*rz; -rx*ry, rx^2+rz^2, -ry*rz; -rx*rz, -ry*rz, rx^2+ry^2];
% Pero más compacto
DespSteiner2C=norm(Pcdm2C_cdm2)^2*eye(3)-Pcdm2C_cdm2*Pcdm2C_cdm2';
I2Ccdm2=I2C+M2C*DespSteiner2C;

% Inercia del eslabón 2 completo sobre cdm
I2cdm2=I2Acdm2+I2Bcdm2+I2Ccdm2;
Jm2=min([I2cdm2(1,1),I2cdm2(2,2),I2cdm2(3,3)]);
```

Tras todo este procedimiento para el eslabón 2 (con 3 sub-eslabones), finalmente tenemos su tensor de inercias buscado (I2cdm2), que ejecutando todo el código conjunto analizado proporciona:

```
>> I2cdm2
I2cdm2 =
    0.0742         0   -0.0720
         0    0.1909         0
   -0.0720         0    0.1199
>> Jm2
Jm2 =
    0.0742
```

Expresamos estos resultados:

$$\mathrm{I2} = \begin{bmatrix} 0.0742 & 0 & -0.0720 \\ 0 & 0.1909 & 0 \\ -0.0720 & 0 & 0.1199 \end{bmatrix} \mathrm{kg} \cdot \mathrm{m}^2$$

$$\mathrm{Jm2} = 0.0742\ \mathrm{kg} \cdot m^2$$

3.2.- Obtención del modelo dinámico

Tras obtener los parámetros dinámicos correspondientes a los diferentes eslabones, se puede proceder a trabajar en el modelo dinámico del Robot y la obtención de las ecuaciones.

Para la obtención de las ecuaciones dinámicas se ha utilizado el código propuesto que implementa el "Método de Newton-Euler" **Newton_Euler_Dinámica**, adaptado para este robot (C2) en concreto. Para no hacer muy extensa la memoria, adjunto el código anexado mediante el que se obtienen las siguientes ecuaciones que modelan la dinámica, recogiéndose las versiones "_ne" de las matrices, ya que estas resuelven las fracciones de precisión infinita del cálculo simbólico y son suficientemente precisas para tomarlas como módelo dinámico (5 decimales).

Sabemos que la ecuación que modela la dinámica del robot tiene la siguiente forma:

$$\tau = M_A(q) \cdot \ddot{q} + V_A(q, \dot{q}) + G_A(q)$$

Donde τ son los pares aplicados a las articulaciones, M_A es la matriz ampliada de inercias, V_A es la matriz de términos de Coriolis ampliada y G_A la matriz de términos gravitatorios ampliada.

A partir de dichas matrices se puede conocer el par que es necesario aplicar a las articulaciones en función de las variables articulares: q_i , $\dot{q}_i$, $\ddot{q}_i$ (posición, velocidad, aceleración)

Por tanto, recogemos así las matrices que modelan la dinámica:

- **Ma_ne (matriz 3x3):**

$$\begin{bmatrix} 5.12 \cdot q3 \cdot \cos^2(q2) - 1.336 \cdot \sin(2.0 \cdot q2) + 1.7316 \cdot \cos^2(q2) - 1.28 \cdot q3 \cdot \sin(2.0 \cdot q2) + 3.2 \cdot q3^2 \cdot \cos^2(q2) + 3.0481 & 0 & 0 \\ 0 & 3.2 \cdot q3^2 + 5.12 \cdot q3 + 32.995 & -1.28 \\ 0 & -1.28 & 4.825 \end{bmatrix}$$

- **Va_ne (matriz 3x1):**

$$\begin{bmatrix} 0.0225 \cdot qd1 - 0.0004 \cdot qd1 \cdot (6680.0 \cdot qd2 \cdot \cos(2.0 \cdot q2) - 8000.0 \cdot q3 \cdot qd3 - 6400.0 \cdot qd3 - 6400.0 \cdot qd3 \cdot \cos(2.0 \cdot q2) + 4329.0 \cdot qd2 \cdot \sin(2.0 \cdot q2)) \\ + 3200.0 \cdot qd3 \cdot \sin(2.0 \cdot q2) + 8000.0 \cdot q3^2 \cdot qd2 \cdot \sin(2.0 \cdot q2) + 6400.0 \cdot q3 \cdot qd2 \cdot \cos(2.0 \cdot q2) - 8000.0 \cdot q3 \cdot qd3 \cdot \cos(2.0 \cdot q2) \\ + 12800.0 \cdot q3 \cdot qd2 \cdot \sin(2.0 \cdot q2)) \\ \\ 0.0144 \cdot qd2 + 5.12 \cdot qd2 \cdot qd3 + 1.336 \cdot qd1^2 \cdot cos(2.0 \cdot q2) + 0.8658 \cdot qd1^2 \cdot sin(2.0 \cdot q2) + 1.28 \cdot q3 \cdot qd1^2 \cdot cos(2.0 \cdot q2) \\ + 2.56 \cdot q3 \cdot qd1^2 \cdot sin(2.0 \cdot q2) + 1.6 \cdot q3^2 \cdot qd1^2 \cdot sin(2.0 \cdot q2) + 6.4 \cdot q3 \cdot qd2 \cdot qd3 \\ \\ 0.0225 \cdot qd3 - 2.56 \cdot qd1^2 \cdot \cos^2(q2) - 3.2 \cdot q3 \cdot qd2^2 + 0.64 \cdot qd1^2 \cdot sin(2.0 \cdot q2) - 2.56 \cdot qd2^2 - 3.2 \cdot q3 \cdot qd1^2 \cdot \cos^2(q2) \end{bmatrix}$$

- **Ga_ne (matriz 3x1):**

$$\begin{bmatrix} 0 \\ g \cdot (3.76 \cdot cos(q2) - 2.08 \cdot sin(q2) + 3.2 \cdot q3 \cdot cos(q2)) \\ 3.2 \cdot g \cdot sin(q2) \end{bmatrix}$$

3.3.- Comprobaciones sobre el modelo dinámico

Tras haber conseguido estas ecuaciones que, como hemos visto, modelan la dinámica del robot, podemos realizar unas comprobaciones sobre la validez de dicho modelo dinámico haciendo comparaciones de los resultados para q_i , $\dot{q}_i$, $\ddot{q}_i$ tanto aplicando estas ecuaciones como haciendo uso del toolbox de robótica (RTB). Luego usando el código **Dinámica_RTB** se crea un objeto del tipo SerialLink configurado con la dinámica de mi robot en cuestión. Por tanto, teniendo ya ambos métodos preparados podemos realizar 2 tipos de comprobaciones:

- **Coincidencia numérica de qdd a través de la función ModeloDinamico_R3GDL y de la función "accel" propia del toolbox que también proporciona qdd por esa vía:**

Antes de realizar la comprobación, mencionar que, en mi caso, al tener un par prismático como tercer grado de libertad con un cierto offset no nulo, esto hace que aparezca un "bug" en el toolbox, el cual no tiene en cuenta dicho offset en sus cálculos de la dinámica, luego hay que pasárselos a mano a la función "accel" para que lo tenga en cuenta.

Para realizar estas comprobaciones, a la función "ModeloDinamico_R3GDL" hay que pasarle un vector de nueve componentes apilando q, qd y tau, mientras que a la función "accel" hay que pasarle los vectores q, qd y tau como parámetros de la función. Partiendo de que el offset del par prismático es L3+L4 = 1m y habiendo ejecutado el código que crea el objeto SerialLink necesario, realizamos lo siguiente:

```
>> q=[2 -3 5];
qd= [0 2 4];
tau=[3 3 3];
>> qdd_NE = ModeloDinamico_R3GDL([q qd tau])

qdd_NE =

    0.0283
   -0.6027
   16.7479

>> qdd_RTB = R3GDL.accel( q+[0 0 L3+L4],qd,tau)

qdd_RTB =

    0.0283
   -0.6041
   16.7466
```

```
>> q=[6 4 -2];
qd= [1 1.5 3];
tau=[-2 4 5];
>> qdd_NE = ModeloDinamico_R3GDL([q qd tau])

qdd_NE =

    3.0403
    0.2437
    3.7488

>> qdd_RTB = R3GDL.accel( q+[0 0 L3+L4],qd,tau)

qdd_RTB =

    3.0403
    0.2445
    3.7440
```

Vemos como con ambos métodos se obtienen resultados similares, quitando errores numéricos a partir del tercer decimal que se pueden considerar despreciables. Luego como una forma de comprobar la validez del modelo esto sería válido.

- **Graficar q, qd, qdd obtenidas a partir de Simulink con ambos métodos y ver de nuevo esta coincidencia analizada numéricamente de manera más visual:**

Usando el siguiente modelo de Simulink, que hace uso de las funciones "ModeloDinamico_R3GDL" y "slaccel", podemos realizar comprobaciones también de esta forma:

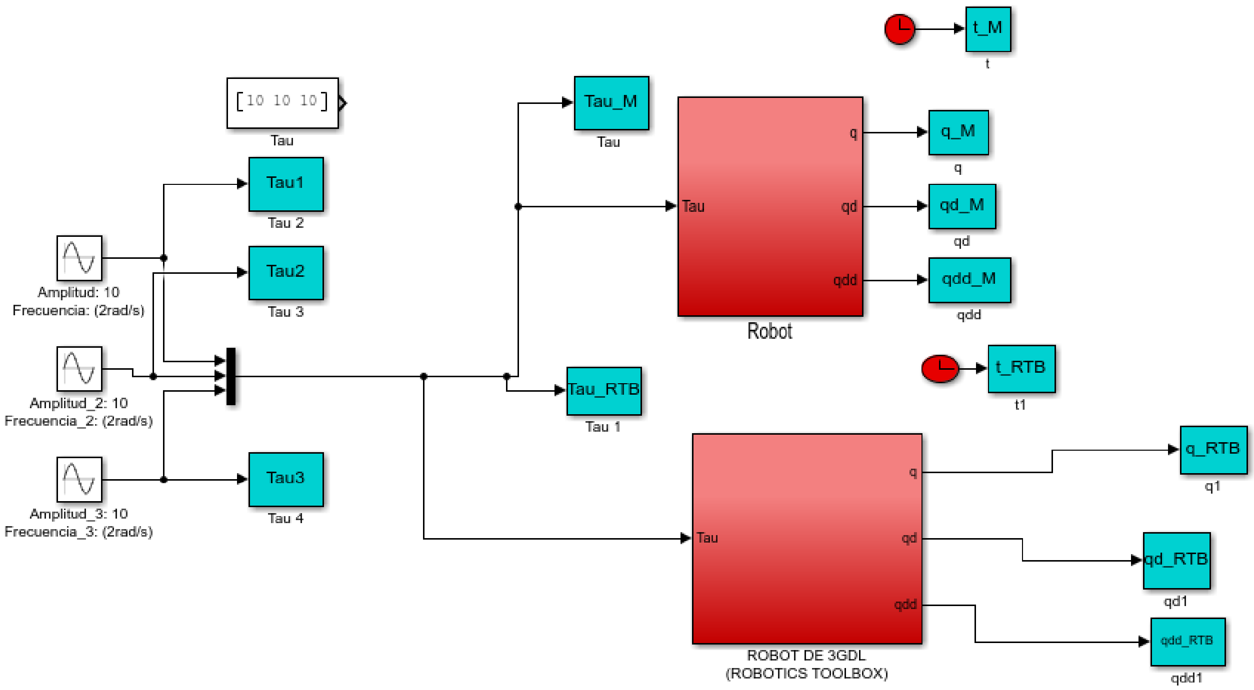


Figura 14: Modelo en Simulink para la comparación de resultados dinámicos por M-Function y por RTB (Robotics Toolbox)

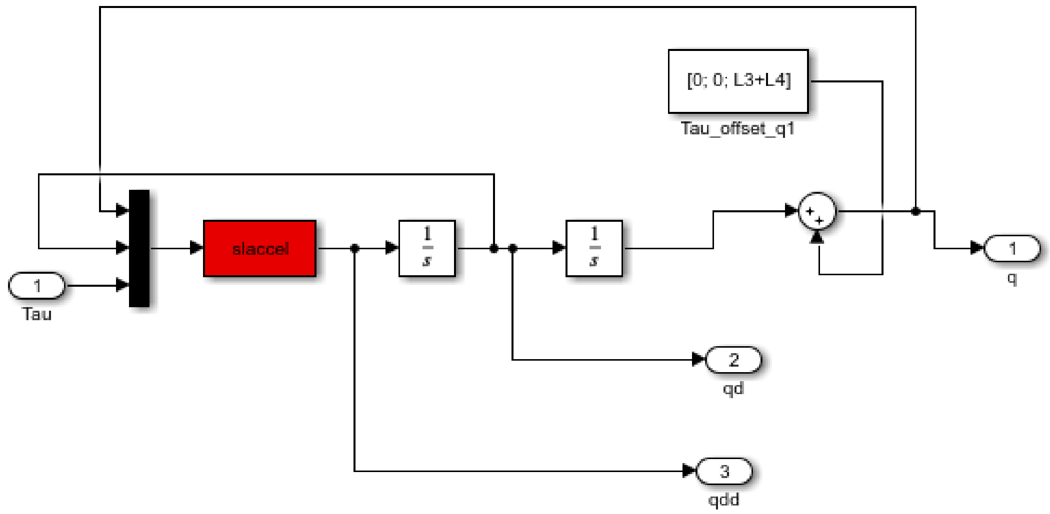
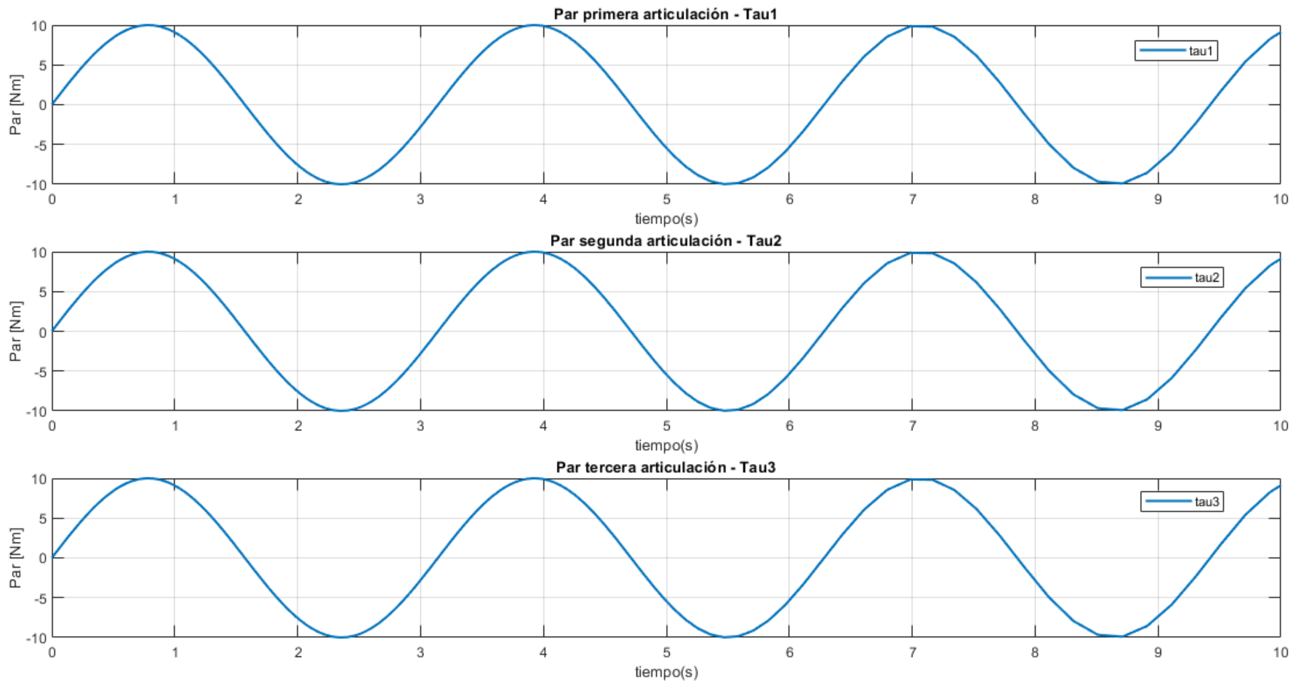


Figura 15: Modelo en Simulink para la implementación "manual" del offset del par prismático en el bloque RTB debido al "bug"

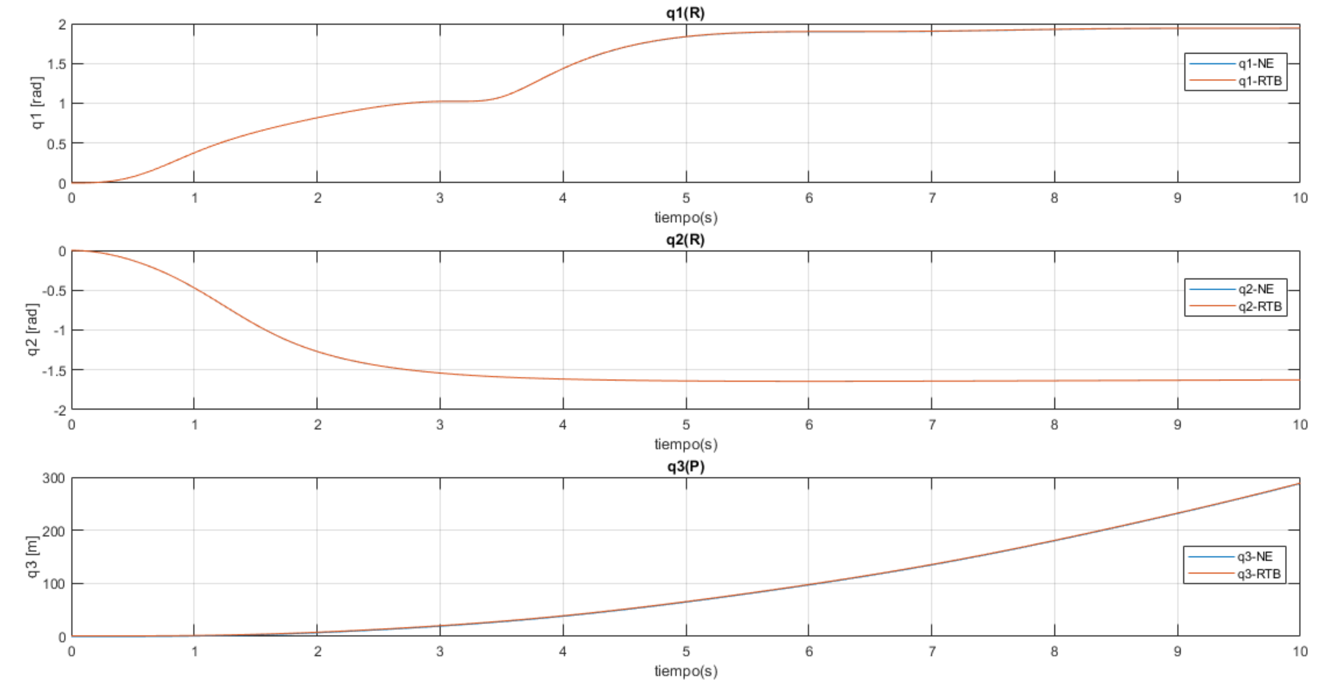
Aquí vemos como se implementa en Simulink la adición manual del offset que como se explicó anteriormente es necesaria para los resultados de la dinámica a través del toolbox.

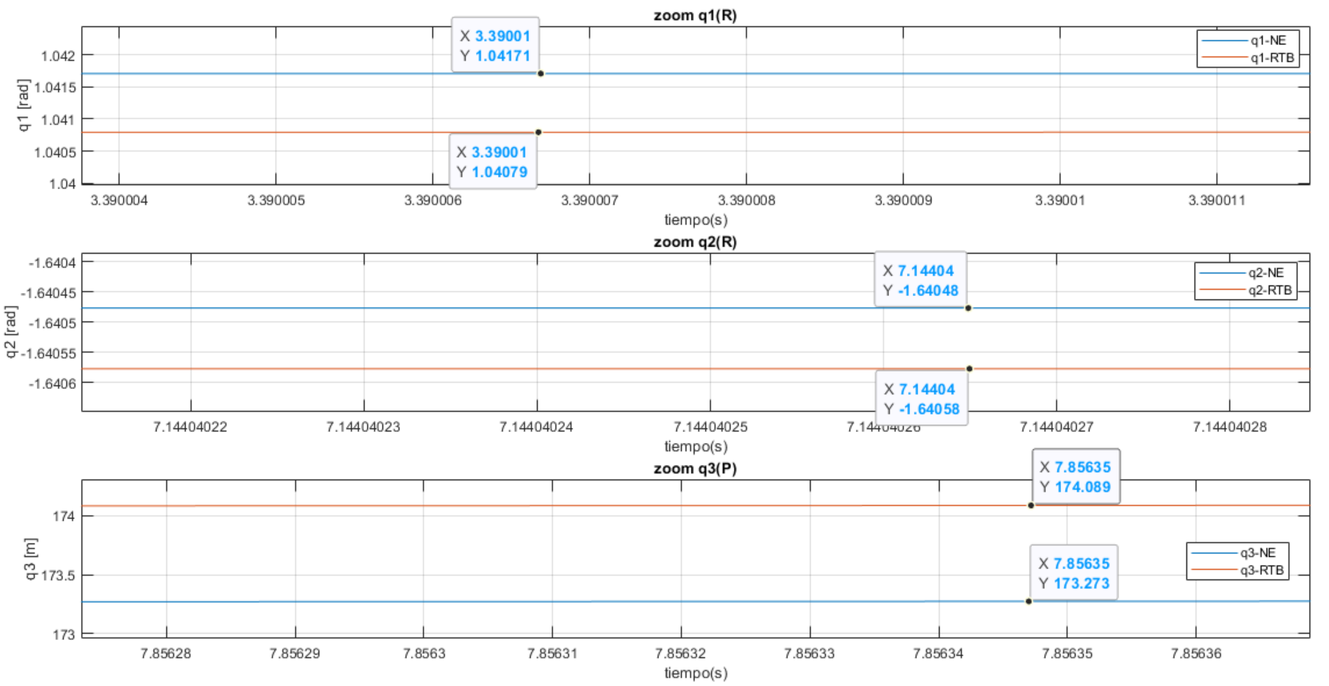
Para esto se propone el uso de un conjunto de pares senoidales para cada articulación del mismo valor:



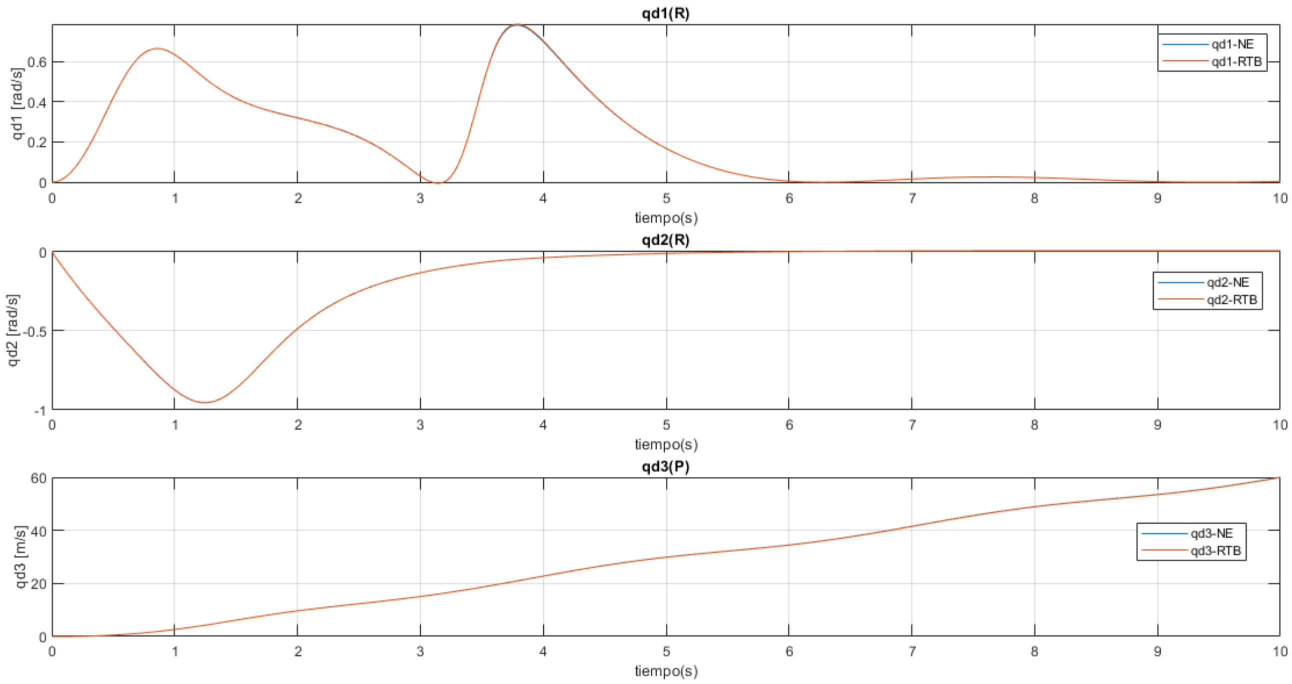
Con estos pares inyectados, se grafican a continuación gráficas comparativas de **q, qd, qdd** obtenidas por ambos métodos para corroborar la corrección del modelo de manera visual, pero equivalente a la comprobación numérica anterior.

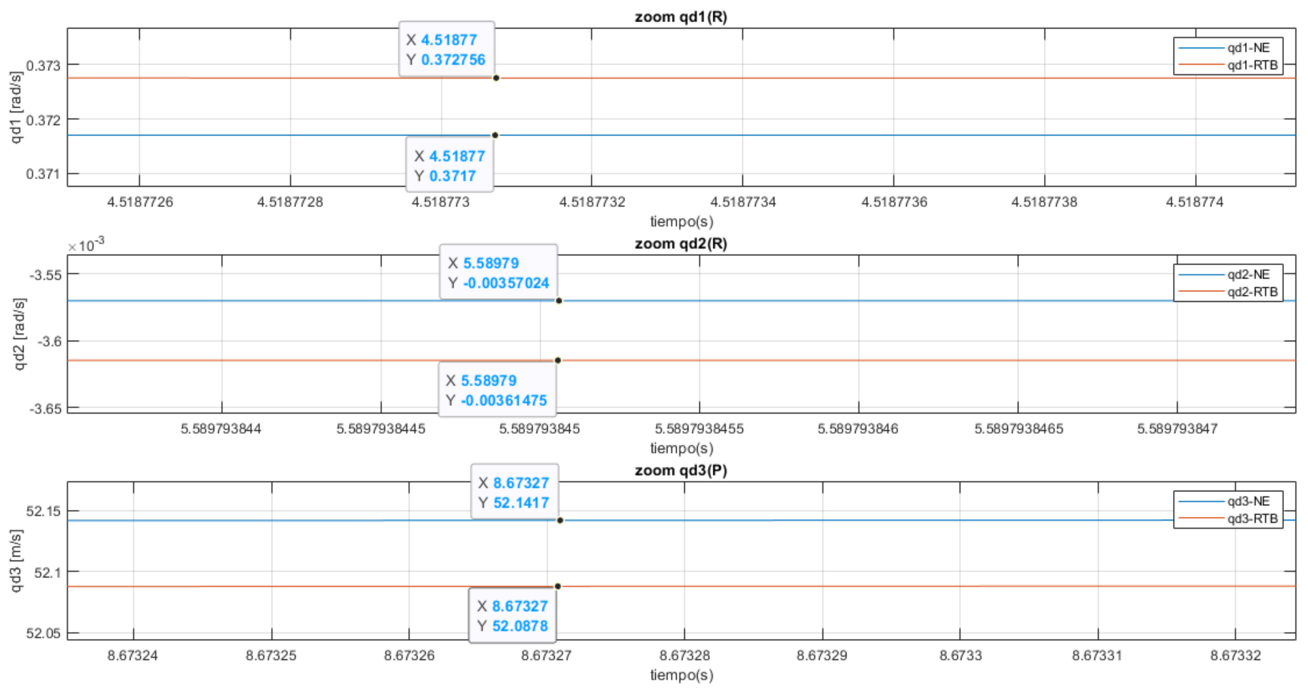
- **Para las posiciones articulares (q_i):**



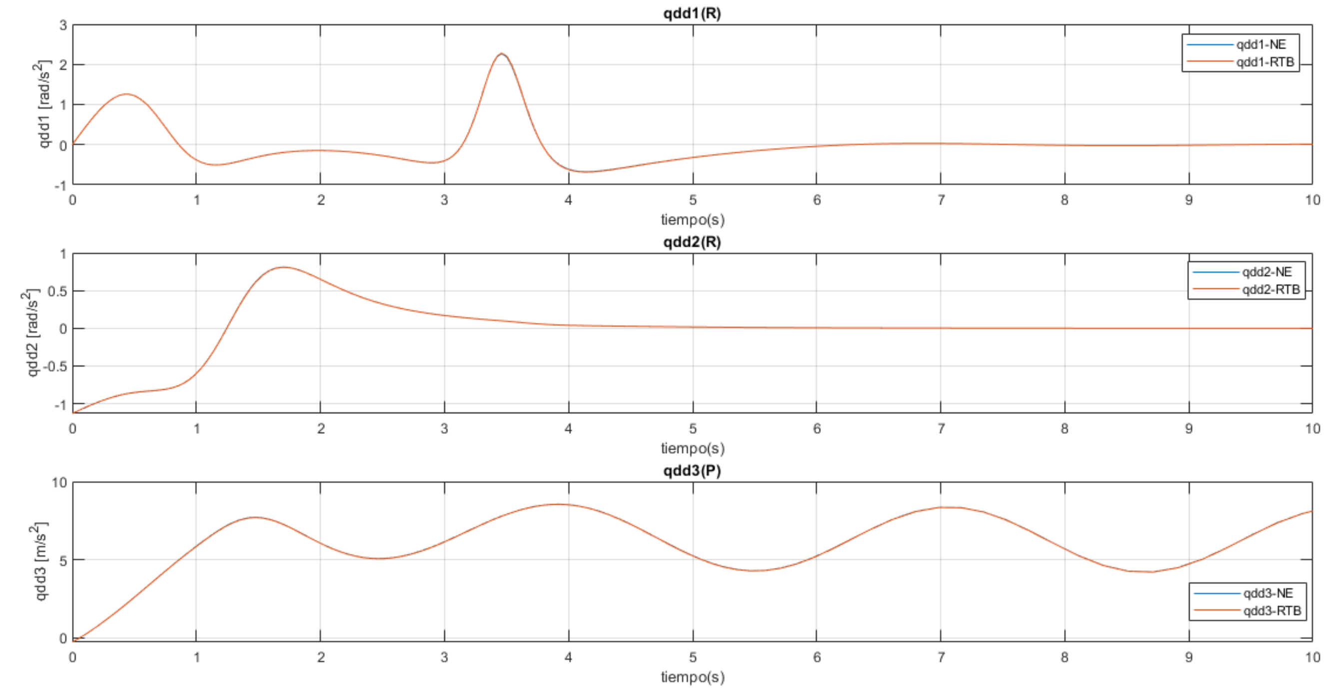


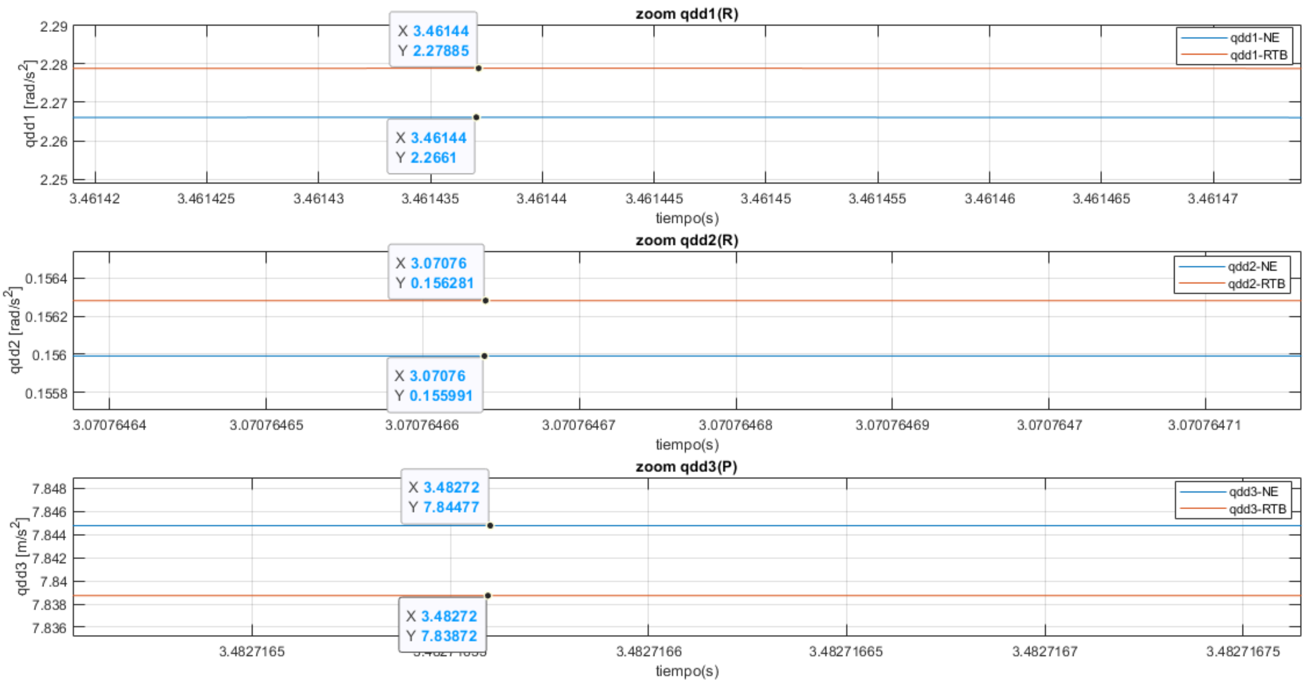
- **Para las velocidades articulares (qd_i):**





- **Para las aceleraciones articulares (qdd_i):**





Vemos entonces, que como esperábamos de las comprobaciones numéricas anteriores, de manera gráfica los resultados obtenidos con ambos modelos son prácticamente iguales, salvo discrepancias despreciables, observables en las gráficas con “zoom”, de cálculo numérico, lo cual nos puede servir como confirmación de que nuestras ecuaciones (Ma, Va, Ga) que modelan la dinámica parecen correctas.

4.- Control cinemático

En este apartado necesitamos recuperar tanto el modelo cinemático directo como el inverso del robot para mediante un generador de trayectorias, poder comprobar la validez de ambos. De esta manera, para una trayectoria recta entre dos puntos generada por dicho generador de trayectorias, se obtienen las variables articulares qr_i , $\dot{q}r_i$, $\ddot{q}r_i$ (posición, velocidad, aceleración). Por tanto, como el bloque GTCL obtiene qr_i , $\dot{q}r_i$, $\ddot{q}r_i$ como hemos dicho, si llevamos las qr_i obtenidas del bloque GTCL a nuestro modelo cinemático directo, deberíamos obtener la misma trayectoria recta que la solicitada si nuestro modelo es correcto.

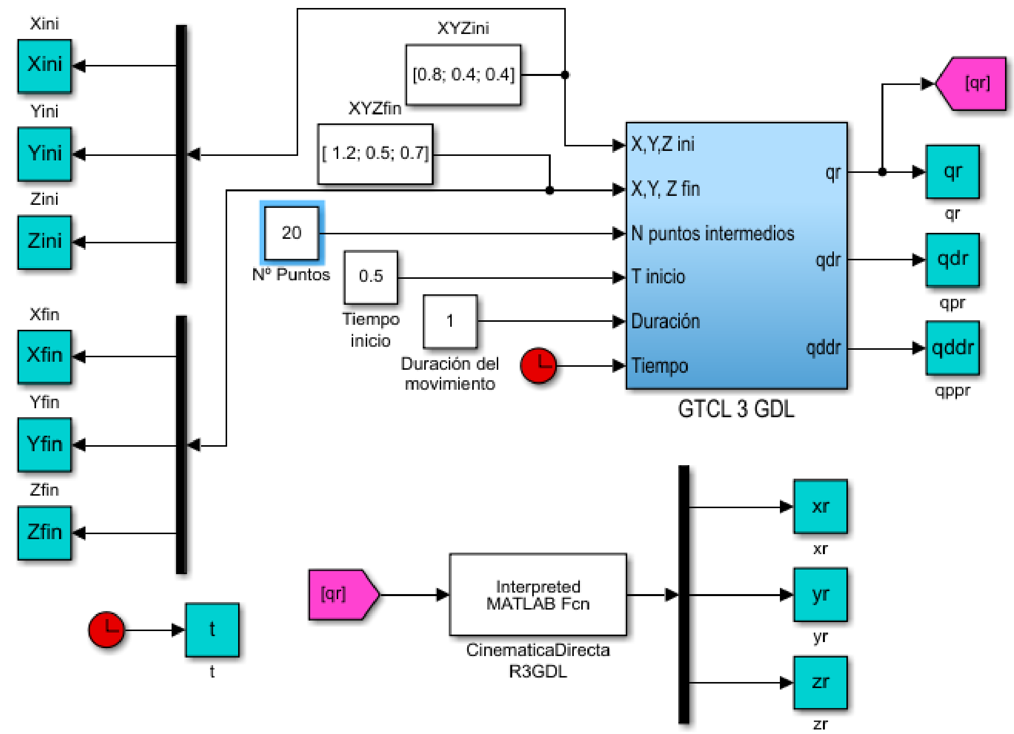


Figura 16: Modelo en Simulink para el generador de trayectorias y comprobación de modelo cinemático directo

En primer lugar y como comprobación más importante es la de si a partir de nuestra cinemática inversa y directa implementadas en Simulink, se devuelve la misma trayectoria (deseada) que la alimentada al bloque GTCL. Tomamos puntos alcanzables para el robot según su estructura física tal como los que vemos en la captura anterior:

$$P_{ini} = [0.8 \quad 0.4 \quad 0.4] \quad y \quad P_{fin} = [1.2 \quad 0.5 \quad 0.7]$$

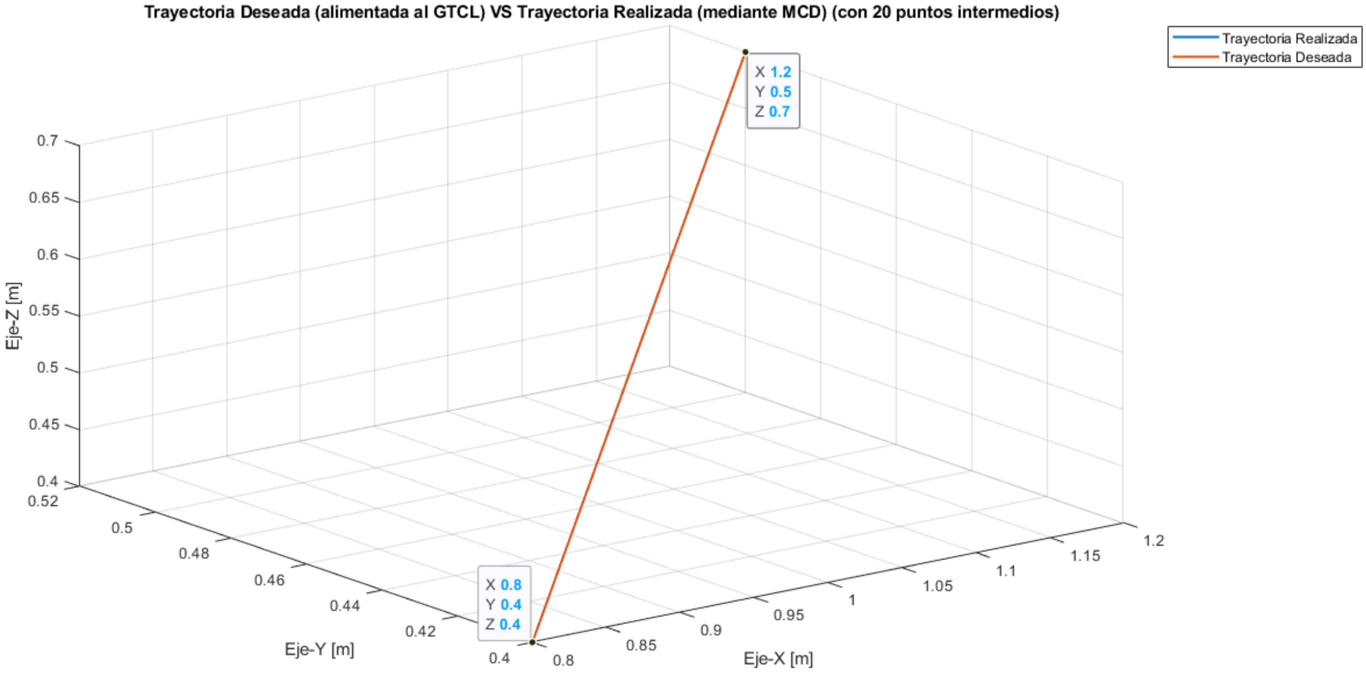
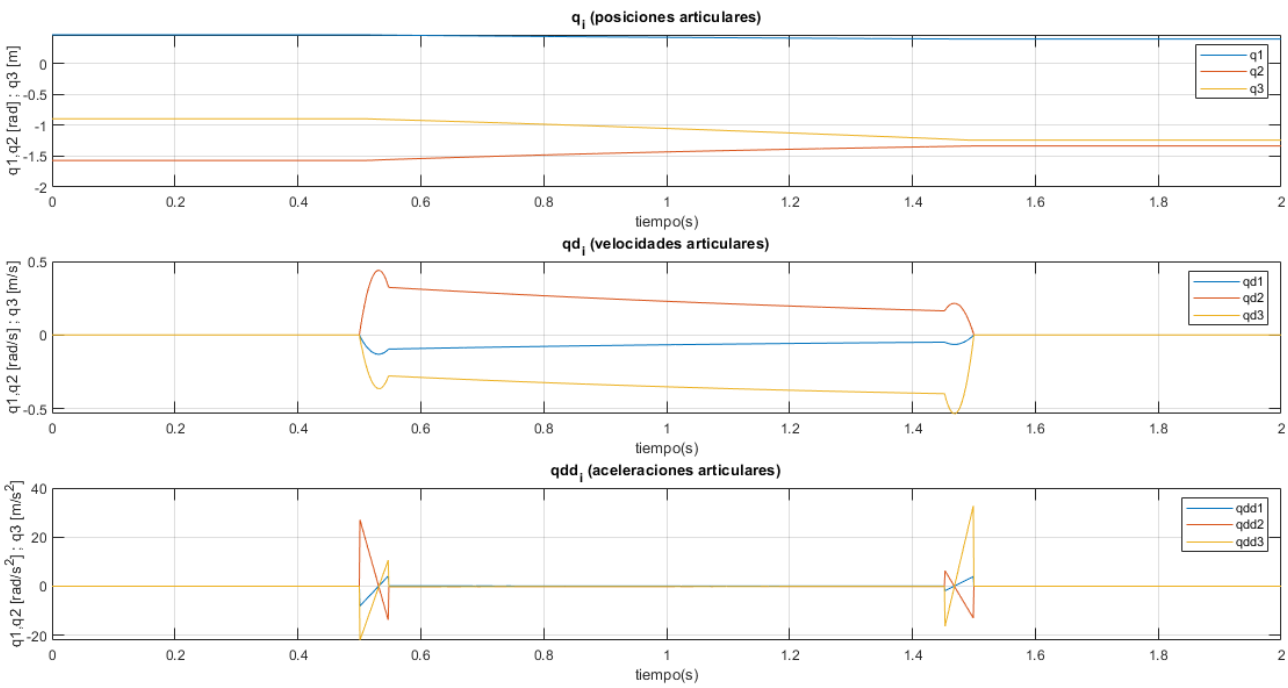


Figura 17: Comparación de trayectorias 3D deseada y realizada para 20 puntos intermedios en el GTCL

Se observa una pequeña discrepancia si se plotea y se gira entre las trayectorias, aun así, apenas apreciable, debido a que 20 puntos intermedios es una cantidad relativamente baja para el GTCL. Si se grafican las trayectorias de las variables articulares (q_i , $\dot{q}_i$, $\ddot{q}_i$) para este caso:



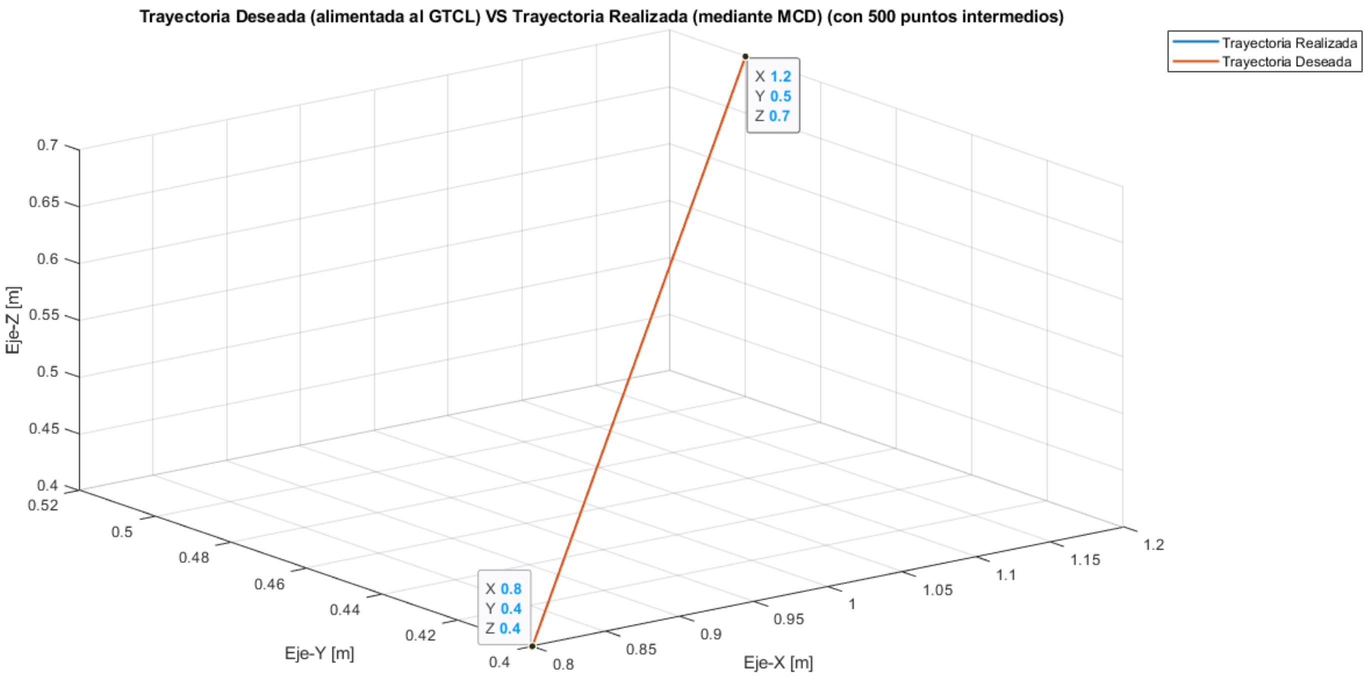
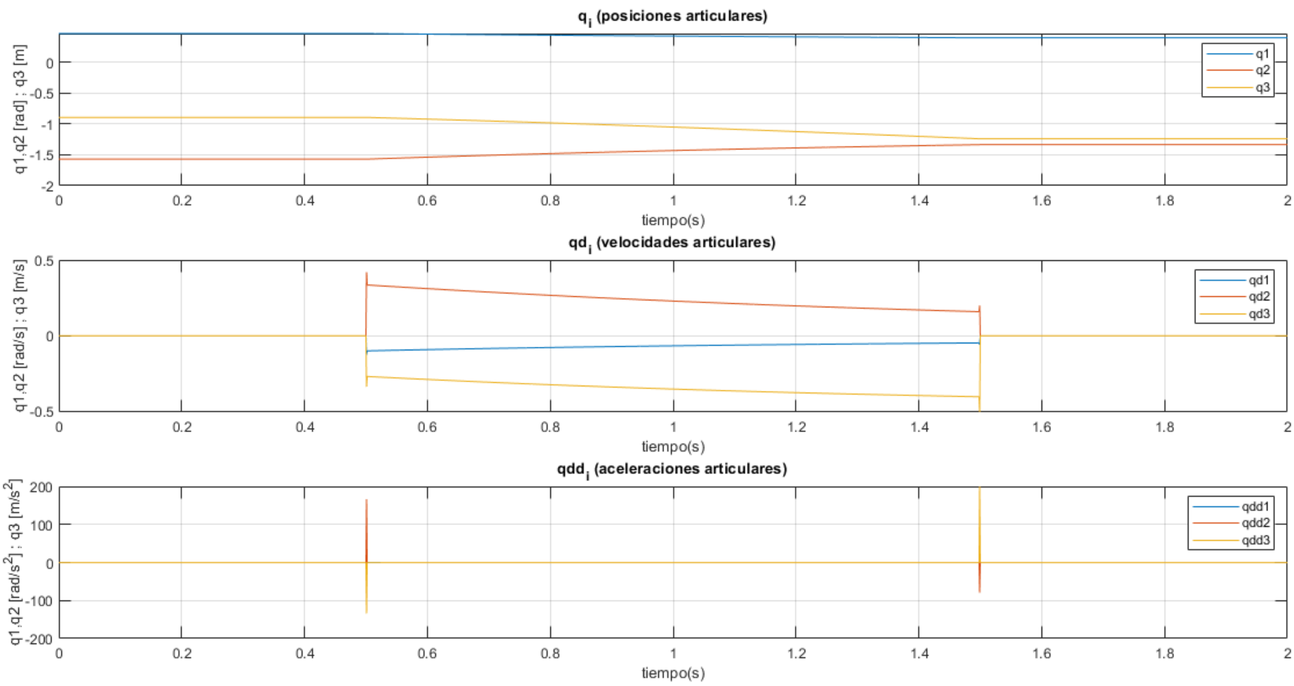


Figura 18: Comparación de trayectorias 3D deseada y realizada para 500 puntos intermedios en el GTCL

Ahora las trayectorias coinciden perfectamente, debido a que se han tomado 500 puntos intermedios, que es una cantidad más razonable para el GTCL. Si se grafican las trayectorias de las variables articulares (q_i , $\dot{q}_i$, $\ddot{q}_i$) para este caso, vemos que se producen picos muy elevados en las aceleraciones, que habría que revisar si son permisibles.



Finalmente se plantea para el caso de la anterior trayectoria con 500 puntos intermedios, para apreciar mejor visualmente la coincidencia de ambas trayectorias mostrar la comparativa en gráficas individuales por ejes “xyz”:

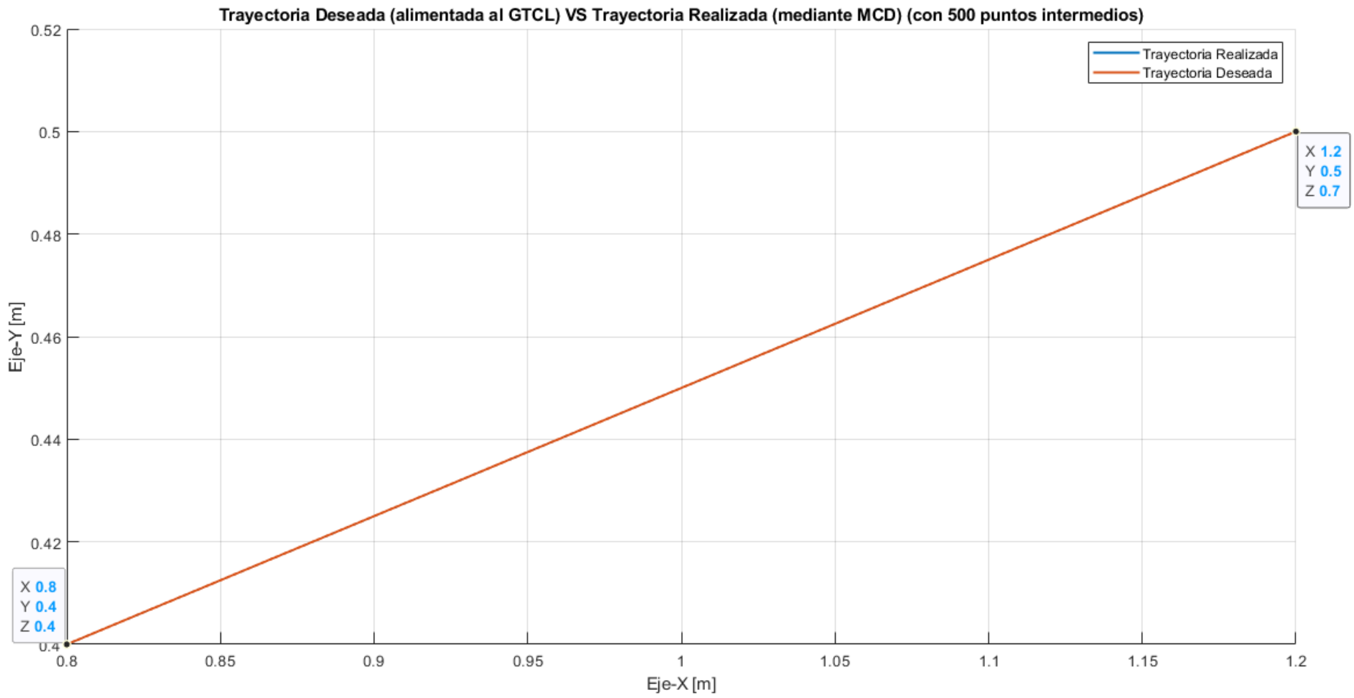


Figura 19: Comparación de trayectorias 2D (Plano XY) deseada y realizada para 500 puntos intermedios en el GTCL

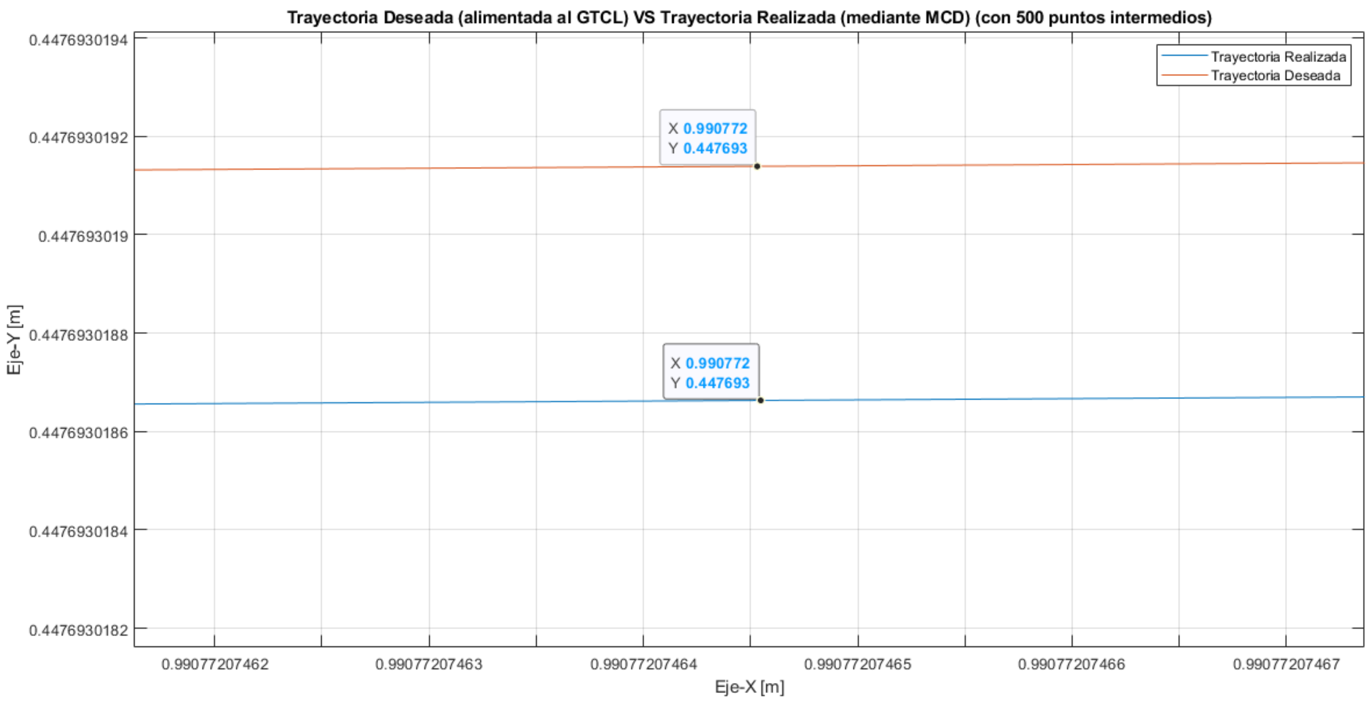


Figura 20: Zoom de la gráfica anterior para mejor visualización de la comparación entre ambas

Como vemos, para este último caso donde comparamos en el plano XY ambas trayectorias desde cerca, vemos que hay que acudir al 10º decimal para poder ser capaces de apreciar la diferencia entre ambas, por lo tanto, consideramos lógicamente despreciable el error, y se da por validado el modelo cinemático directo, así como el inverso.

Para este apartado, los códigos que se han utilizado son los siguientes:

- [CinematicaDirectaGTCL]
- [CinematicaInversaGTCL]
- [GraficasGTCL]

5.- Control dinámico

Hasta ahora, se ha estudiado la cinemática en profundidad, así como el control cinemático, mediante el generador de trayectorias proporcionado. En este apartado lo que se busca es un control de la dinámica, es decir, poder realizar control sobre los pares que aplican los motores a cada articulación a lo largo del tiempo de manera que el robot pueda realizar la trayectoria que se le solicita de manera correcta.

Para esto, se necesita obtener las funciones de transferencia de cada una de las articulaciones, que implica en otras palabras una linealización del modelo.

Partimos primeramente de la ecuación dinámica completa del robot:

$$\tau = (M(q) + R^2 \cdot J_m) \cdot \ddot{q} + (C(q, \dot{q}) + R^2 \cdot B_m) \cdot \dot{q} + G(q) + F(q, \dot{q})$$

$$\tau = M_A(q) \cdot \ddot{q} + V_A(q, \dot{q}) + G_A(q) + F(q, \dot{q})$$

Siendo ambas expresiones equivalentes, y siendo M_A, V_A y G_A, las matrices obtenidas en apartados anteriores del proyecto.

Si ahora expresamos las variables como variables incrementales y linealizamos el modelo en torno a un punto de equilibrio (con $\dot{q}_{eq} = 0$ y $\ddot{q}_{eq} = 0$), nos quedaremos con el siguiente desarrollo al linealizar la siguiente expresión:

$$\tau = M_A(q) \cdot \ddot{q} + V_A(q, \dot{q}) + G_A(q)$$

- Linealización 1:

$$\boxed{M_A(q) \cdot \ddot{q}} \cong \underbrace{M_A(q_{eq}) \cdot \ddot{q}_{eq}}_{0} + \underbrace{\frac{\partial M_A}{\partial q}\bigg|_{q_{eq}} \cdot \ddot{q}_{eq}}_{0} \cdot \Delta q + \boxed{\underbrace{M_A(q_{eq})}_{\text{constante}} \cdot \Delta \ddot{q}_{eq}}$$

- Linealización 2:

$$\boxed{V_A(q, \dot{q})} = C_A(q, \dot{q}) \cdot \dot{q} \cong \underbrace{C_A(q_{eq}, \dot{q}_{eq}) \cdot \dot{q}_{eq}}_{0} + \underbrace{\frac{\partial C_A(q, \dot{q})}{\partial q}\bigg|_{q_{eq}} \cdot \dot{q}_{eq}}_{0} \cdot \Delta q +$$

$$+ \left(\underbrace{\frac{\partial C_A(q, \dot{q})}{\partial q} \cdot \dot{q}_{eq} + C_A(q_{eq}, \dot{q}_{eq})}_{0} \right) \cdot \Delta \dot{q} \cong \boxed{0}$$

- Suposición 3:

Los términos $G_A(q)$ gravitatorios se desprecian y se consideran como perturbaciones en cuanto al diseño de control.

En este momento, es necesario tener en cuenta una serie de consideraciones llevadas a cabo:

- Para la matriz de inercia se van a considerar únicamente las componentes de la diagonal, evaluadas en su caso más desfavorables. Esto se hace debido a que los términos fuera de la diagonal ya de por sí tienen poco peso, pero además el efecto que incluyen las reductoras hace que estos términos sean despreciables respecto de los de la diagonal.
- Los términos gravitatorios se desprecian y se consideran como perturbaciones en cuanto al diseño de control.
- Los términos de Coriolis se desprecian y en la matriz V_A sólo consideraremos los términos $R^2 \cdot B_m$
- Los errores inducidos debidos a estas consideraciones/aproximaciones no son grandes y además pueden quedar absorbidos a través de un correcto diseño del controlador.
- Se desprecian los términos de fricción: $F(q, \dot{q}) \approx 0$

Con estas consideraciones y tras el desarrollo de linealización, nos queda lo siguiente a estudiar:

$$\boxed{\Delta\tau = \left(M_A(diag)|_{eq}\right) \cdot \Delta\ddot{q} + (R^2 \cdot B_m) \cdot \Delta\dot{q}}$$

Esta expresión que está evaluada en el punto de equilibrio se pasará del dominio temporal al dominio de Laplace mediante las transformadas de Laplace para obtener así las deseadas funciones de transferencia de cada articulación.

Por tanto, si recogemos los diferentes elementos para particularizar dicha expresión para mi caso concreto queda lo siguiente (Las expresiones completas de $M_A, V_A\ y\ G_A$ están recogidas en su apartado correspondiente, aquí pondré los valores numéricos procedentes de este desarrollo a partir de dichas matrices).

Para extraer estos valores de Ma, se ha acudido al caso más desfavorable de q2 y q3, donde si q2 = 0 se obtiene el mayor valor de los términos y q3, tomando su valor máximo comentado cuando se estudiaron sus limites de rango sería q3 = 1 y con estos valores queda. Mientras que, para los términos de Coriolis, se acude sólo al termino asociado a la fricción del motor.

$$\begin{bmatrix} \Delta\tau_1 \\ \Delta\tau_2 \\ \Delta\tau_3 \end{bmatrix} = \begin{bmatrix} 13.0997 & 0 & 0 \\ 0 & 41.315 & 0 \\ 0 & 0 & 4.825 \end{bmatrix} \cdot \begin{bmatrix} \Delta\ddot{q}_1 \\ \Delta\ddot{q}_2 \\ \Delta\ddot{q}_3 \end{bmatrix} + \begin{bmatrix} 0.0225 \cdot \Delta\dot{q}_1 \\ 0.0144 \cdot \Delta\dot{q}_2 \\ 0.0225 \cdot \Delta\dot{q}_3 \end{bmatrix}$$

Con esto, podemos desacoplar las ecuaciones:

$$\Delta\tau_1 = 13.0997 \cdot \Delta\ddot{q}_1 + 0.0225 \cdot \Delta\dot{q}_1$$

$$\Delta\tau_2 = 41.315 \cdot \Delta\ddot{q}_2 + 0.0144 \cdot \Delta\dot{q}_2$$

$$\Delta\tau_3 = 4.825 \cdot \Delta\ddot{q}_3 + 0.0225 \cdot \Delta\dot{q}_3$$

Aplicando ahora transformada de Laplace nos queda:

$$\tau_1(s) = s \cdot (13.0997 \cdot s + 0.0225) \cdot q_1(s)$$

$$\tau_2(s) = s \cdot (41.315 \cdot s + 0.0144) \cdot q_2(s)$$

$$\tau_3(s) = s \cdot (4.825 \cdot s + 0.0225) \cdot q_3(s)$$

Reescribiendo como funciones de transferencia tenemos finalmente:

$$G_{11}(s) = \frac{q_1(s)}{\tau_1(s)} = \frac{1}{s \cdot (13.0997 \cdot s + 0.0225)}$$

$$G_{22}(s) = \frac{q_2(s)}{\tau_2(s)} = \frac{1}{s \cdot (41.315 \cdot s + 0.0144)}$$

$$G_{33}(s) = \frac{q_3(s)}{\tau_3(s)} = \frac{1}{s \cdot (4.825 \cdot s + 0.0225)}$$

Ahora ya se puede pasar al diseño de controladores que se piden:

DISEÑO DE CONTROLADOR PD DESCENTRALIZADO:

Gracias a la linealización, se ha logrado desacoplar el modelo, es decir cada componente de τ_i sólo depende de sus correspondientes variables articulares q_i (τ_1 sólo depende de $\dot{q}_1\ y\ \ddot{q}_1$). Esto permite realizar controladores conocidos como descentralizados, es decir, un controlador para cada articulación, que es lo que se hará a continuación.

Como no se especifica si el PD tiene que ser con o sin cancelación de dinámica, voy a optar por la versión con cancelación de dinámica, que es como se realizó en clase.

Planteando la función de transferencia con coeficientes genéricos:

$$G_{ii}(s) = \frac{1}{(a_i \cdot s + b_i) \cdot s}$$

Si sabemos que un controlador tipo PD tiene la siguiente estructura:

"Forma teórica": $C_{ii}(s) = K_{c_i} \cdot (1 + T_{d_i} \cdot s)$

"Forma Simulink": $C_{ii}(s) = K_{P_i} + K_{D_i} \cdot s$

Omitiendo el desarrollo visto en clase nos quedan una serie de expresiones:

$$\begin{cases} K_{c_i} = \dfrac{3 \cdot b_i}{t_s} \\[2ex] T_{d_i} = \dfrac{a_i}{b_i} \end{cases}$$

Aquí tenemos 1 parámetro de diseño libre, t_s. Con esto sólo nos queda determinar el tiempo de subida, para ello por diseño podemos elegirlo para que sea más o menos agresivo el control, podemos empezar con $\boldsymbol{t_s = 0.01s}$.

Utilizando el siguiente código: ControladoresPD, particularizamos los parámetros de cada controlador para cada articulación.

- **Forma "Teórica":**

$K_{c_1} = 6.75$; $K_{c_2} = 4.32$; $K_{c_3} = 6.75$

$T_{d_1} = 582.208$; $T_{d_2} = 2869.097$; $T_{d_3} = 214.444$

$$C_{11}(s) = 6.75 \cdot (1 + 582.208 \cdot s)$$

$$C_{22}(s) = 4.32 \cdot (1 + 2869.097 \cdot s)$$

$$C_{33}(s) = 6.75 \cdot (1 + 214.444 \cdot s)$$

- **Forma "Simulink":**

$K_{P_1} = 6.75$; $K_{P_2} = 4.32$; $K_{P_3} = 6.75$

$K_{D_1} = 3929.91$; $K_{D_2} = 12394.5$; $K_{D_3} = 1447.5$

$$C_{11}(s) = 6.75 + 3929.91 \cdot s$$

$$C_{22}(s) = 4.32 + 12394.5 \cdot s$$

$$C_{33}(s) = 6.75 + 1447.5 \cdot s$$

Tras estos cálculos, se implementa el controlador en Simulink, lógicamente esta última versión, "Forma Simulink", para esta implementación, se pide que se genere una trayectoria de referencia de manera que el robot parta desde la posición HOME, y que termine en un punto situado en un **incremento** de coordenadas cartesianas $(\Delta X, \Delta Y, \Delta Z) = (-0.15, -0.15, 0.15)m$

Además, se pide que se analicen resultados para distintas velocidades de movimiento, modificando esto en Simulink con "duración del movimiento".

Por tanto, las posiciones inicial y final del efector final para cumplir lo que se pide serán:

$$P_{ini(HOME)} = [0 \quad 1 \quad 1.2] \quad y \quad P_{fin} = [-0.15 \quad 0.85 \quad 1.35]$$

Siendo esta posición HOME la de mi robot "C2" en su "posición de dibujo".

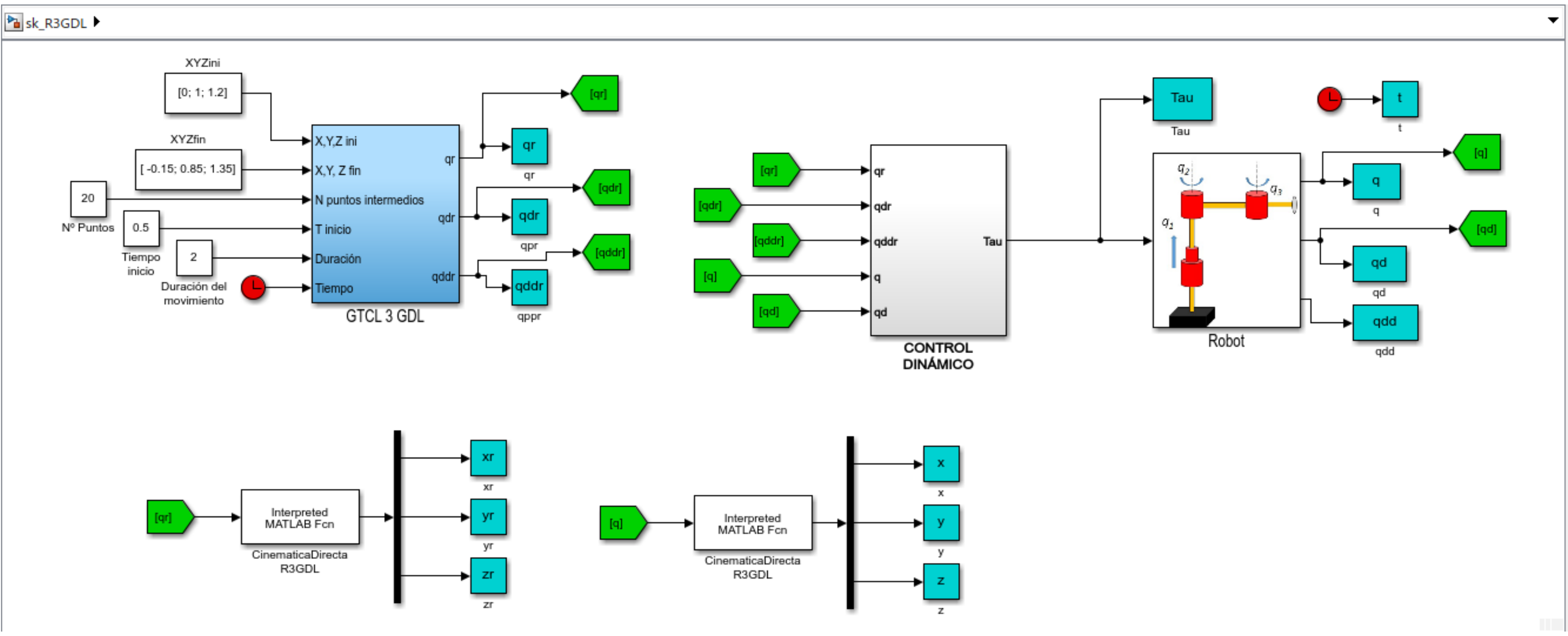


Figura 21: Implementación general en Simulink del sistema conjunto

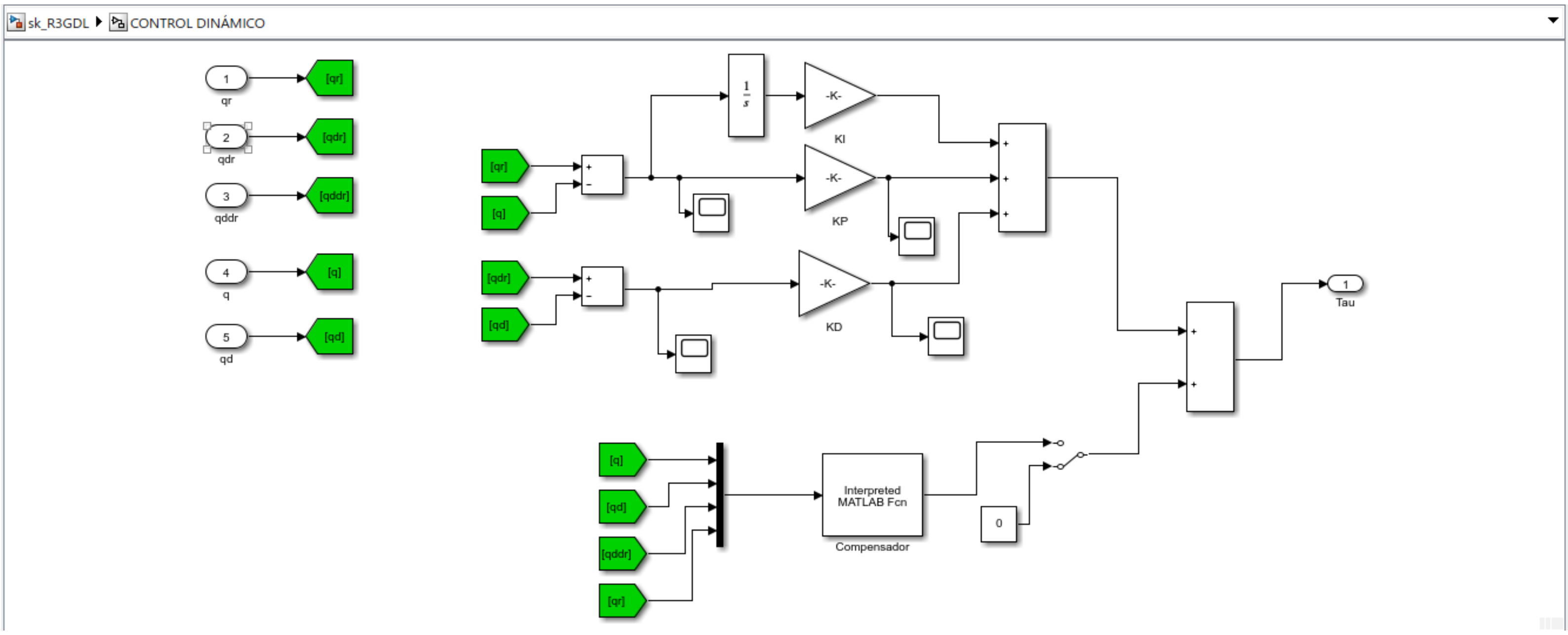
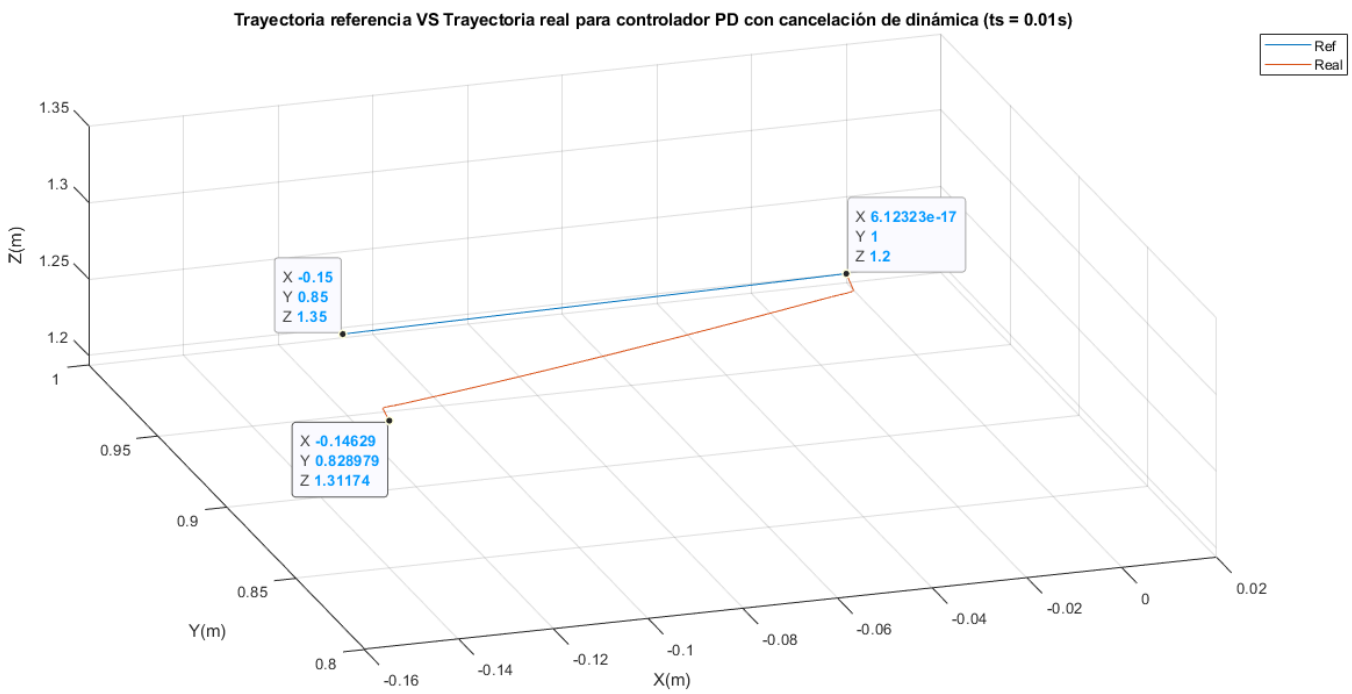


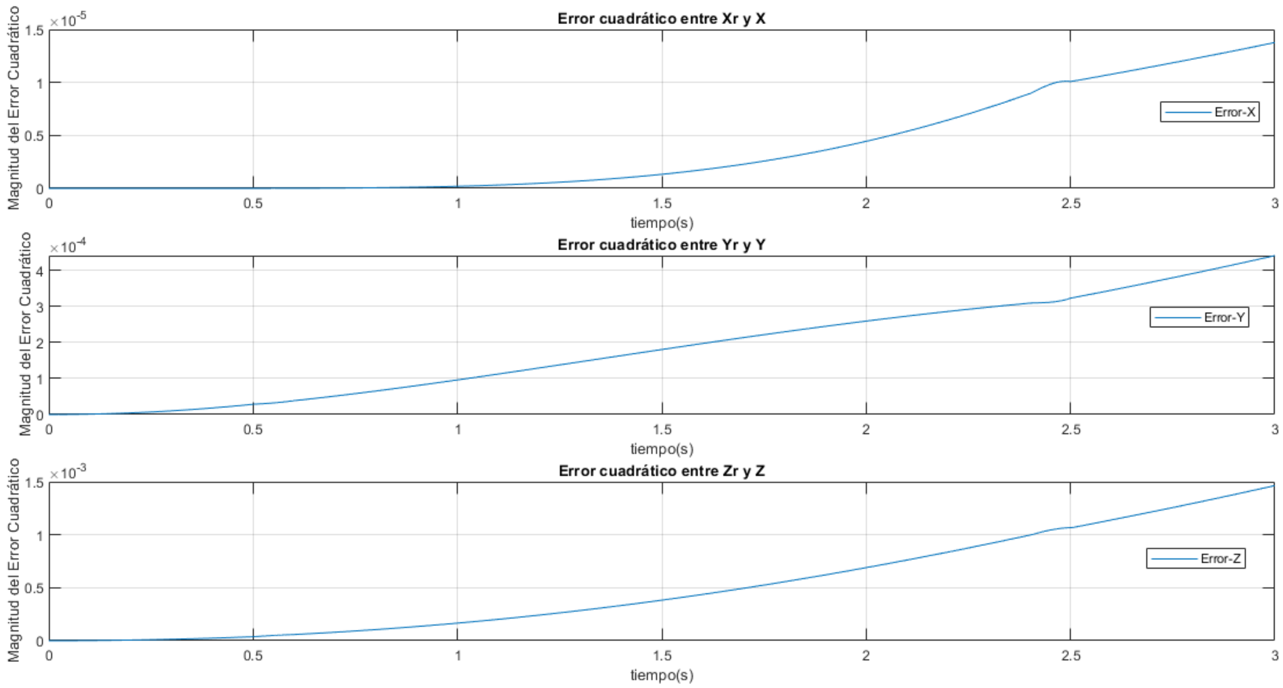
Figura 22: Implementación en Simulink del controlador PD (con opción para incluir compensador de gravedad)

Como vemos, aunque en la figura anterior está preparada para un controlador PID, sólo se alimentan los parámetros KP y KD calculados anteriormente (KI se anulaba en el código)

Podemos ya por fin recoger resultados de dicho controlador PD.

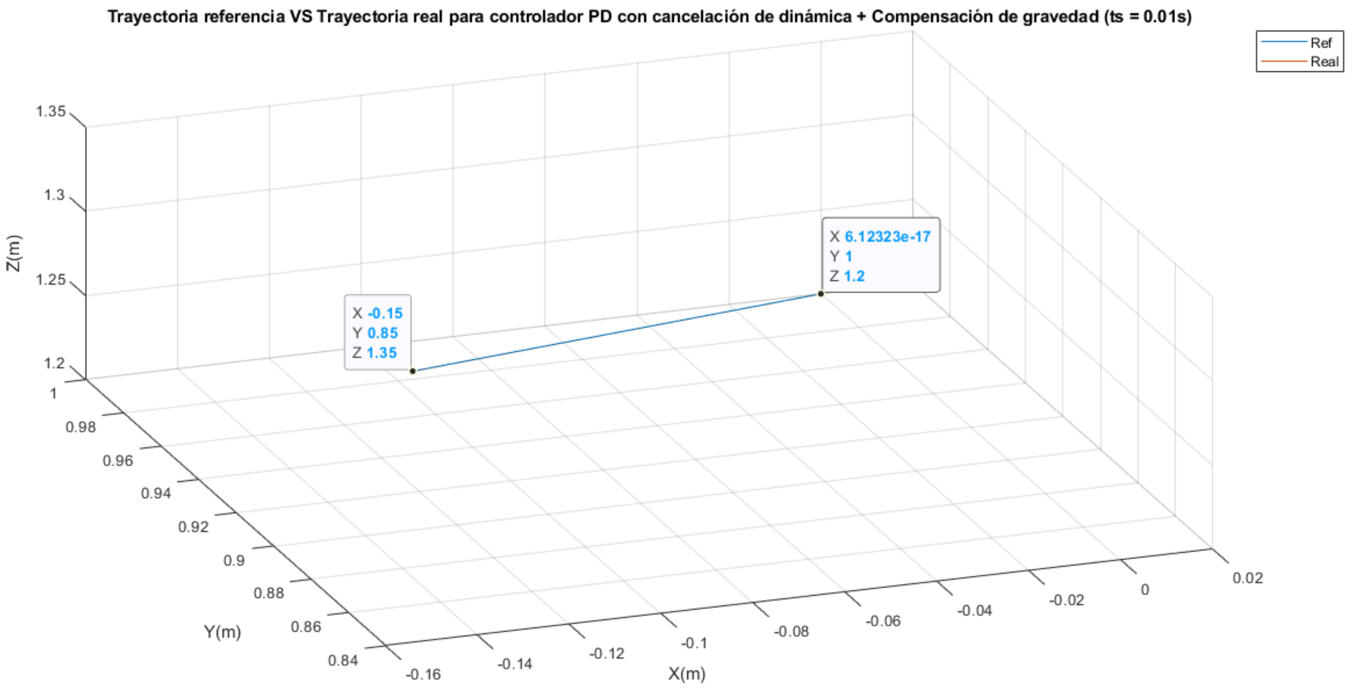
- **Resultados para 20 puntos intermedios en el GTCL y duración del movimiento de 2s (entre t=0.5s y t = 2.5s):**

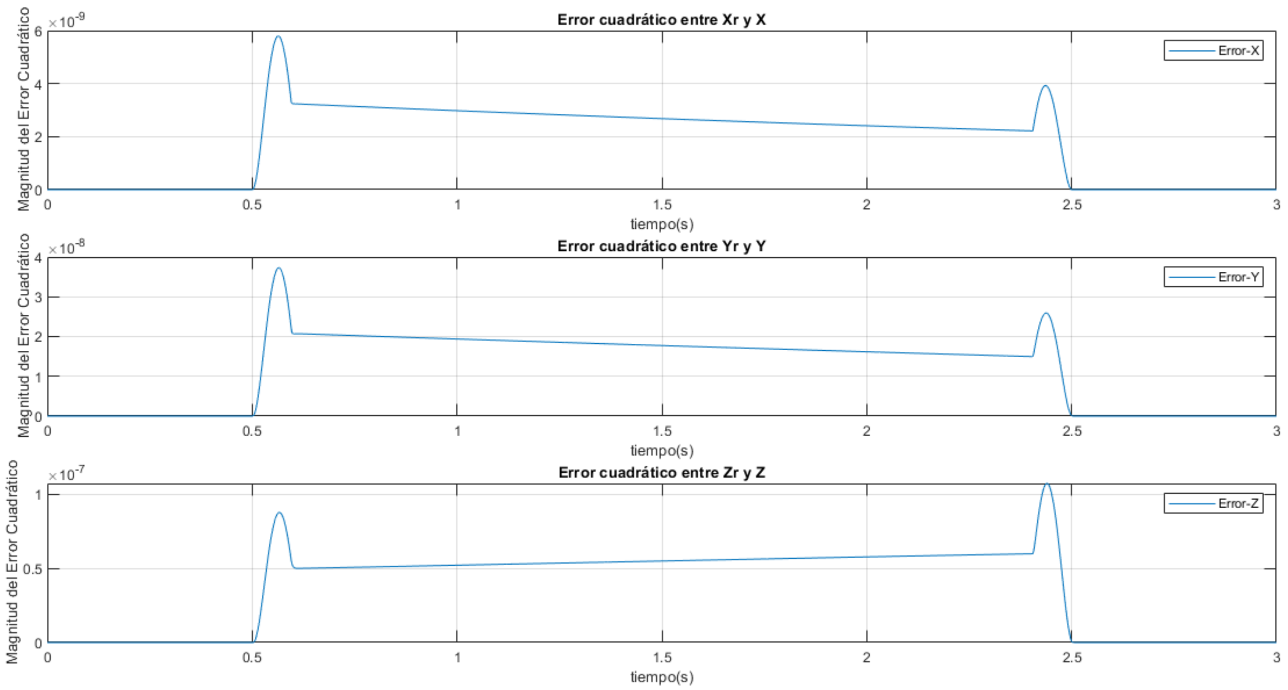




Vemos que el error en coordenadas cartesianas acumulado por el PD es notable y además creciente, posiblemente debido a grandes efectos de la gravedad al introducir la misma como perturbación y no haberla tenido en cuenta para el diseño de control.

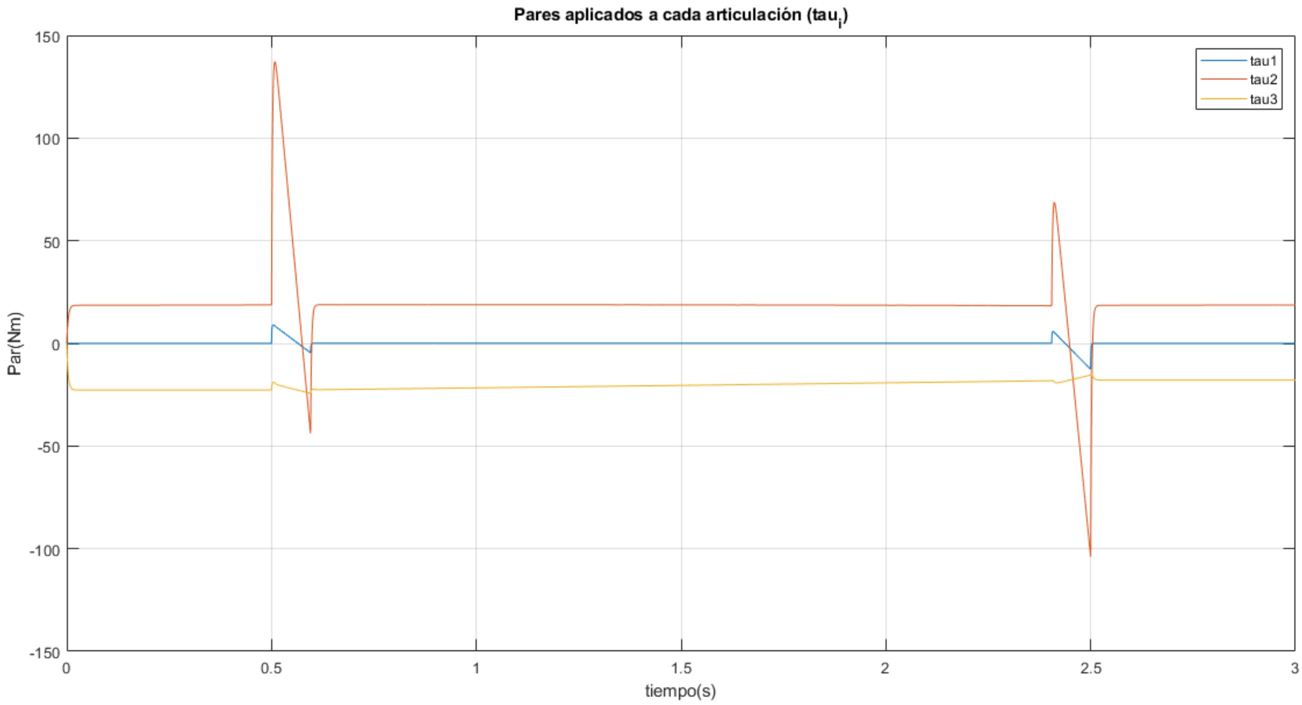
Podemos confirmar que esto es así si incluimos el compensador de gravedad y volvemos a graficar:



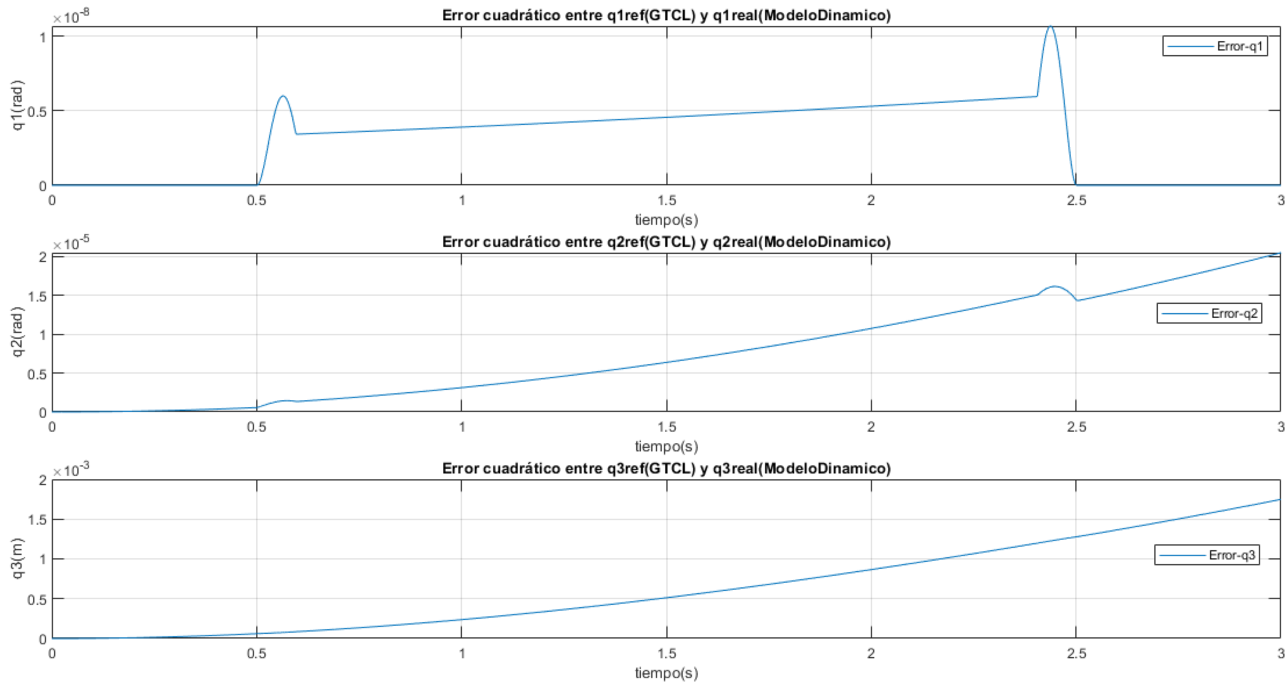


Como vemos, al incluir esta compensación de la gravedad, ambas trayectorias coinciden y vemos en la gráfica de los errores en cartesianas, que, aún siendo un PD, el error ya no es creciente y además es mucho menor y, por tanto, despreciable respecto del error que se tenía antes de esta compensación de gravedad.

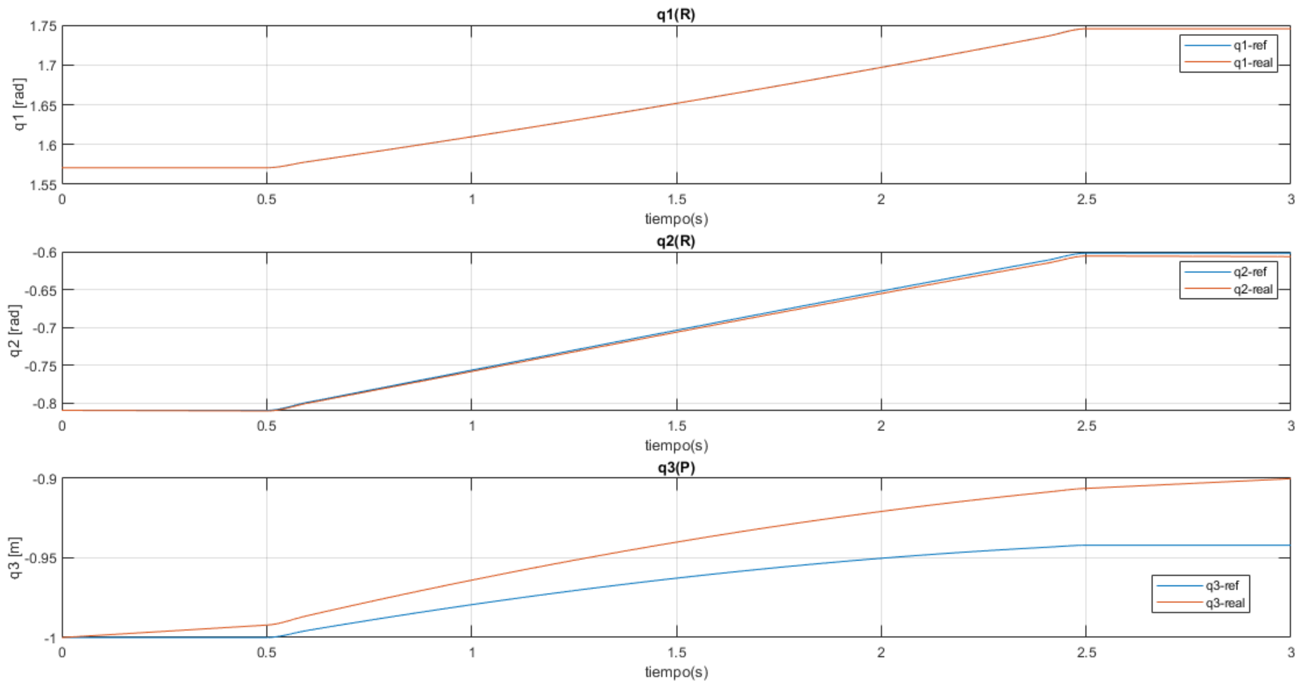
Tras haber comparado el PD con y sin compensación de gravedad, se siguen mostrando resultados para el PD normal (sin compensación):

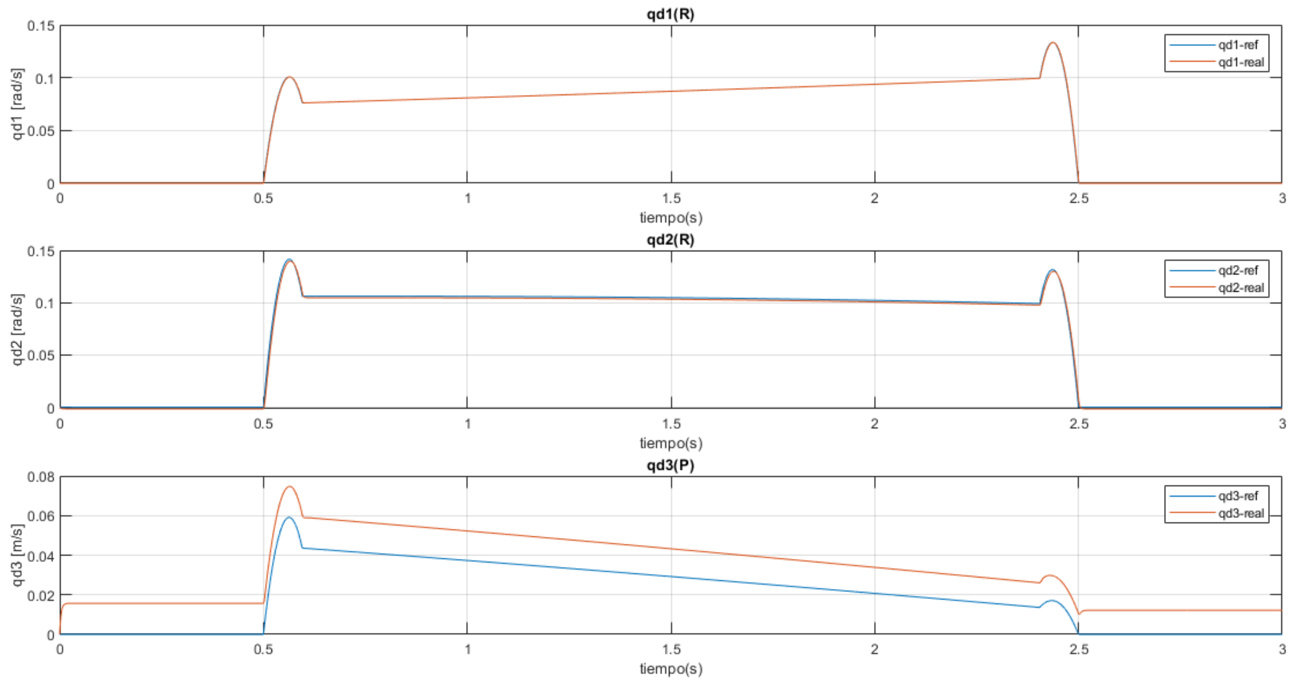


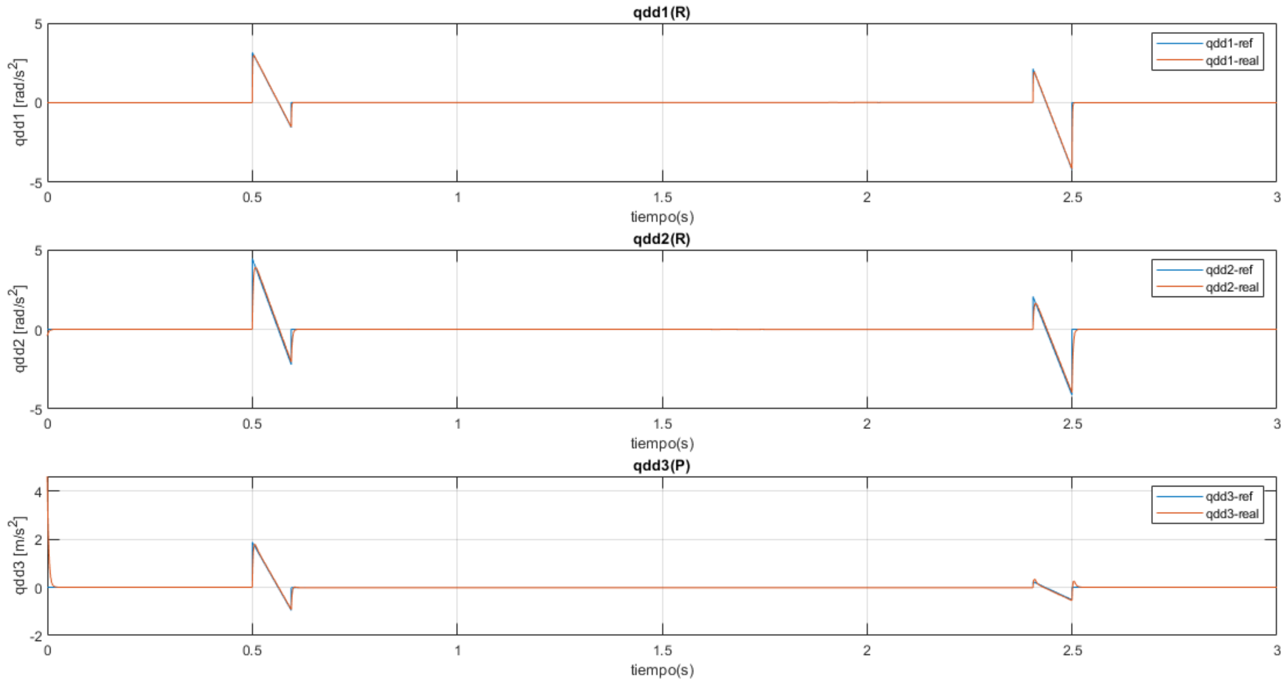
Apreciamos que debido a la agresividad del controlador (ts = 0.01s), hay picos intensos de par.



Al igual que vimos en los errores en cartesianas, para las variables articulares (qi-ref VS qi-real) se aprecia como el error es creciente para las articulaciones q2 y q3 que son las que experimentan la influencia de la gravedad y por tanto entra como perturbación en dichas articulaciones, y al ser PD tendremos como sabemos error en régimen permanente no nulo y en este caso creciente con el tiempo.

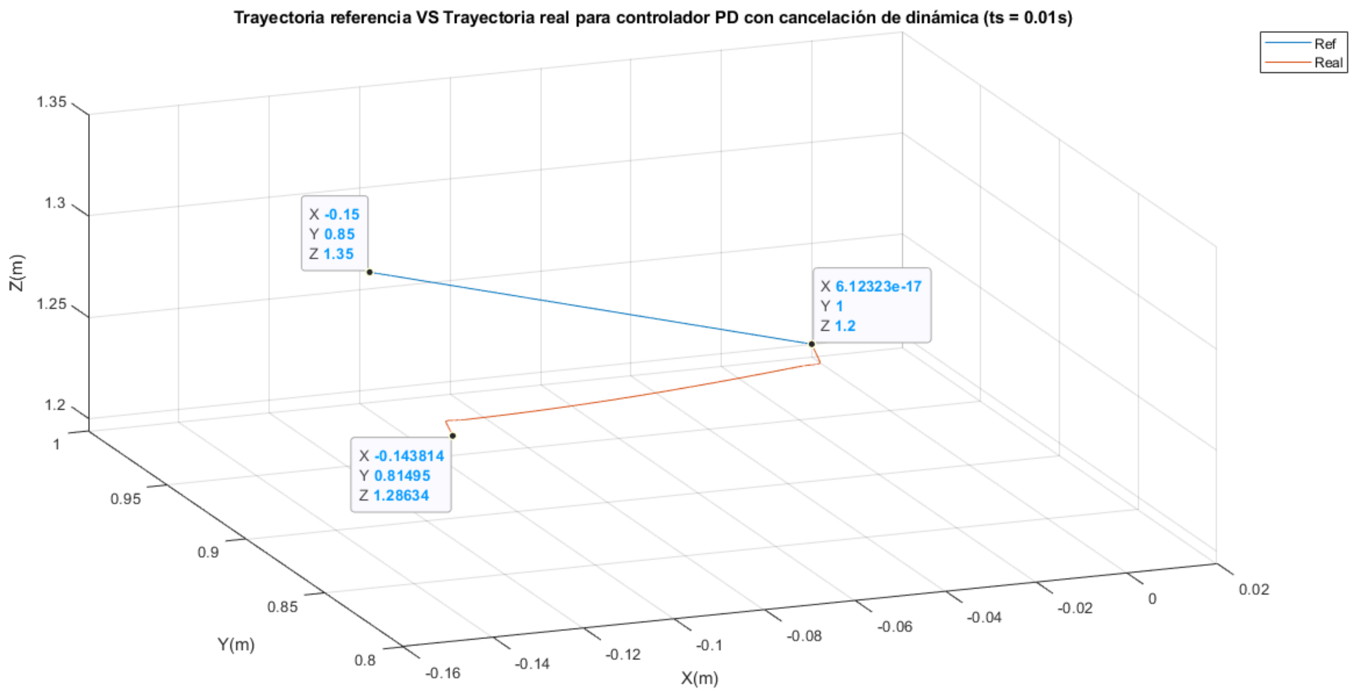


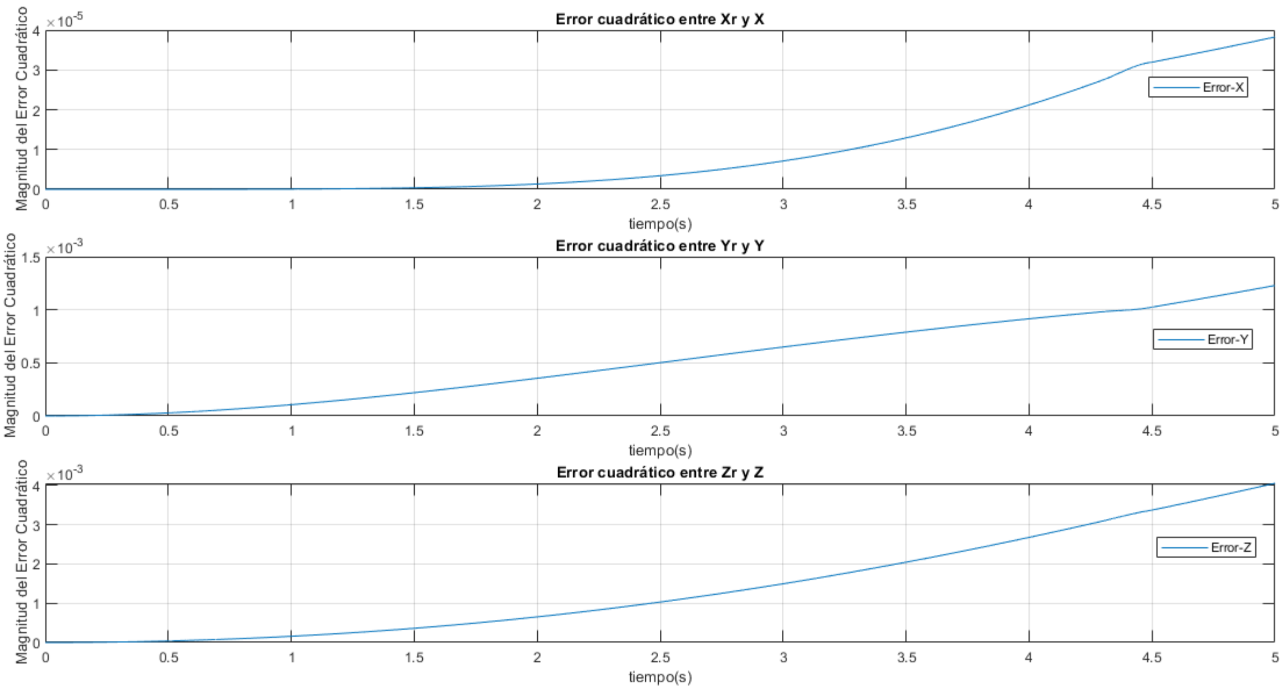




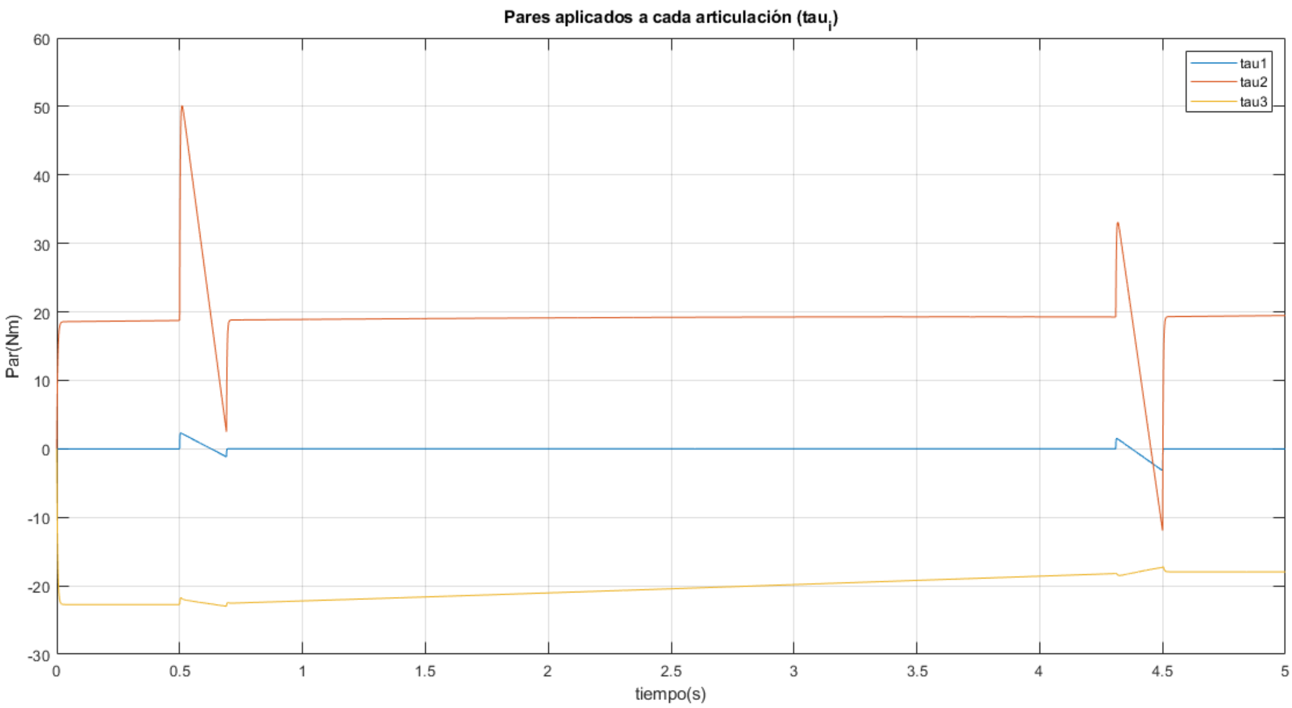
Comprobamos en estas gráficas como es en la tercera articulación (prismática) la que mayor error acumula, como esperábamos de las gráficas de errores entre qi-ref y qi-real.

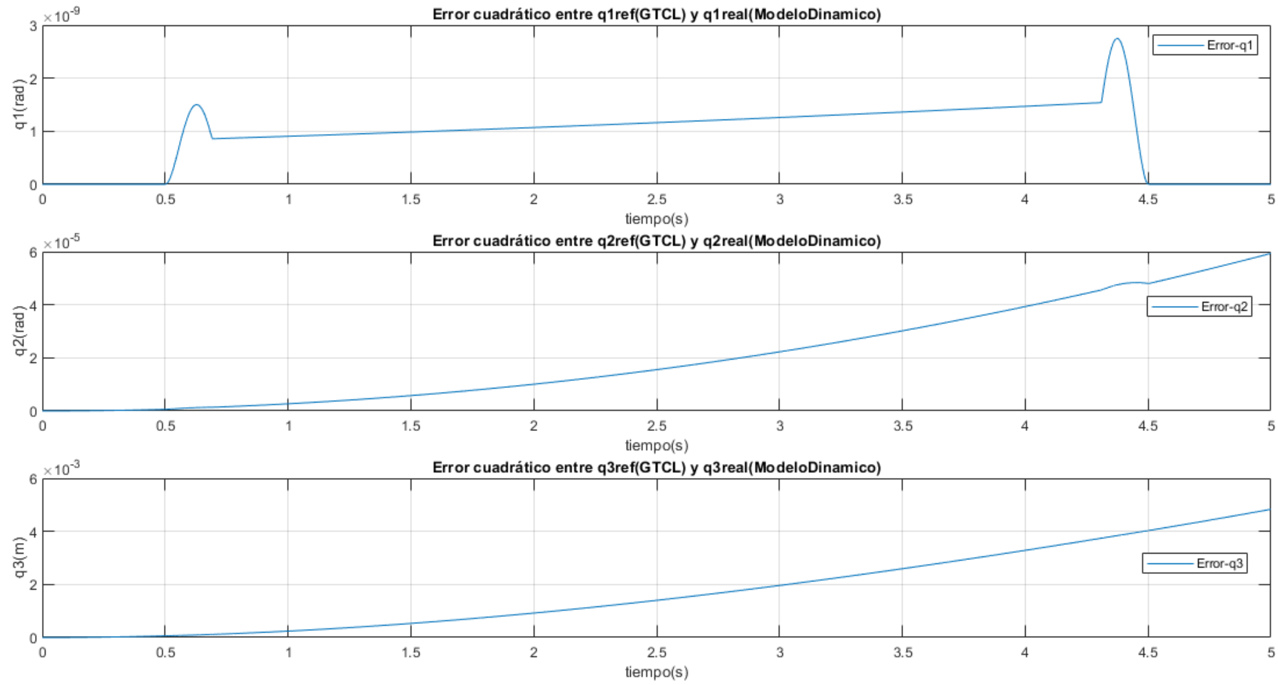
- **Resultados para 20 puntos intermedios en el GTCL y duración del movimiento de 4s (entre t=0.5s y t = 4.5s) (No se va a repetir la comparativa de compensación de gravedad):**



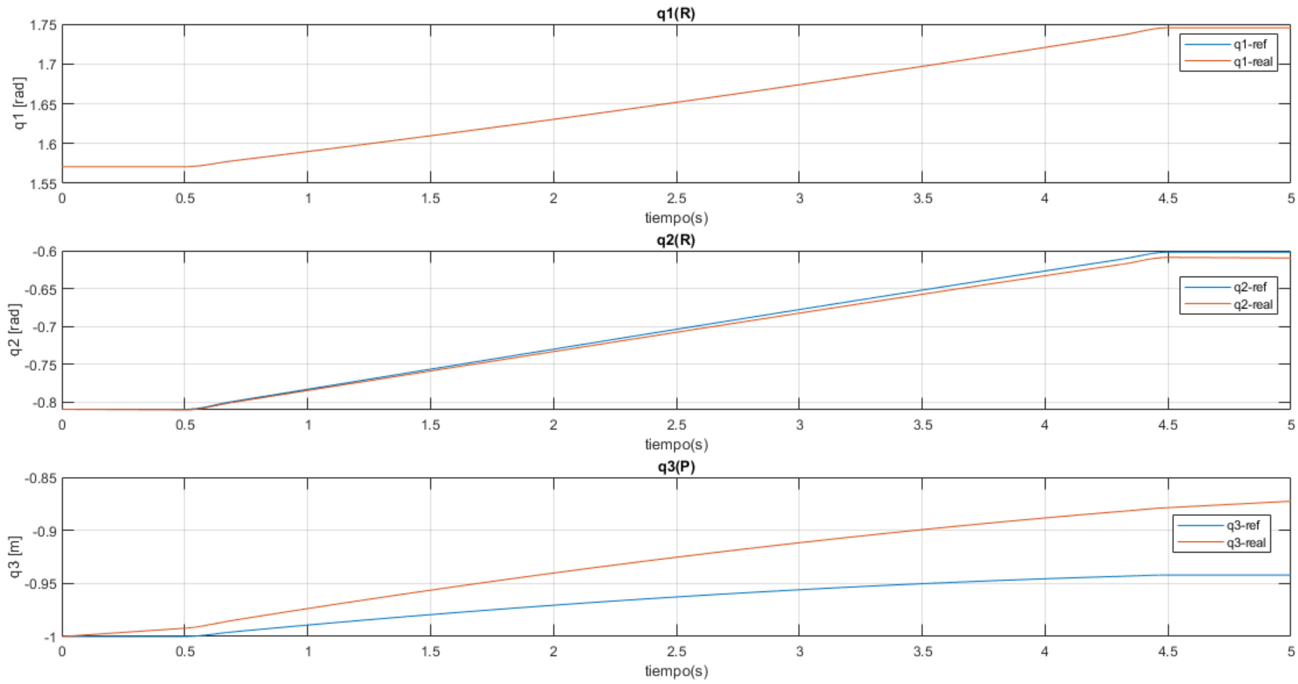


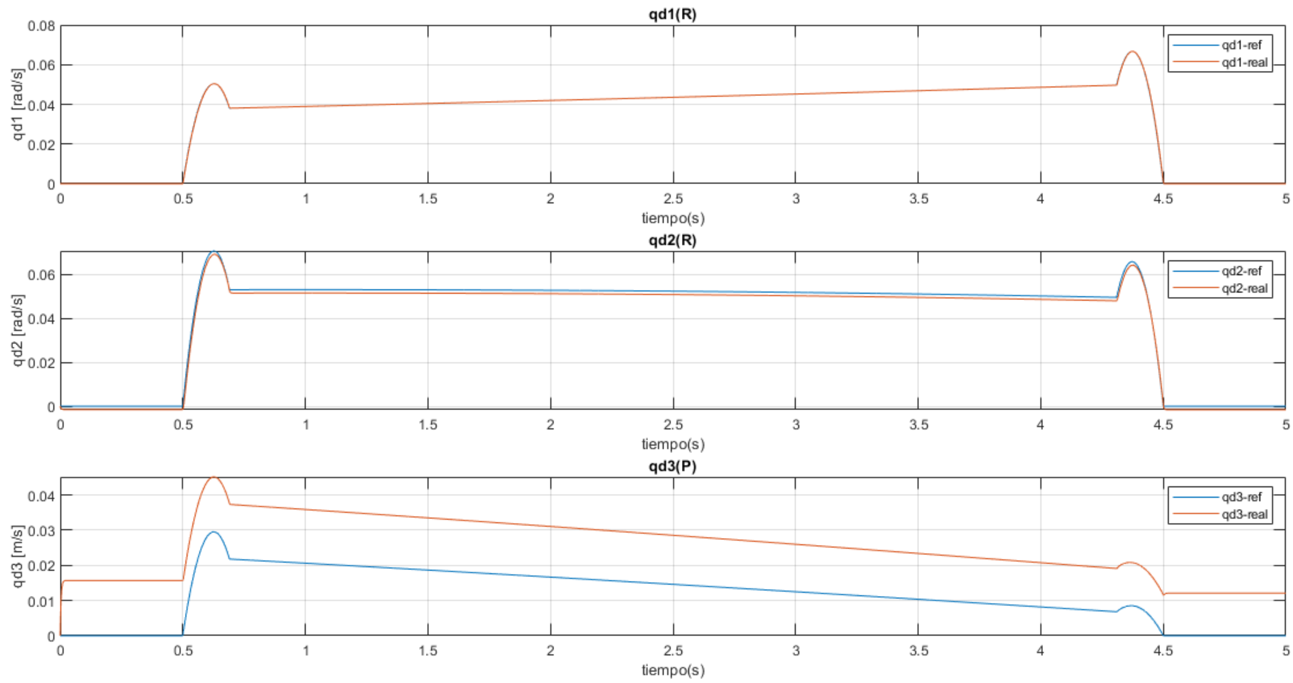
Comprobamos que, al aumentar la duración del movimiento, el error acumulado lógicamente es mayor que en el caso anterior, ya que, al ser el error creciente en el tiempo, al ser más prolongado el movimiento, acumula más error.

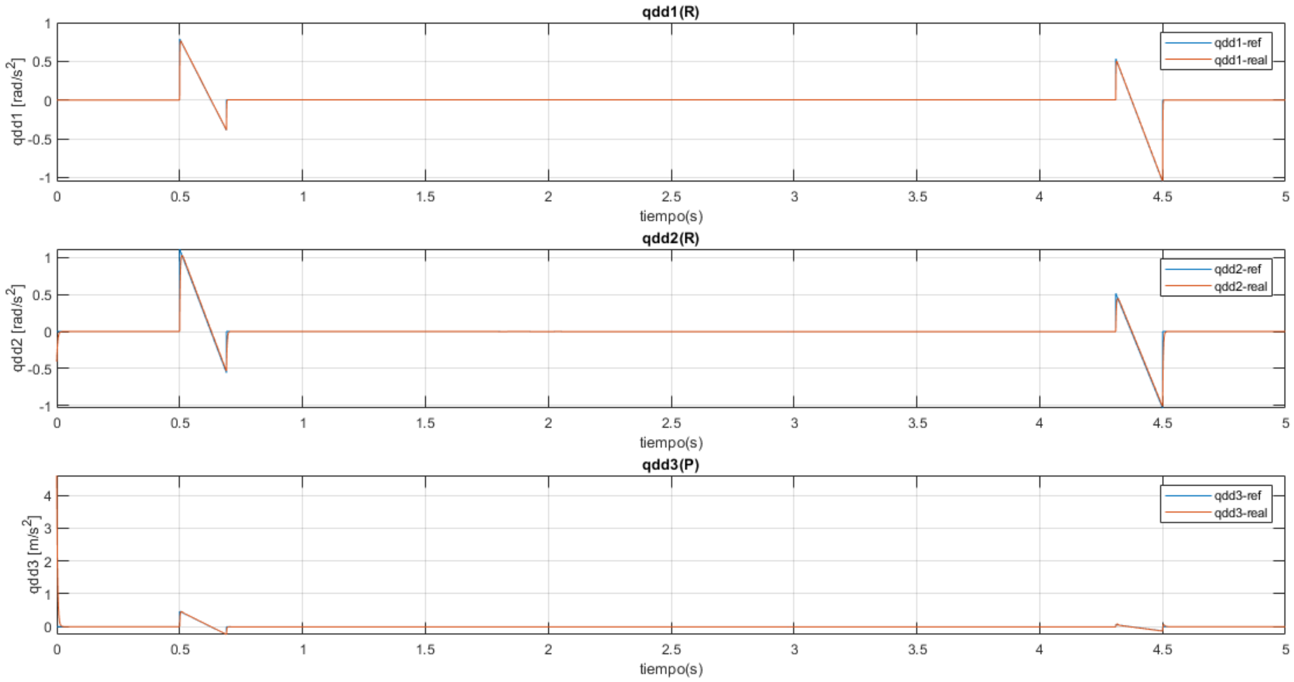




Las conclusiones para estas gráficas son bastante similares que las comentadas en el apartado anterior (2s de duración), no merece la pena repetirlo.


q1(R)
q1 [rad]
q1-ref
q1-real
tiempo(s)
q2(R)
q2 [rad]
q2-ref
q2-real
tiempo(s)
q3(P)
q3 [m]
q3-ref
q3-real
tiempo(s)


qd1(R)
qd1 [rad/s]
qd1-ref
qd1-real
tiempo(s)
qd2(R)
qd2 [rad/s]
qd2-ref
qd2-real
tiempo(s)
qd3(P)
qd3 [m/s]
qd3-ref
qd3-real
tiempo(s)



Se observa un comportamiento similar que para el caso de duración de 2s, notando picos más suaves en cuanto a pares y aceleraciones.

DISEÑO DE CONTROLADOR PID DESCENTRALIZADO:

Planteando la función de transferencia con coeficientes genéricos:

$$G_{ii}(s) = \frac{1}{(a_i \cdot s + b_i) \cdot s}$$

Si sabemos que un controlador tipo PID tiene la siguiente estructura:

"Forma teórica": $$C_{ii}(s) = K_{c_i} \cdot \frac{(1+\tau_{1_i} \cdot s) \cdot (1+\tau_{2_i} \cdot s)}{s}$$

Donde deberemos particularizar $K_{c_i}, \tau_{1_i}\ y\ \tau_{2_i}$, que posteriormente se "convertirán" a los parámetros necesarios para su correcta implementación en Simulink (K_{P_i}, K_{D_i}, K_{I_i}).

Omitiendo el desarrollo visto en clase nos quedan una serie de expresiones:

$$\tau_{1_i} = \frac{a_i}{b_i}$$

1

$$\tau_{2_i} = \frac{2 \cdot \delta}{\omega_n} = \frac{2 \cdot t_s}{6} = \frac{t_s}{3}$$

$$\boxed{t_s = \frac{6}{\omega_n}}$$

$$K_{c_i} = b_i \cdot \omega_n^2 = b_i \cdot \frac{36}{t_s^2}$$

Aquí tenemos 1 parámetro de diseño libre, t_s. Con esto sólo nos queda determinar el tiempo de subida, para ello por diseño podemos elegirlo para que sea más o menos agresivo el control, podemos empezar con $\boldsymbol{t_s = 0.01s}$.

Utilizando el siguiente código: ControladoresPID, particularizamos los parámetros de cada controlador para cada articulación.

- **Forma "Teórica":**

$K_{c_1} = 8100$; $K_{c_2} = 5184$; $K_{c_3} = 8100$

$\tau_{1_1} = 582.208$; $\tau_{1_2} = 2869.097$; $\tau_{1_3} = 214.444$

$\tau_{2_1} = 0.003333$; $\tau_{2_2} = 0.003333$; $\tau_{2_3} = 0.003333$

$$C_{11}(s) = 8100 \cdot \frac{(1 + 582.208 \cdot s) \cdot (1 + 0.003333 \cdot s)}{s}$$

$$C_{22}(s) = 5184 \cdot \frac{(1 + 2869.097 \cdot s) \cdot (1 + 0.003333 \cdot s)}{s}$$

$$C_{33}(s) = 8100 \cdot \frac{(1 + 214.444 \cdot s) \cdot (1 + 0.003333 \cdot s)}{s}$$

- **Parámetros "Simulink":**

$K_{P_1} = 4715919$; $K_{P_2} = 14873417.28$; $K_{P_3} = 1737027$

$K_{D_1} = 15719.64$; $K_{D_2} = 49578$; $K_{D_3} = 5790$

$K_{I_1} = 8100$; $K_{I_2} = 5184$; $K_{I_3} = 8100$

Tras estos cálculos, se implementa el controlador en Simulink, lógicamente esta última versión, "Forma Simulink", para esta implementación, se pide que se genere una trayectoria de referencia de manera que el robot parta desde la posición HOME, y que termine en un punto situado en un **incremento** de coordenadas cartesianas $(\Delta X, \Delta Y, \Delta Z) = (-0.15, -0.15, 0.15)m$

Además, se pide que se analicen resultados para distintas velocidades de movimiento, modificando esto en Simulink con "duración del movimiento".

Por tanto, las posiciones inicial y final del efector final para cumplir lo que se pide serán:

$$P_{ini(HOME)} = [0 \quad 1 \quad 1.2] \quad y \quad P_{fin} = [-0.15 \quad 0.85 \quad 1.35]$$

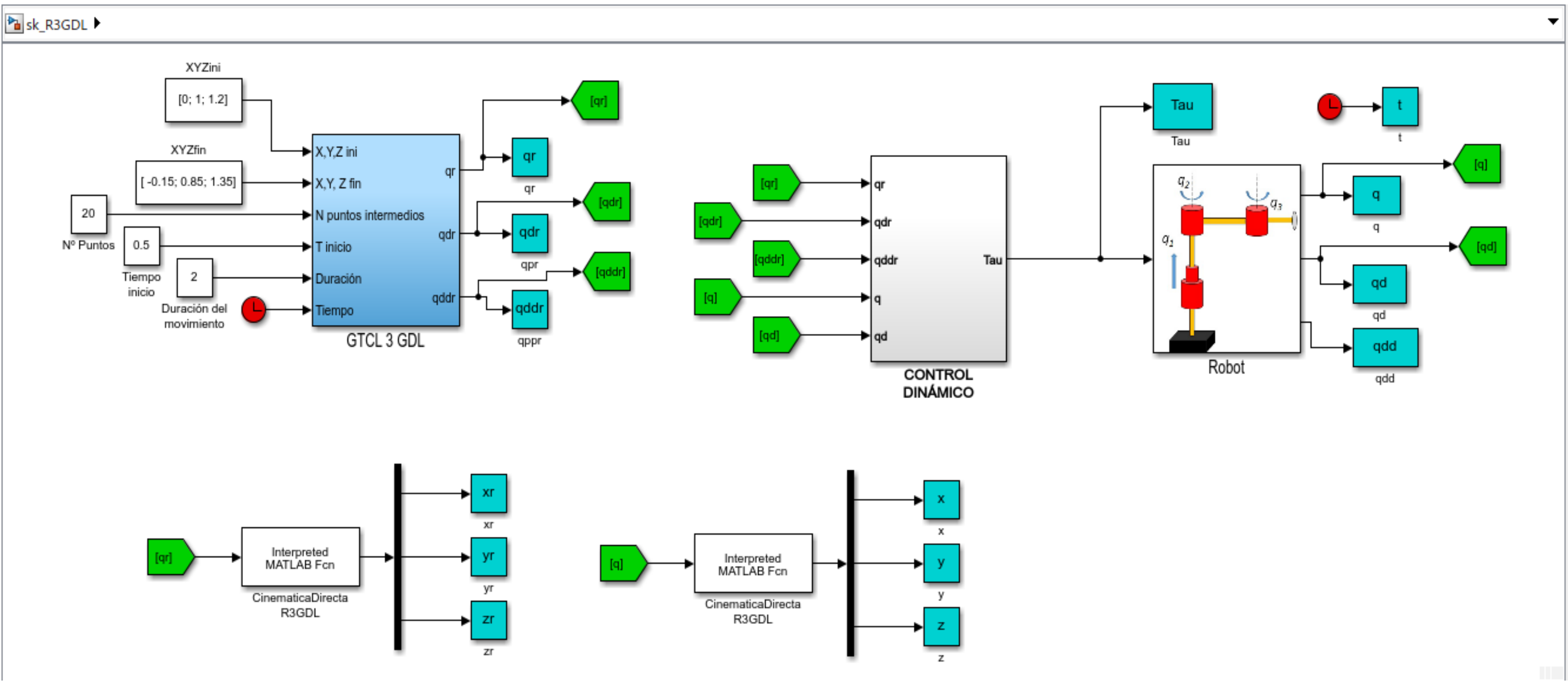


Figura 23: Implementación general en Simulink del sistema conjunto

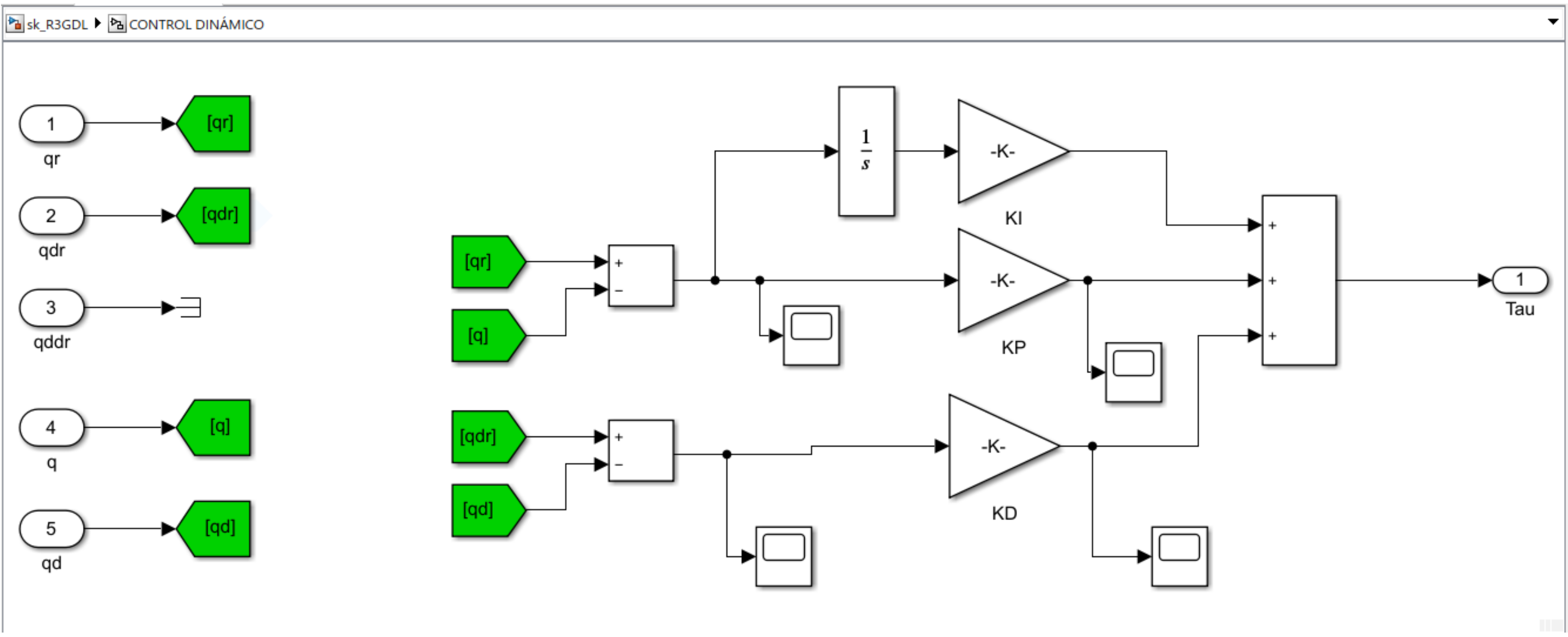
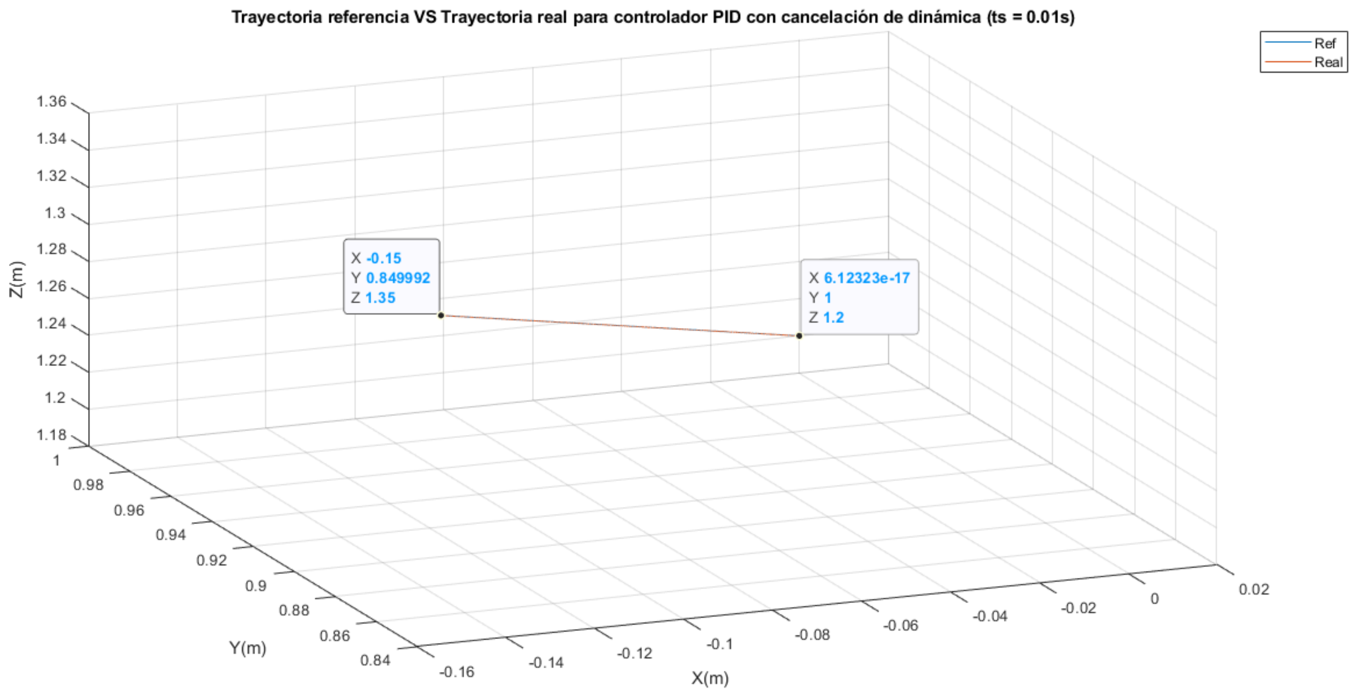


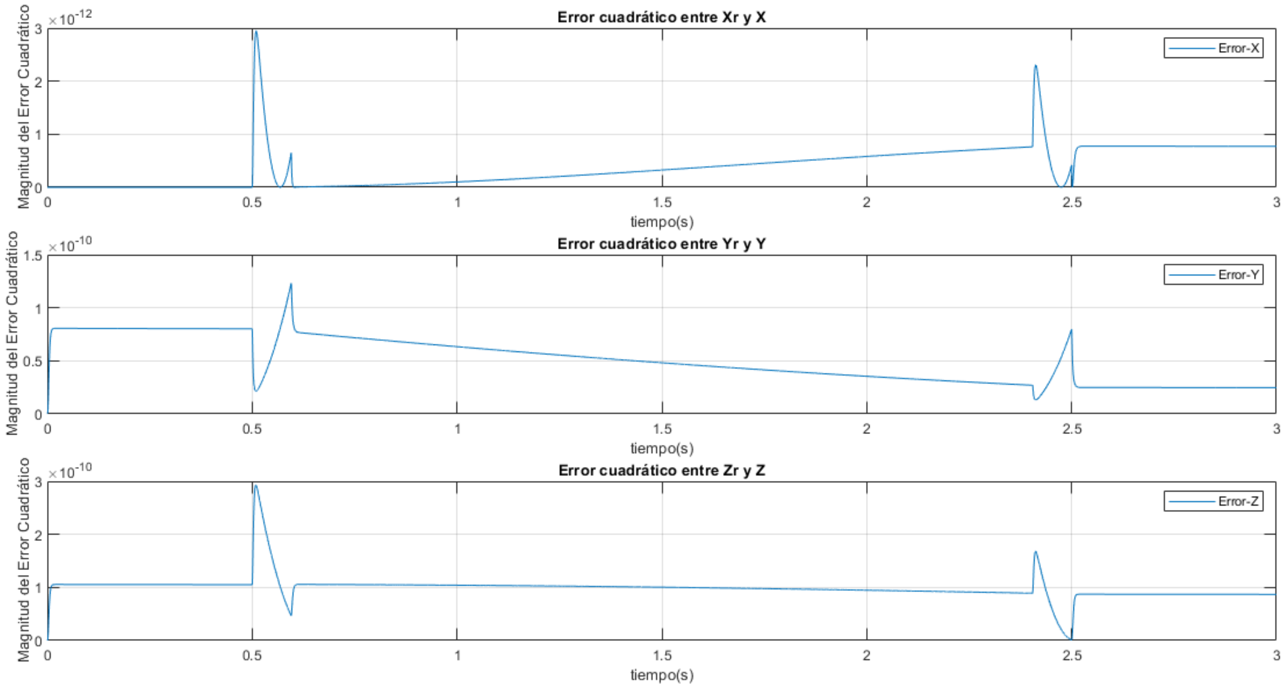
Figura 24: Implementación en Simulink del controlador PID

Como vemos, el esquema es el mismo que para el control tipo PD, sin embargo, para el PD, se anulaban los términos integrales (Ki), sin embargo, ahora, como hemos visto tenemos estos términos presentes al tratarse de un control tipo PID y por tanto entra en juego. De esta forma obtenemos de manera bastante similar en cuanto a estructura de Simulink el control tipo PID.

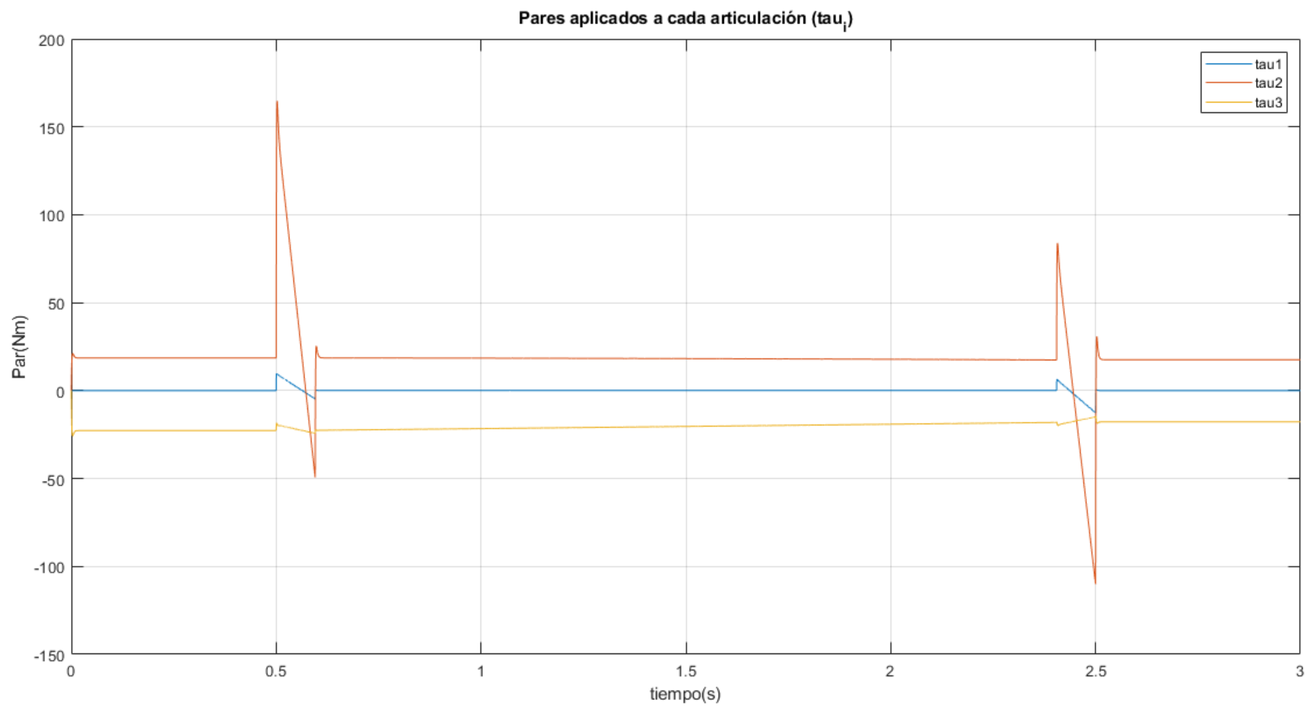
Podemos ya por fin recoger resultados de dicho controlador PID.

- **Resultados para 20 puntos intermedios en el GTCL y duración del movimiento de 2s (entre t=0.5s y t = 2.5s):**

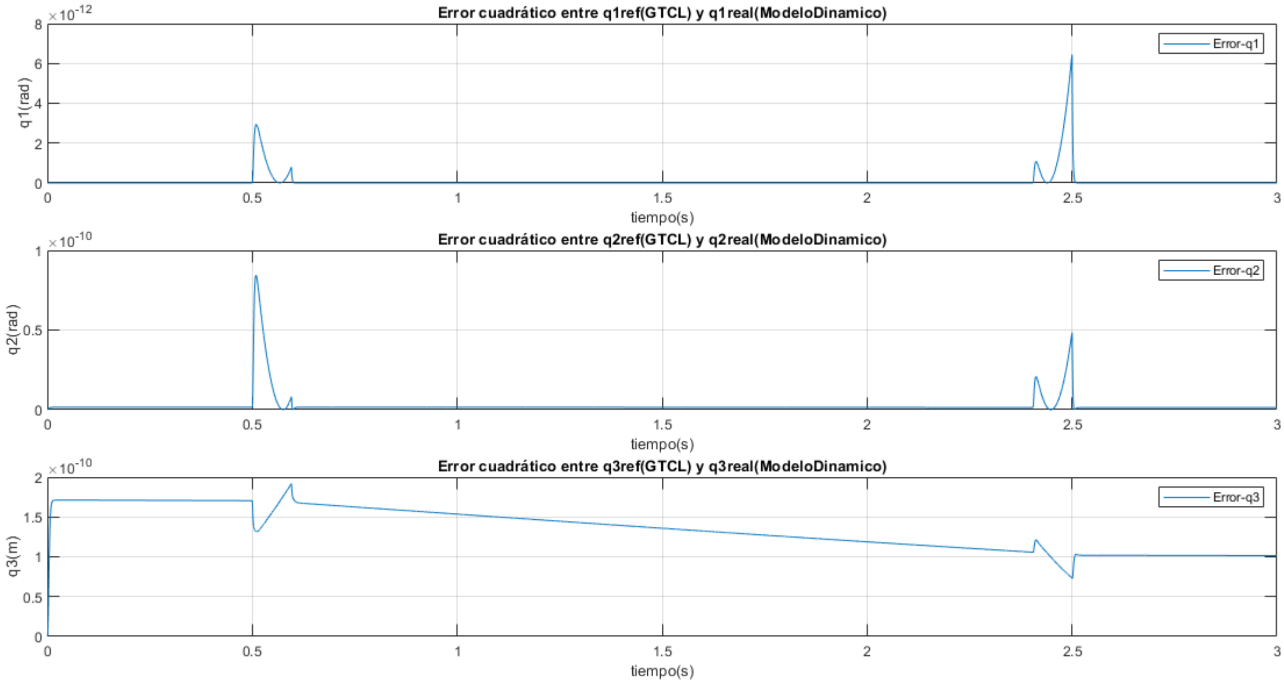




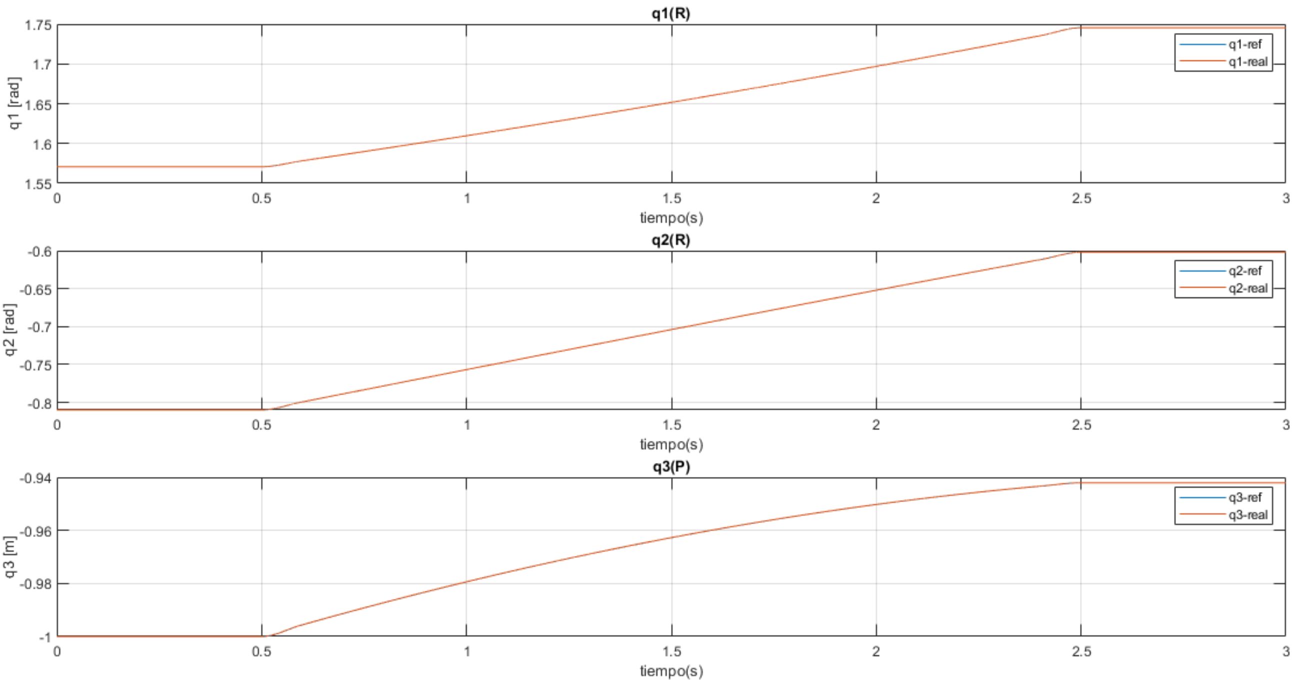
Ahora vemos como al implementar un control tipo PID, si que coinciden perfectamente la trayectoria referencia y la que realiza el robot realmente. Esto se aprecia con mayor detalle en esta gráfica de los errores cuadráticos, donde vemos que el error en régimen permanente es despreciable y ya no es creciente, sino que se corrige gracias al término integral.

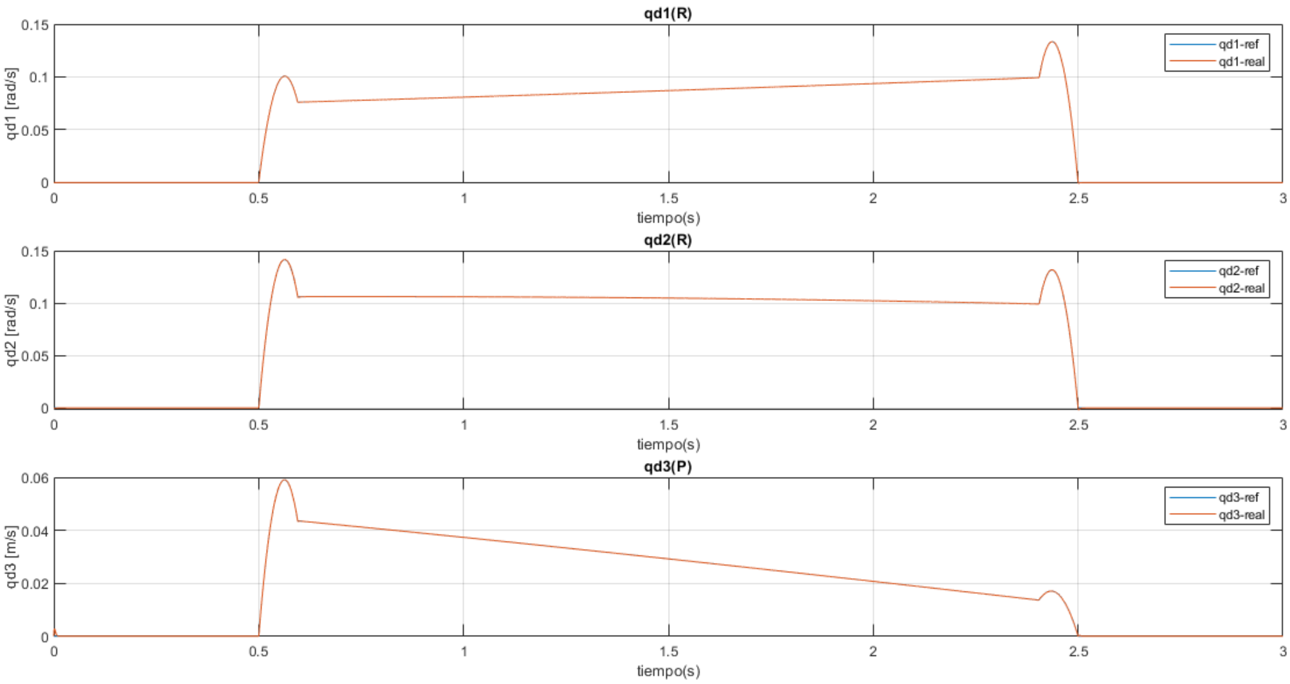


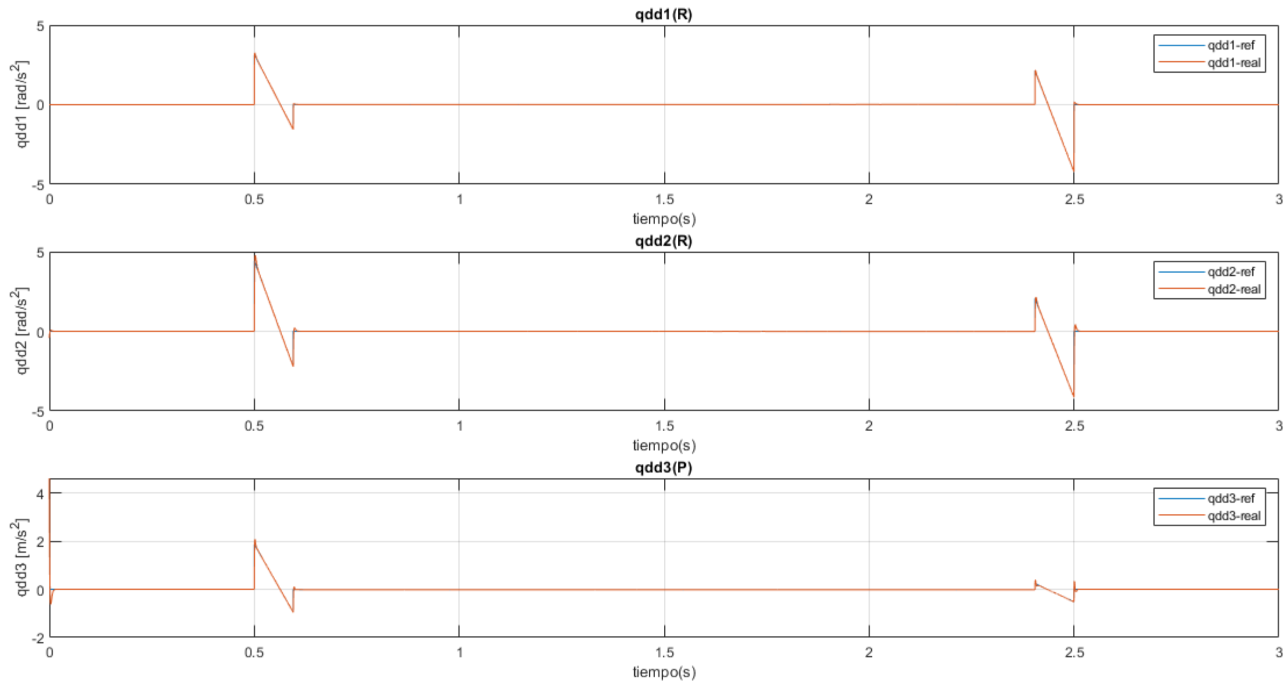
Apreciamos que debido a la agresividad del controlador (ts = 0.01s), hay picos intensos de par también para este tipo de control.



Gracias al PID, al igual que para los errores en cartesianas, los errores cuadráticos en las variables articulares son también despreciables y tienden a 0.

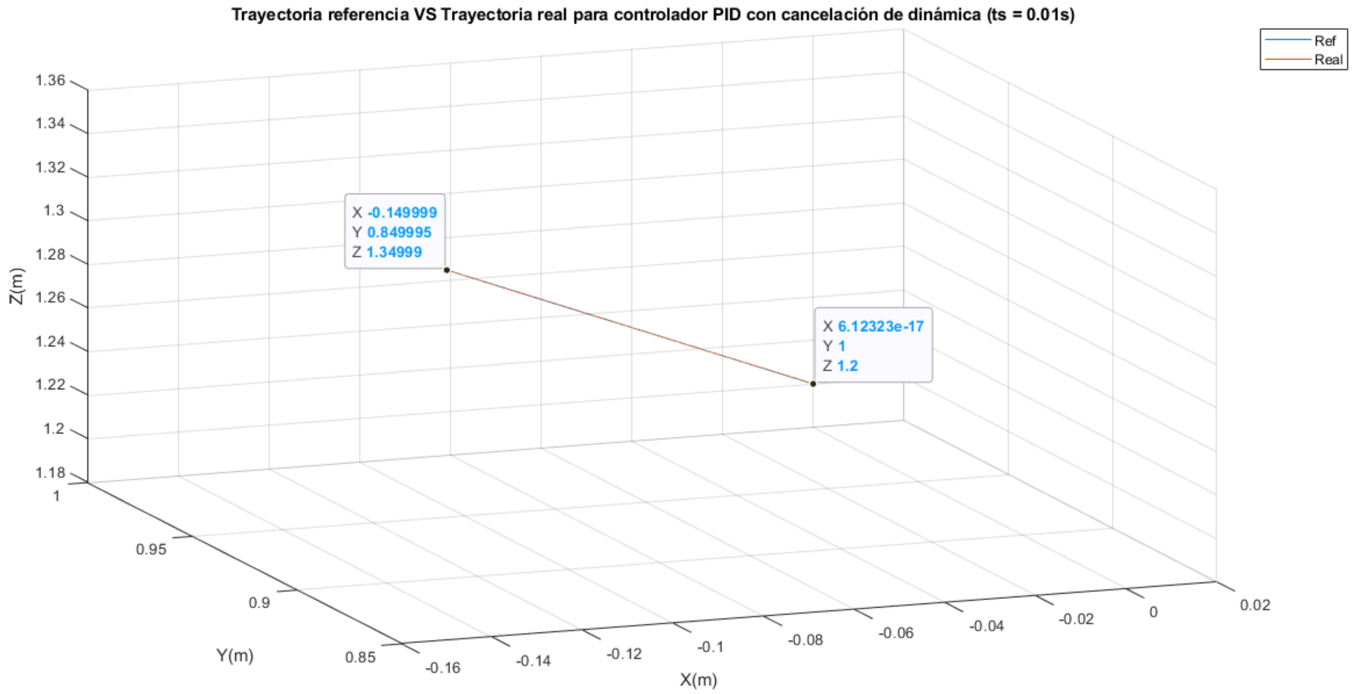


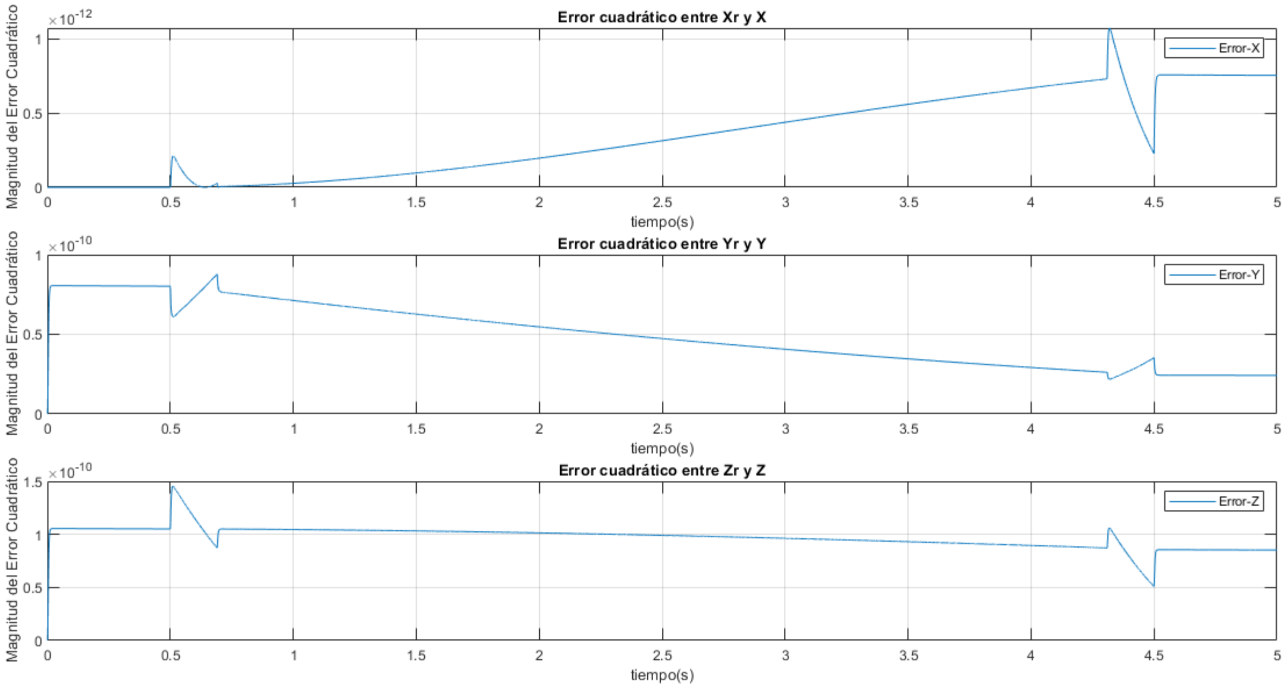




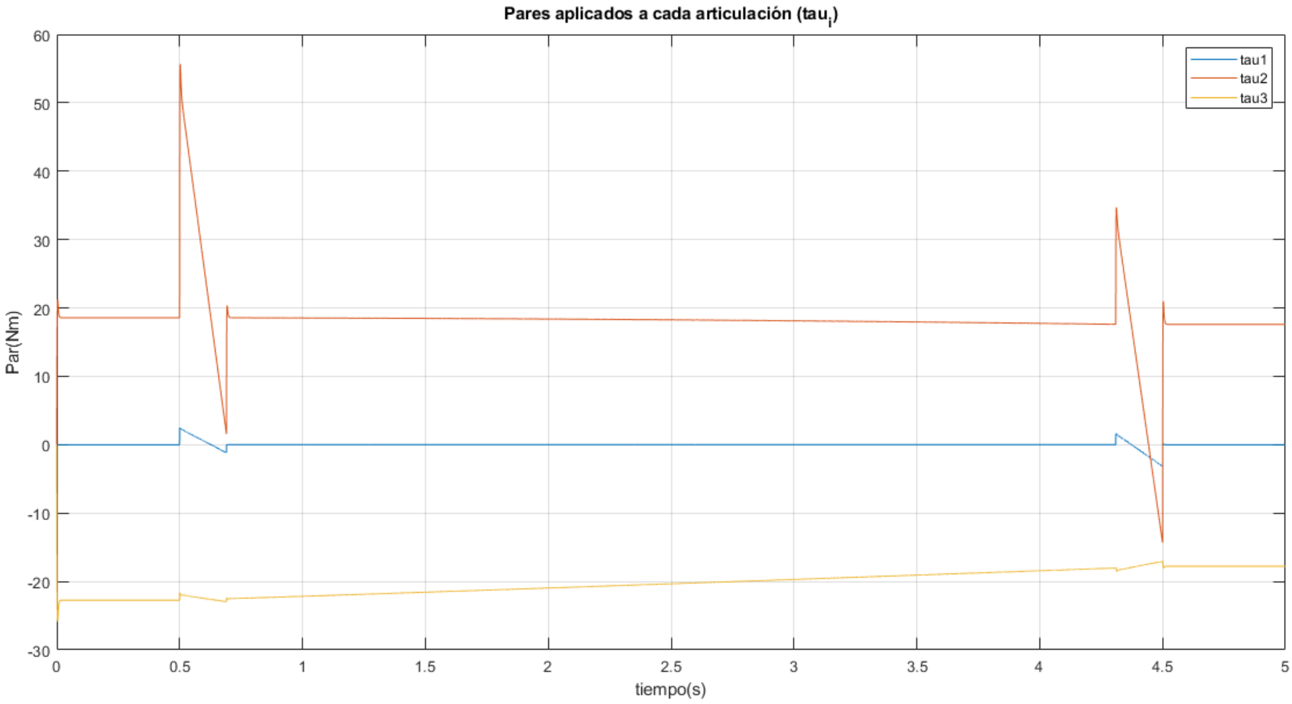
Como esperábamos de la gráfica de error cuadrático entre qi-ref y qi-real, los resultados son mucho mejores y tanto posiciones, velocidades y aceleraciones se ajustan perfectamente, con un error en régimen permanente despreciable (nulo).

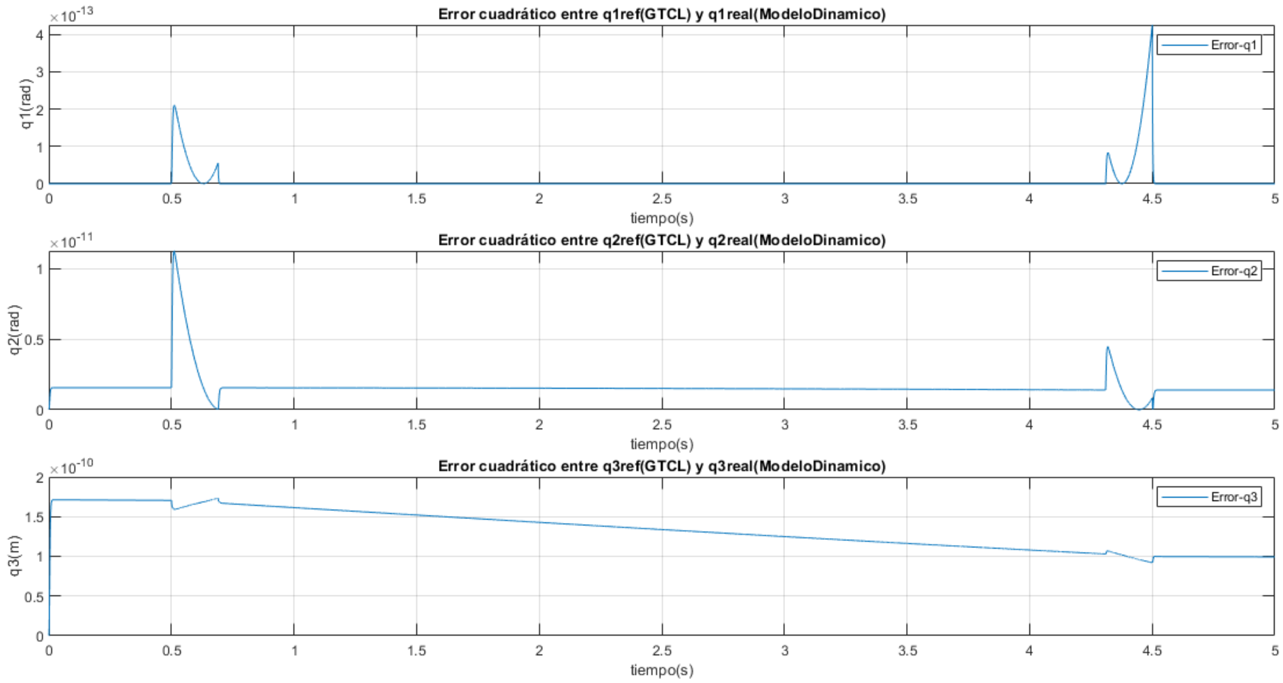
- **Resultados para 20 puntos intermedios en el GTCL y duración del movimiento de 4s (entre t=0.5s y t = 4.5s)**

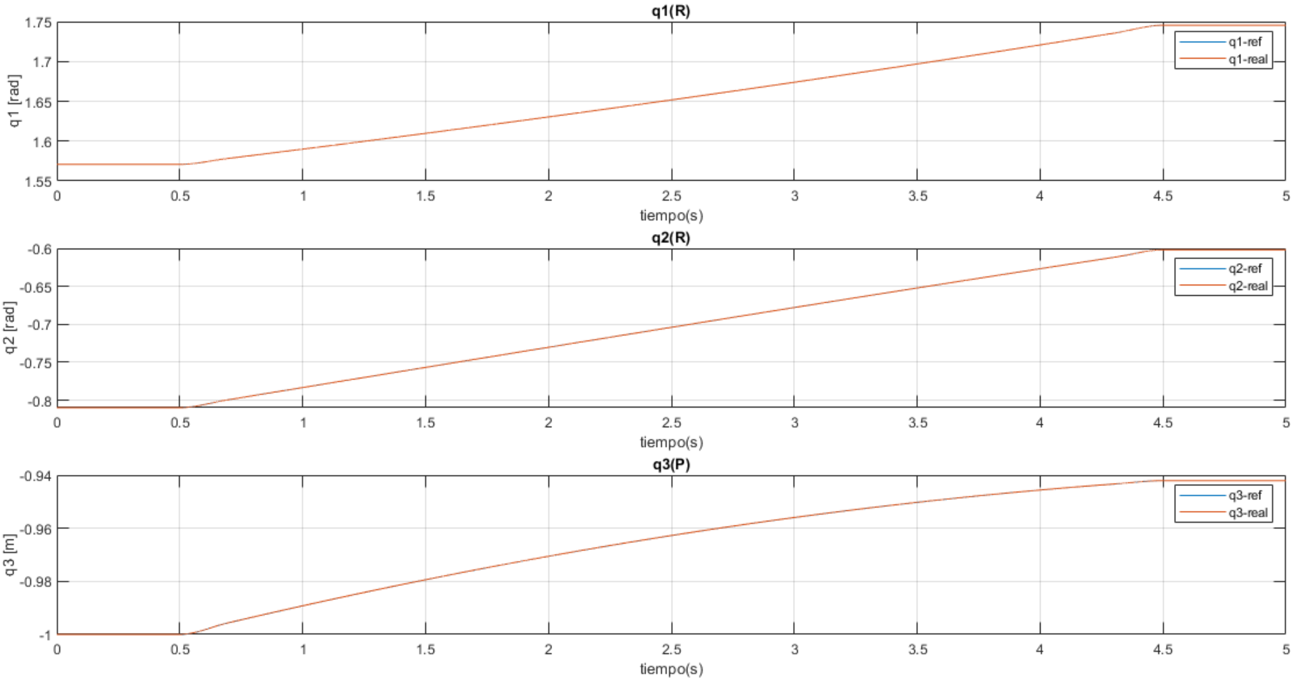


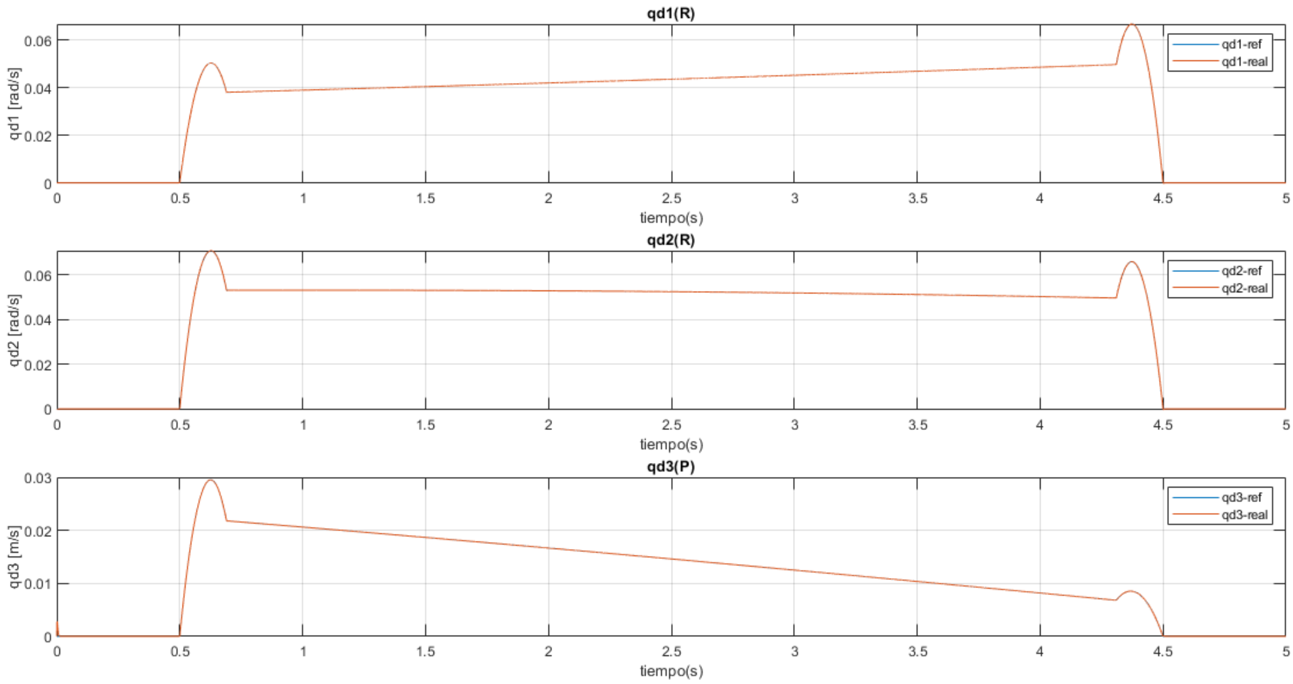


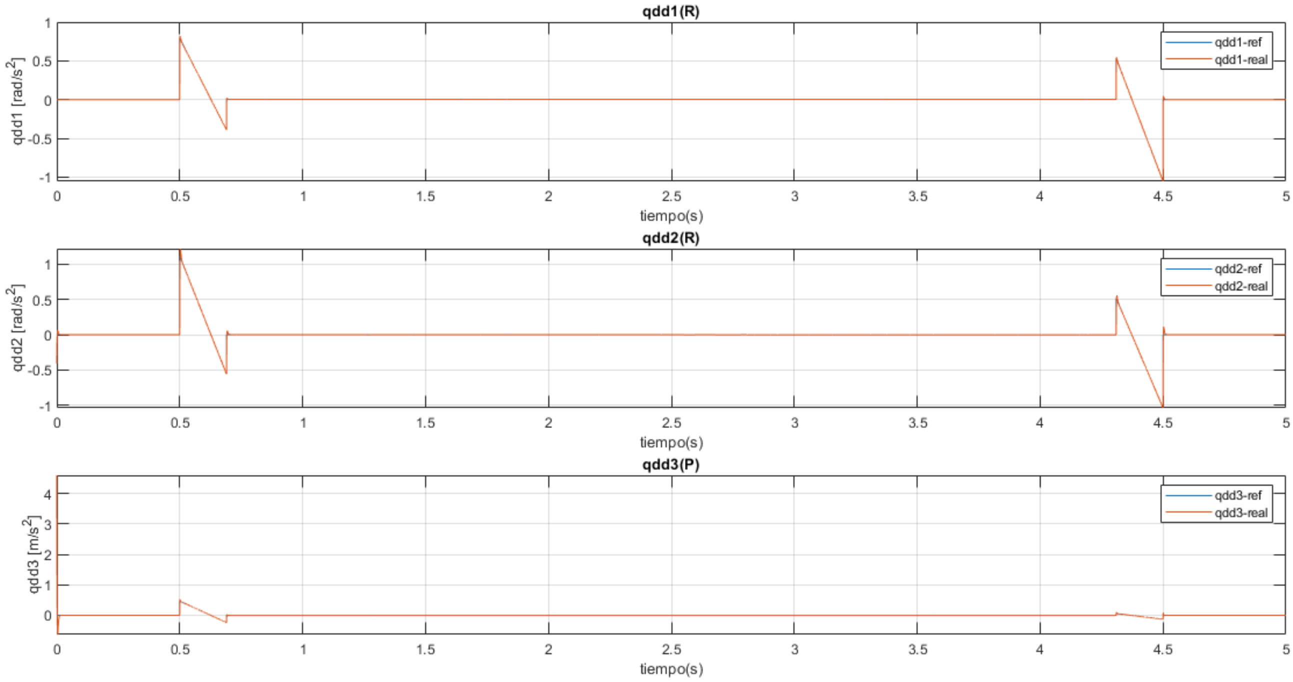
Al aumentar la duración del movimiento, los resultados del control PID son similares, es decir, la trayectoria que realiza el robot se ajusta perfectamente y vemos de nuevo que los errores cuadráticos en cartesianas son despreciables.


Pares aplicados a cada articulación (tau_i)
tau1
tau2
tau3
Par(Nm)
tiempo(s)


Error cuadrático entre q1ref(GTCL) y q1real(ModeloDinamico)
Error-q1
q1(rad)
tiempo(s)
Error cuadrático entre q2ref(GTCL) y q2real(ModeloDinamico)
Error-q2
q2(rad)
tiempo(s)
Error cuadrático entre q3ref(GTCL) y q3real(ModeloDinamico)
Error-q3
q3(m)
tiempo(s)


q1(R)
q1 [rad]
tiempo(s)
q1-ref
q1-real
q2(R)
q2 [rad]
tiempo(s)
q2-ref
q2-real
q3(P)
q3 [m]
tiempo(s)
q3-ref
q3-real


qd1(R)
qd1 [rad/s]
tiempo(s)
qd1-ref
qd1-real
qd2(R)
qd2 [rad/s]
tiempo(s)
qd2-ref
qd2-real
qd3(P)
qd3 [m/s]
tiempo(s)
qd3-ref
qd3-real



Se observa un comportamiento similar que para el caso de duración de 2s, notando picos más suaves en cuanto a pares y aceleraciones.

DISEÑO DE CONTROLADOR DE PAR CALCULADO:

Tras haber trabajado con controladores tipo PD y PID, vamos a realizar un tipo de control algo más complejo, el control conocido como par calculado.

Al igual que se hizo en clase, vamos a diseñar un PD con par calculado.

Este control es algo diferente y por ello se parte de que, si la cancelación es buena nuestra:

$$G(s) = \frac{1}{s^2}\ (doble\ integrador)$$

Mientras que el controlador tendrá la siguiente estructura:

$$C(s) = K_P + K_D \cdot s$$

Omitiendo el desarrollo visto en clase nos quedan una serie de expresiones:

Se sigue considerando lo siguiente para las expresiones: $\delta = 1\ y\ t_s = \frac{6}{\omega_n}$

$$\begin{cases} K_P = \omega_n^2 = \dfrac{36}{t_s^2} \\ K_D = 2 \cdot \omega_n = \dfrac{12}{t_s} \end{cases}$$

En este caso, particularizando (ControladoresParCalculado) para el mismo tiempo de subida que en los casos anteriores, $\boldsymbol{t_s = 0.01s}$ nos quedan:

$$K_{P\_1,2,3} = 360000$$

$$K_{D\_1,2,3} = 1200$$

Sin desarrollar mucho más expresando los controladores, se adjunta la implementación en Simulink de este tipo de control, que es diferente a los dos anteriores en cuanto a implementación.

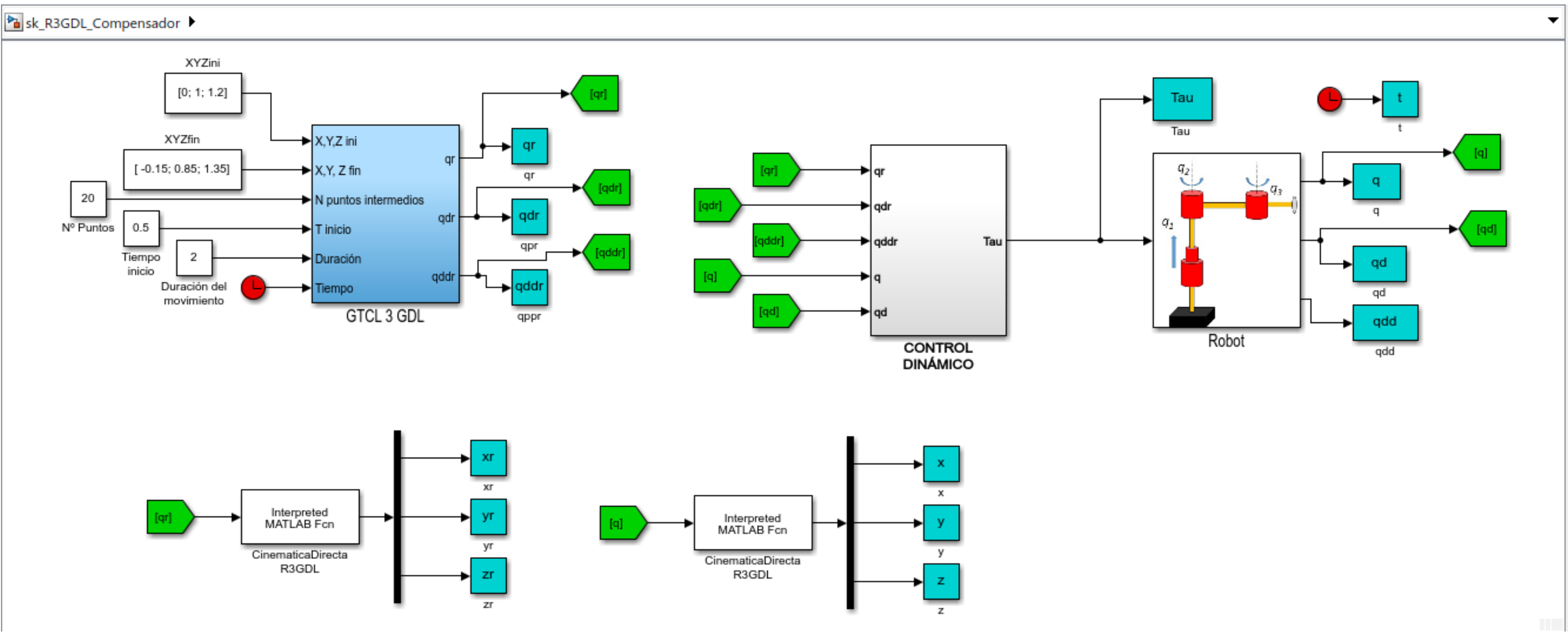


Figura 25: Implementación general en Simulink del sistema conjunto

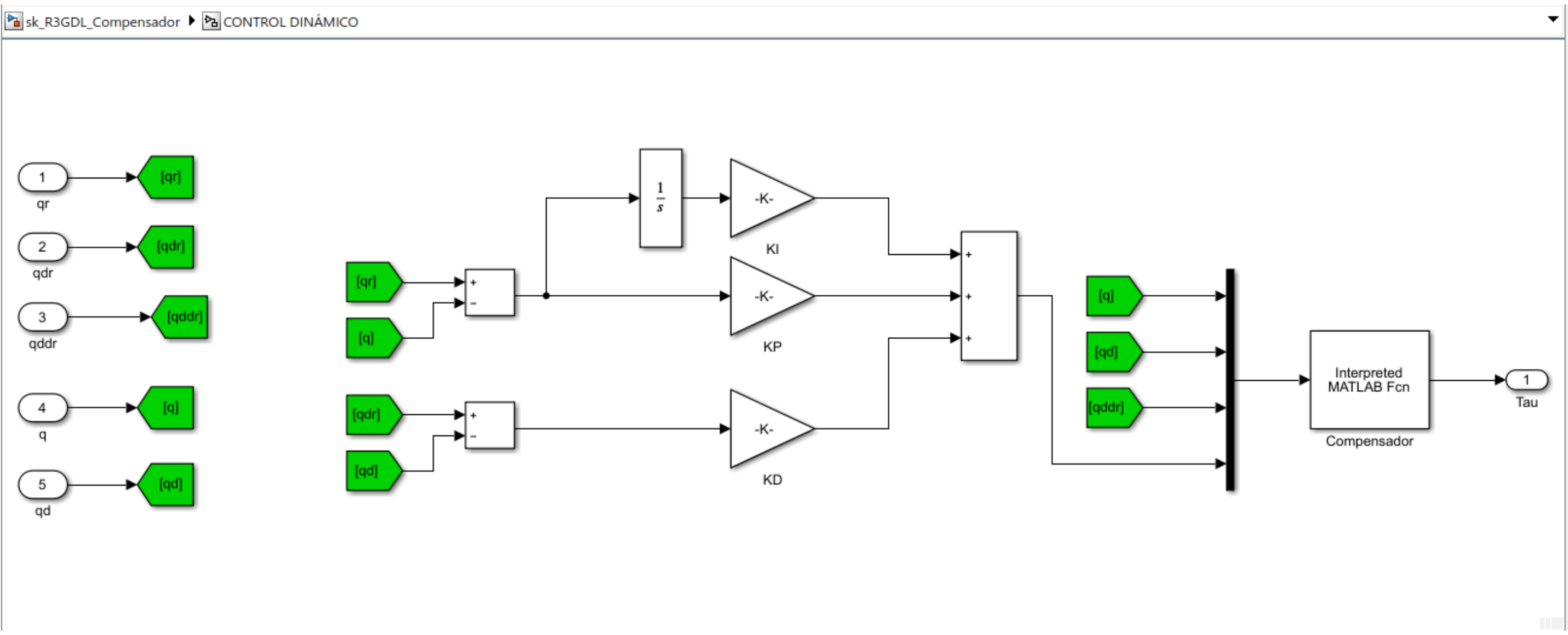
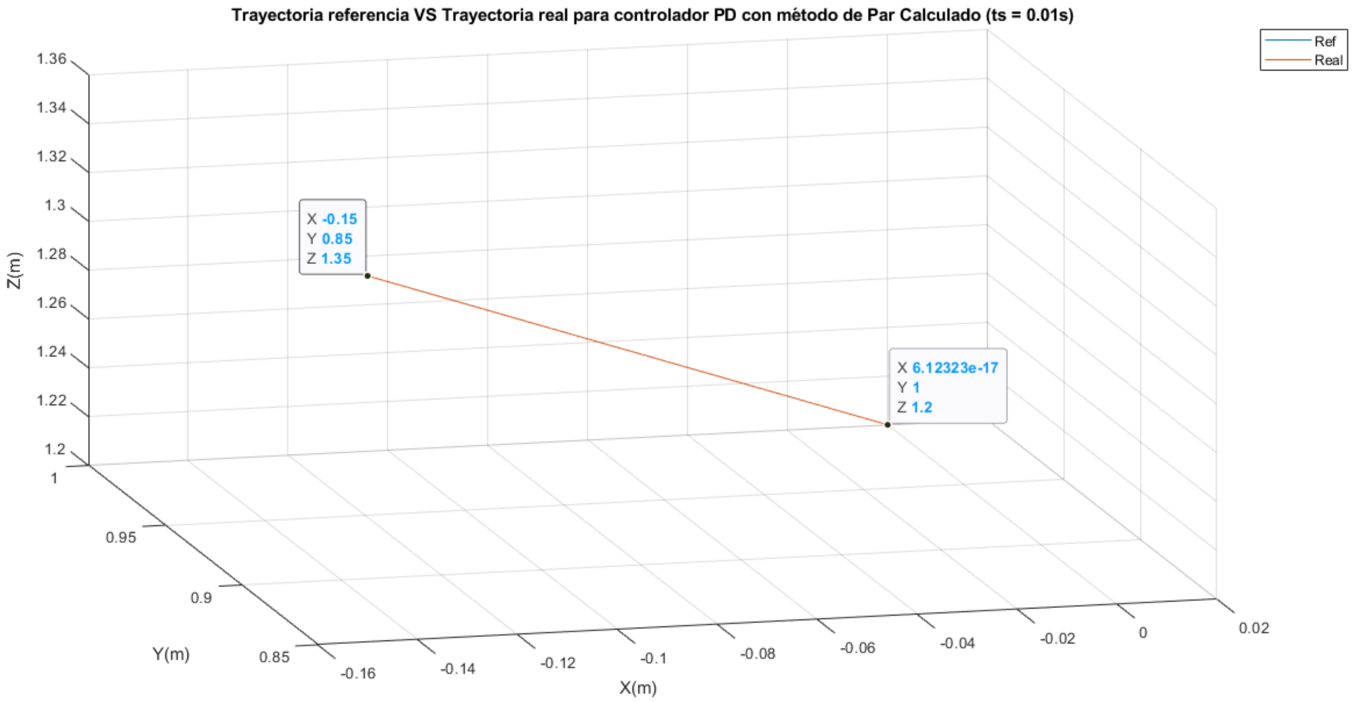


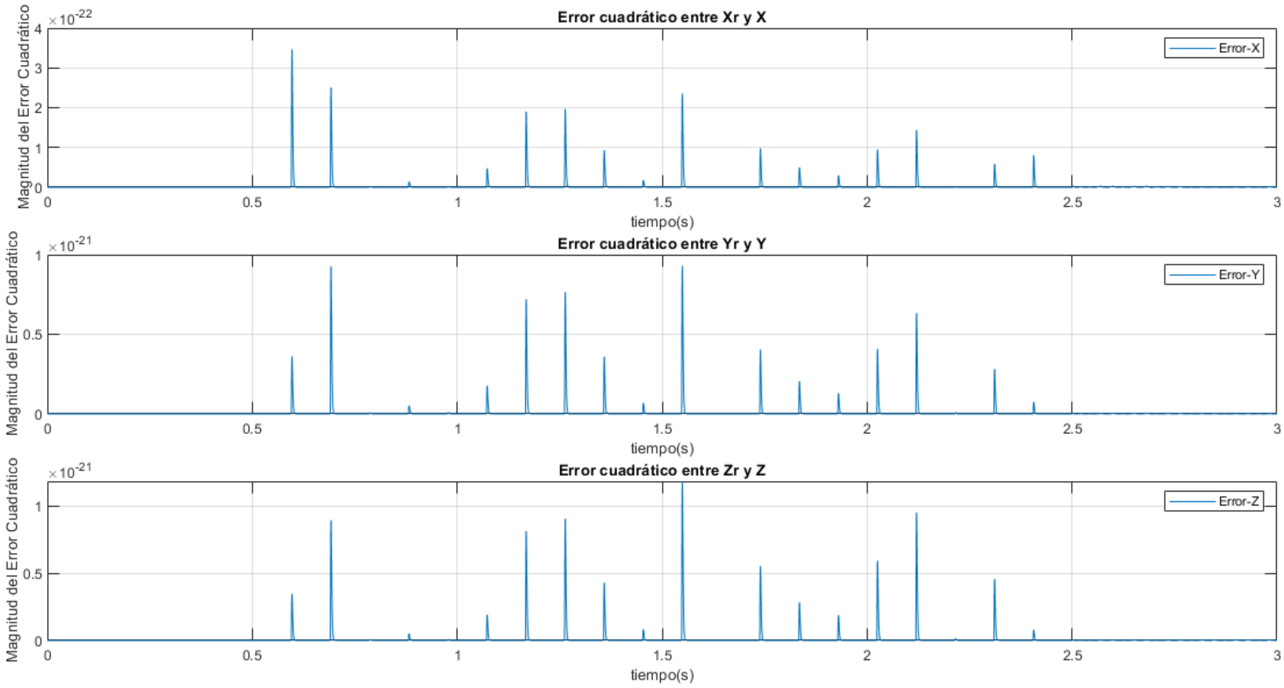
Figura 26: Implementación en Simulink del controlador PD para tipo Par Calculado

Donde es el bloque de Compensador el que implementa el algoritmo de este tipo de control conocido como Par calculado. Dicho bloque lleva implícita la siguiente función anexada: Compensador_ParCalculado

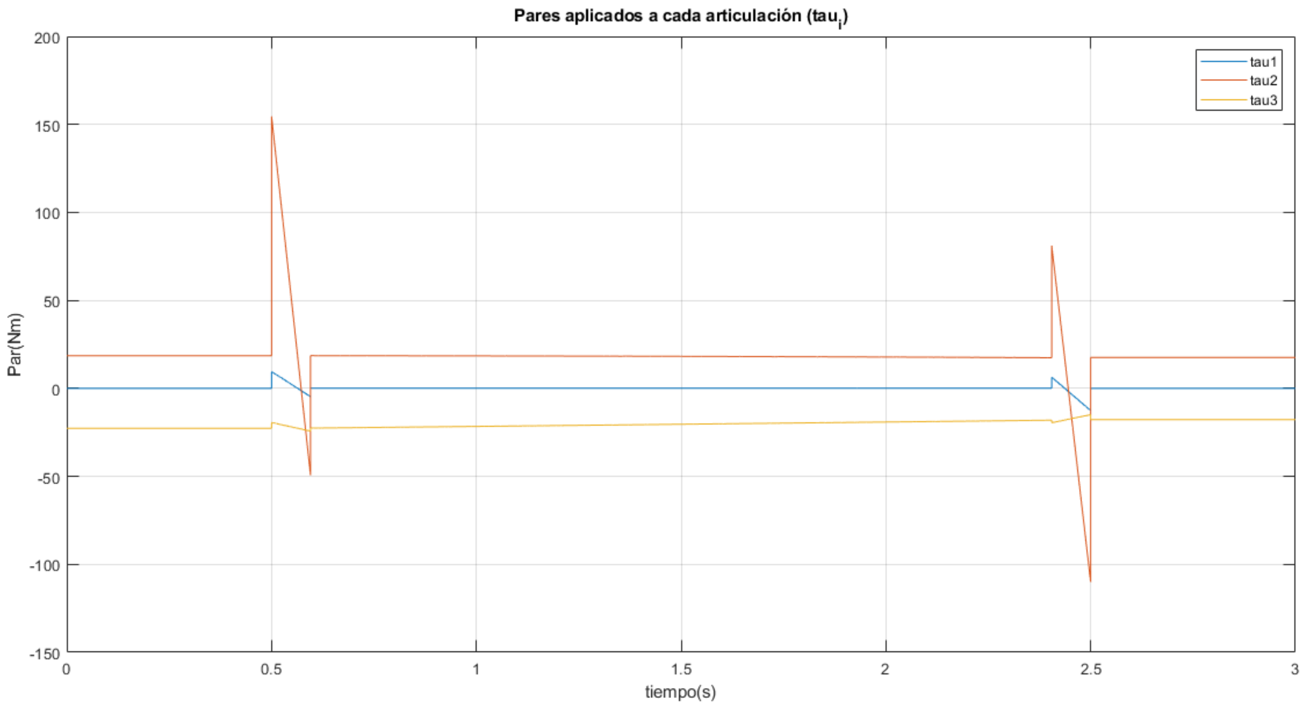
Podemos ya por fin recoger resultados de dicho controlador PD con par calculado.

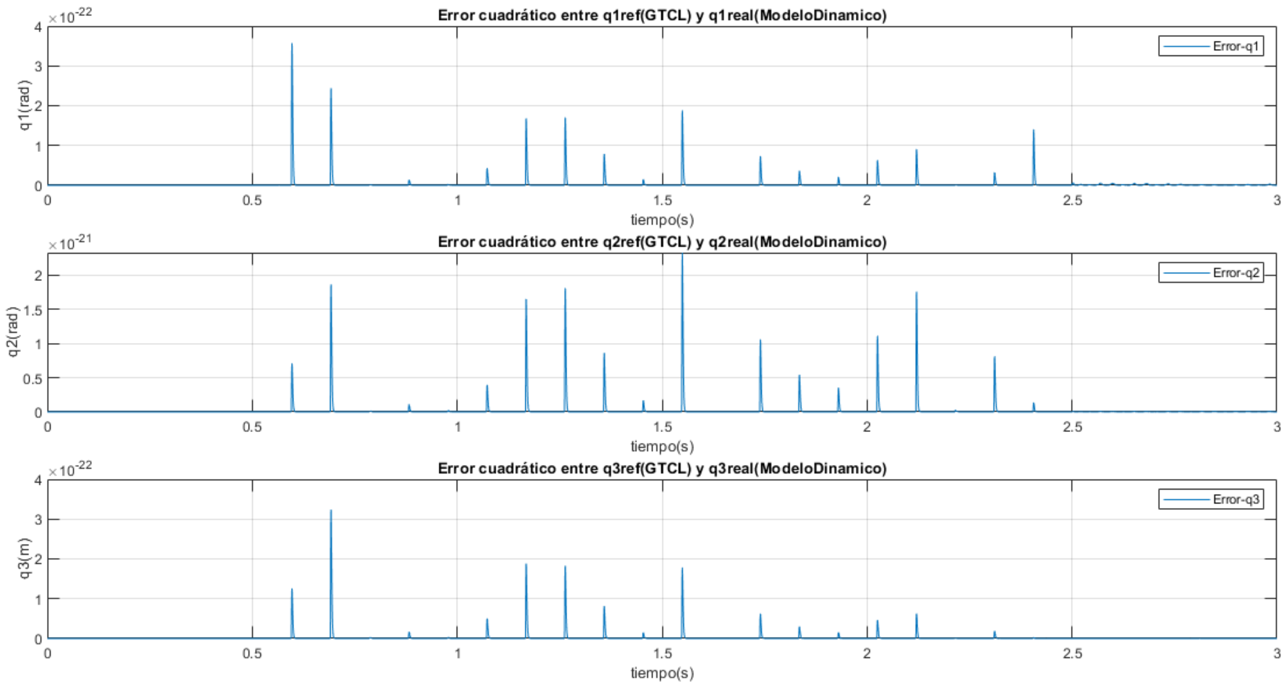
- **Resultados para 20 puntos intermedios en el GTCL y duración del movimiento de 2s (entre t=0.5s y t = 2.5s):**



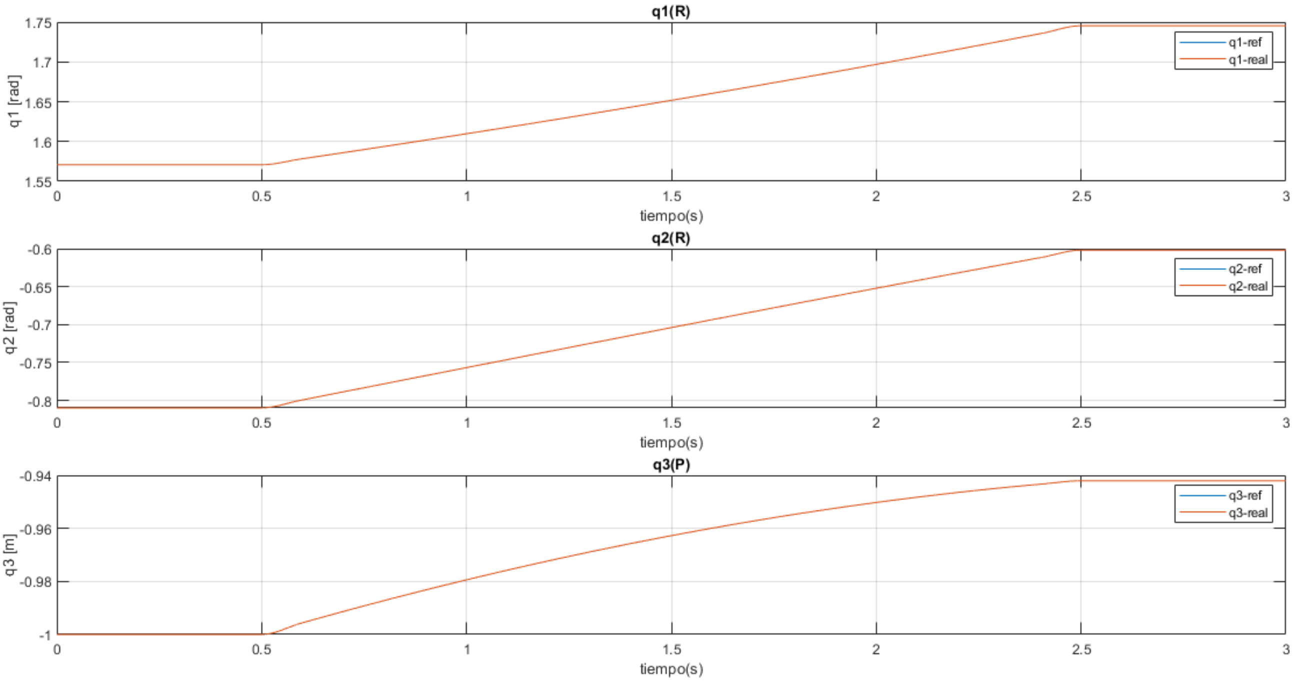


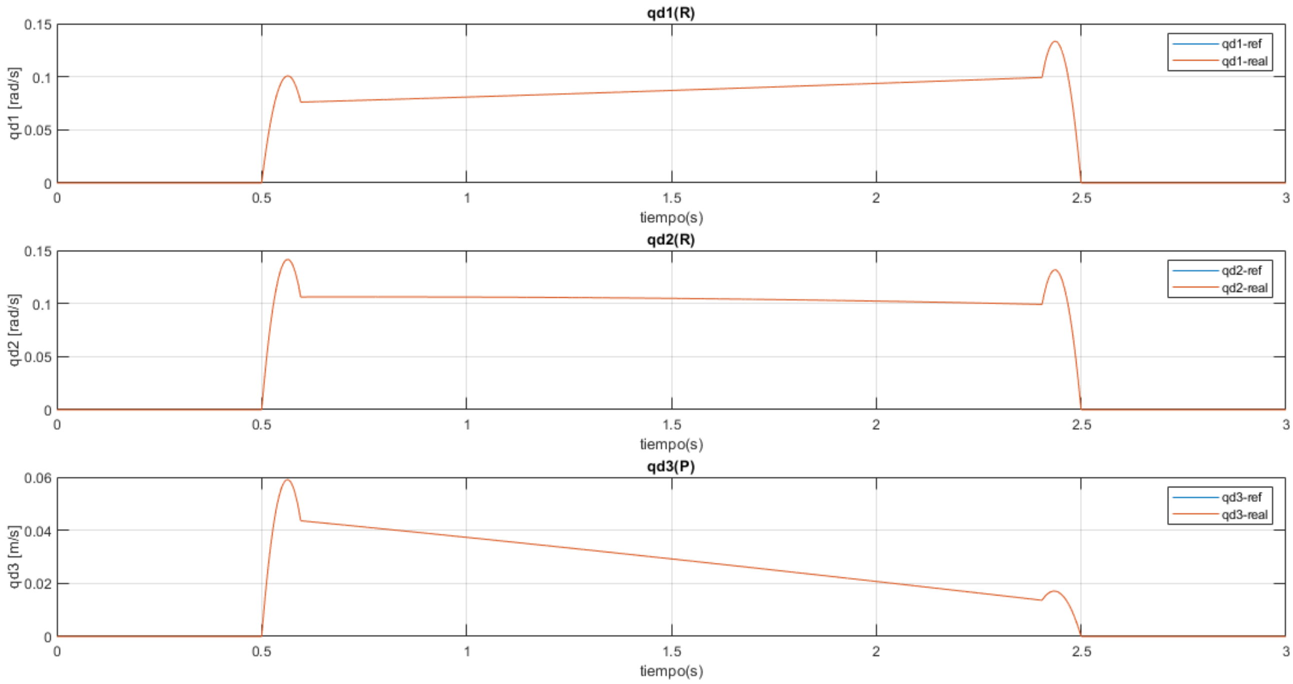
Observamos que si bien con el PID los errores cuadráticos eran despreciables (1e-10), con este método de control aún mejora los resultados del PID, siendo más despreciables aún (1e-21).

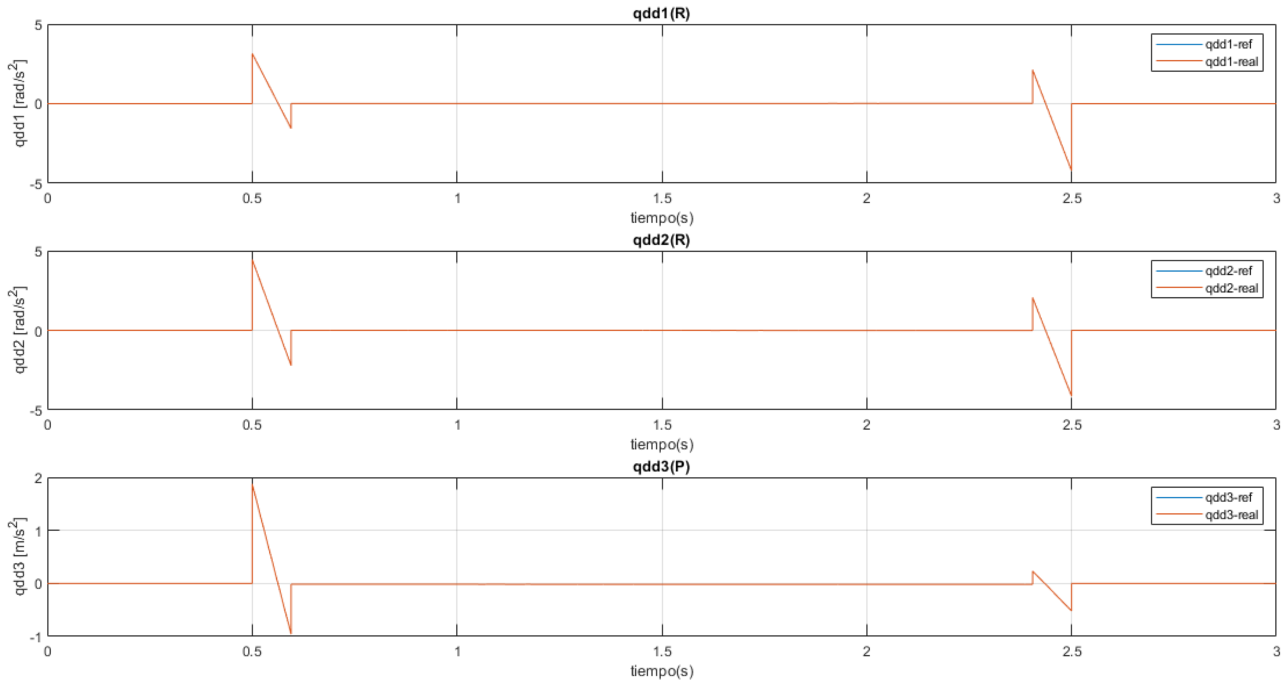




Al igual que para el error cuadrático en cartesianas, el error cuadrático para las variables articulares (qi-ref VS qi-real) es del mismo orden de magnitud que el de cartesianas (1e-22), despreciable como vemos.

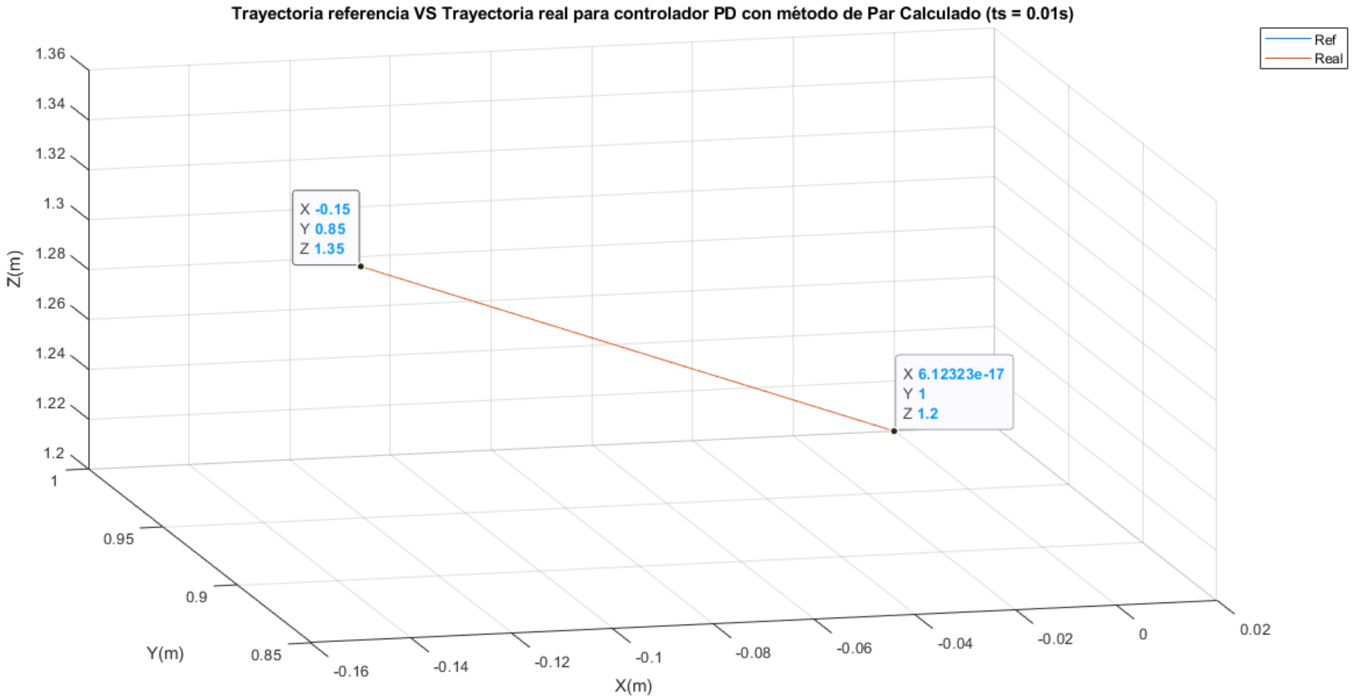

q1(R)
q1 [rad]
tiempo(s)
q1-ref
q1-real
q2(R)
q2 [rad]
tiempo(s)
q2-ref
q2-real
q3(P)
q3 [m]
tiempo(s)
q3-ref
q3-real

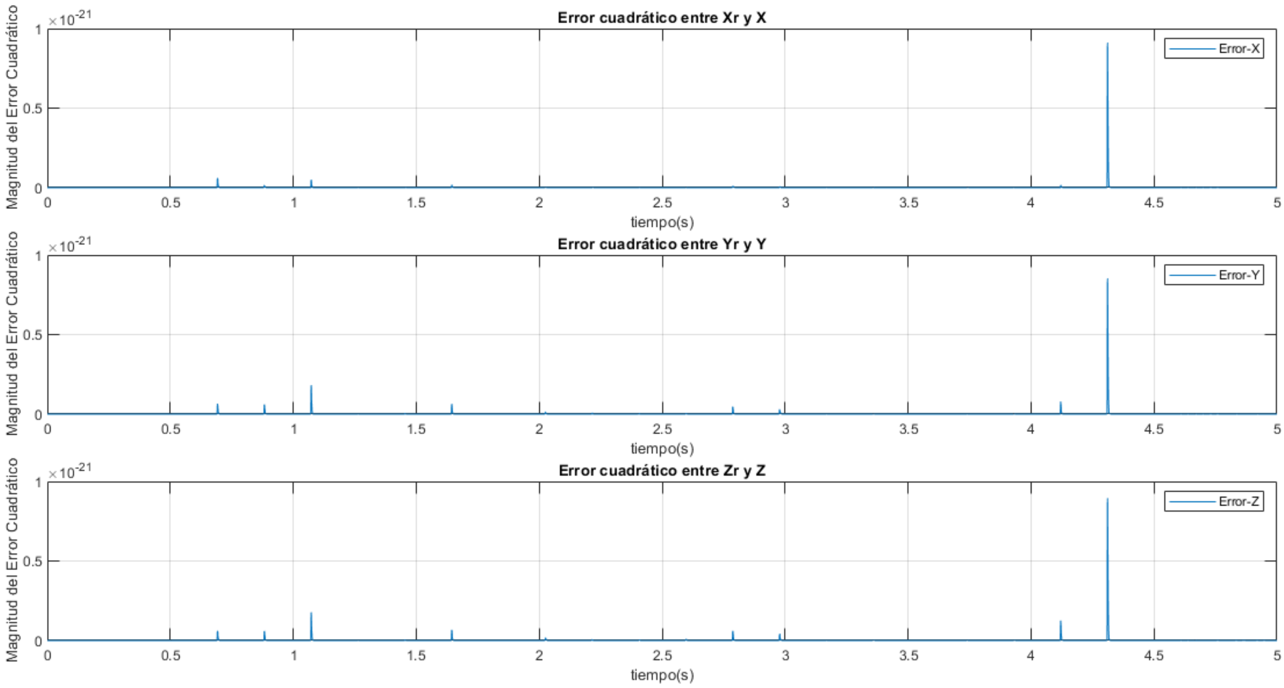

qd1(R)
qd1 [rad/s]
tiempo(s)
qd1-ref
qd1-real
qd2(R)
qd2 [rad/s]
tiempo(s)
qd2-ref
qd2-real
qd3(P)
qd3 [m/s]
tiempo(s)
qd3-ref
qd3-real



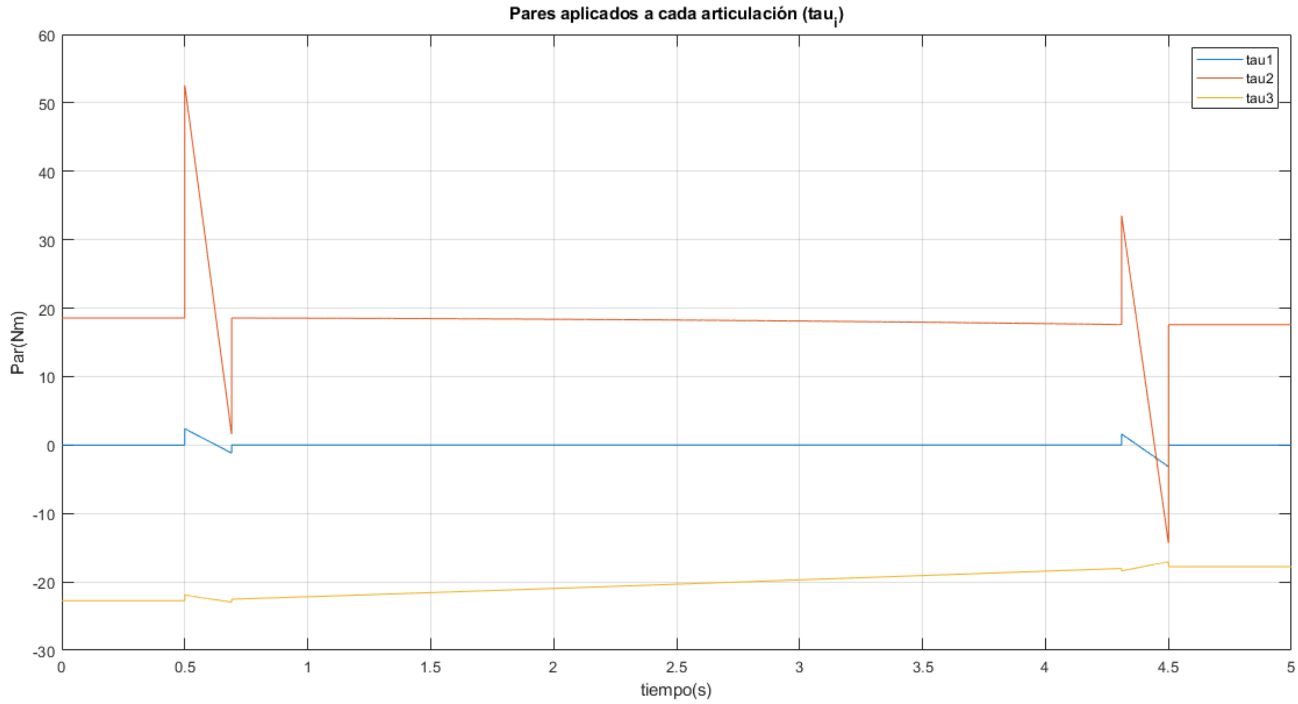
Como esperábamos de la gráfica de error cuadrático entre qi-ref y qi-real, los resultados son mucho mejores y tanto posiciones, velocidades y aceleraciones se ajustan perfectamente, con un error en régimen permanente despreciable (nulo).

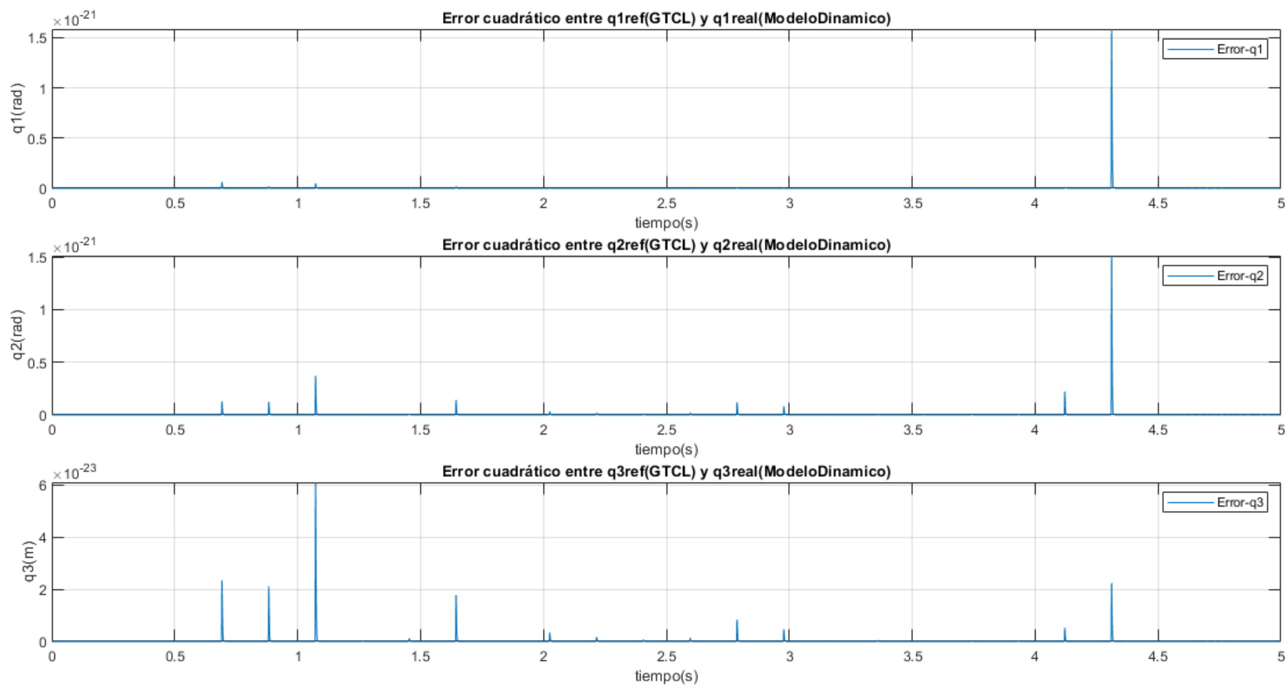
- **Resultados para 20 puntos intermedios en el GTCL y duración del movimiento de 4s (entre t=0.5s y t = 4.5s)**



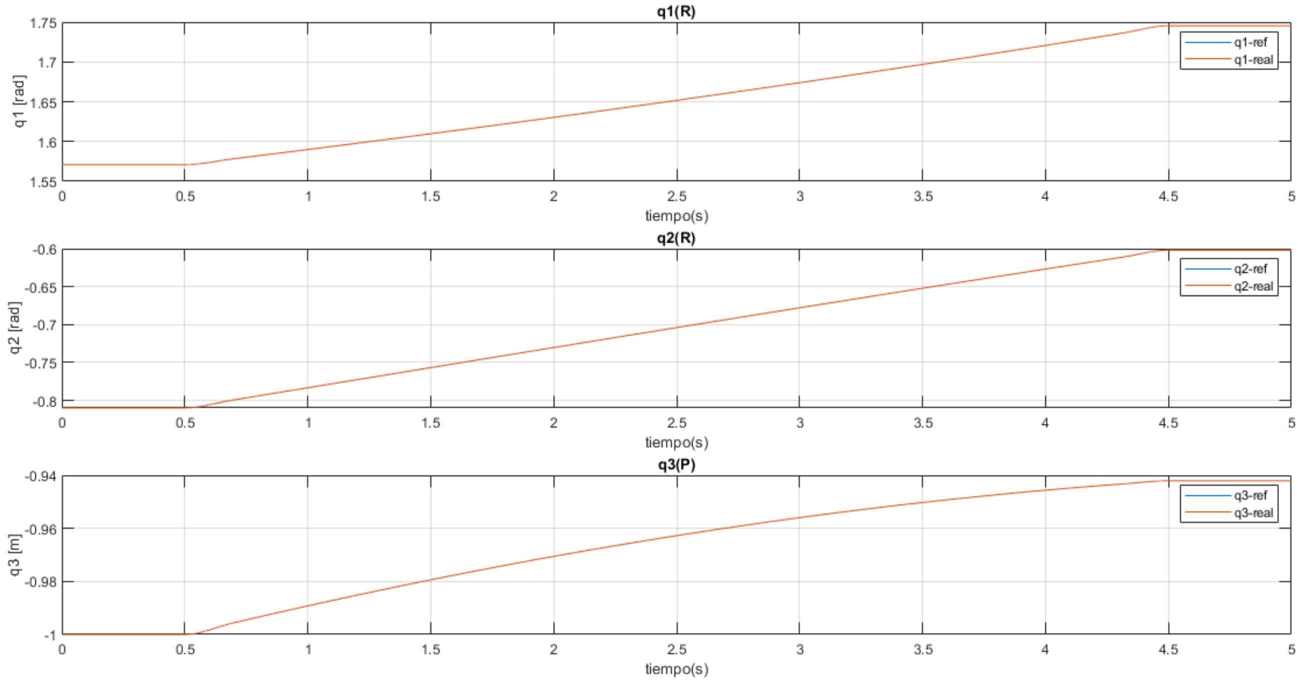


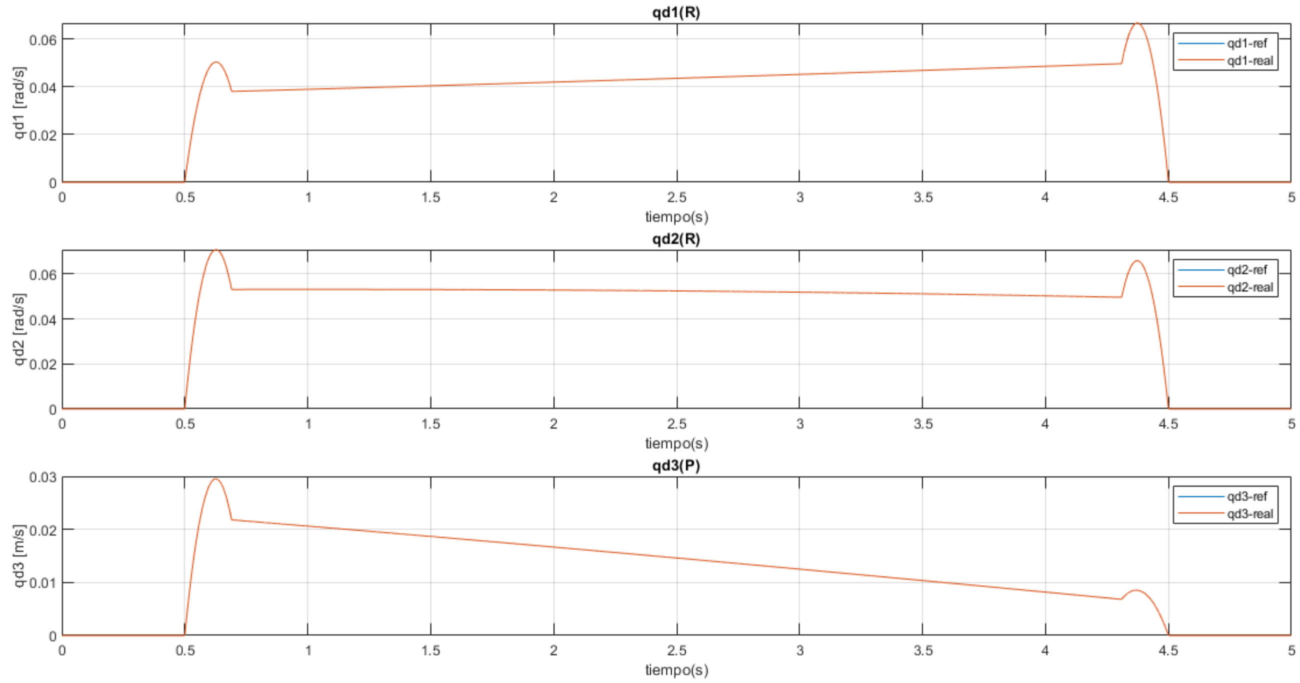
La precisión sigue siendo igual de buena y la magnitud del error cuadrático también vuelve a ser totalmente despreciable.

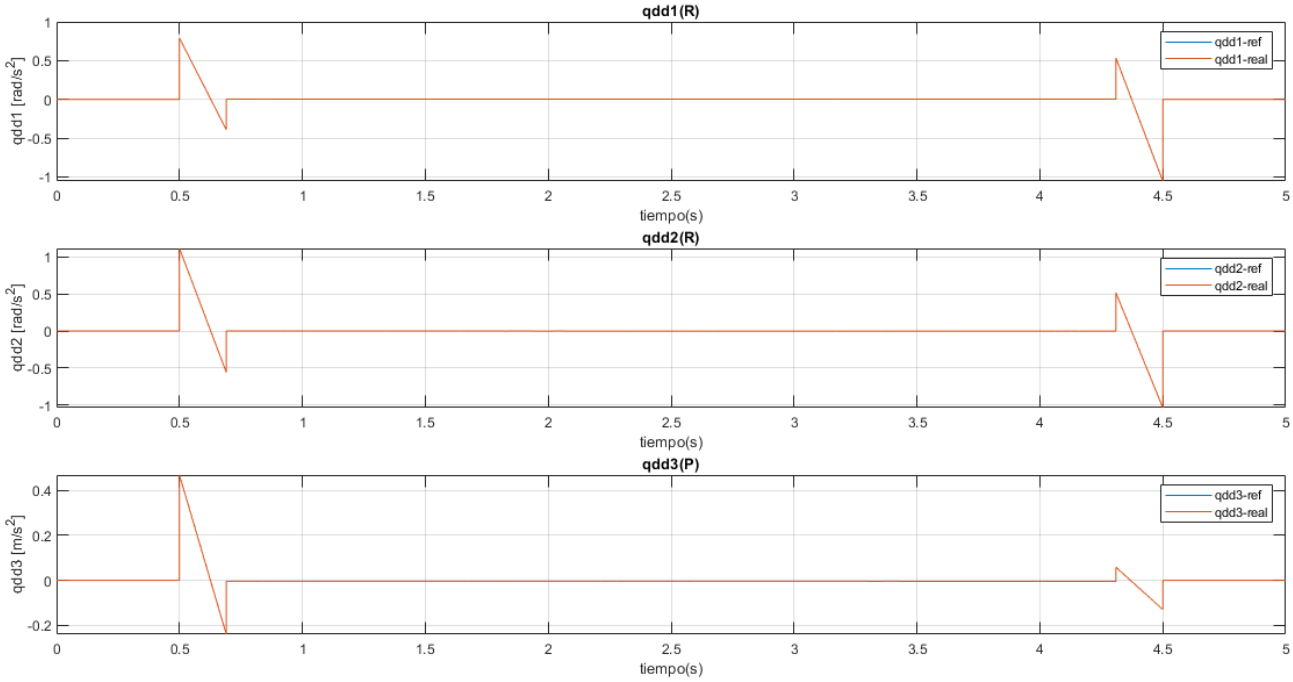




Al igual que para el error cuadrático en cartesianas, el error cuadrático para las variables articulares (qi-ref VS qi-real) es del mismo orden de magnitud que el de cartesianas (1e-21), despreciable como vemos.


q1(R)
q1 [rad]
q1-ref
q1-real
tiempo(s)
q2(R)
q2 [rad]
q2-ref
q2-real
tiempo(s)
q3(P)
q3 [m]
q3-ref
q3-real
tiempo(s)


qd1(R)
qd1 [rad/s]
qd1-ref
qd1-real
tiempo(s)
qd2(R)
qd2 [rad/s]
qd2-ref
qd2-real
tiempo(s)
qd3(P)
qd3 [m/s]
qd3-ref
qd3-real
tiempo(s)



Al igual que para un movimiento de duración 2s, ocurre que si se hace más lento el movimiento (4s), como esperábamos de la gráfica de error cuadrático entre qi-ref y qi-real, los resultados son mucho mejores y tanto posiciones, velocidades y aceleraciones se ajustan perfectamente, con un error en régimen permanente despreciable (nulo).

Para sacar estas gráficas para los 3 tipos de control se ha utilizado el siguiente código genérico: Gráficas_Controladores, modificado para cada caso (títulos, leyendas…).

- **Conclusiones finales para este apartado de control:**

Tras analizar detenidamente los resultados conseguidos con los diferentes tipos de control: PD, PID y PD con método de par calculado, observamos que, aunque en teoría un PD es suficiente para tener buenos resultados, aunque acumule error. Sin embargo, al no tener en cuenta en el diseño de control los efectos de la gravedad, sino que se consideran perturbaciones externas, puede ocurrir, como en mi caso, que el robot se vea muy afectado por ese efecto gravitatorio y por tanto el control con PD acumula un error en la trayectoria y demás variables que no son para nada despreciables en tareas de robótica.

Se comprobó que, si se incluía en el PD la compensación de gravedad, este error acumulado desaparecía y con esta ampliación el PD si que ofrecía resultados más que aceptables.

Posteriormente, se vio que al diseñar el control PID, sin tener que incluir ningún tipo de compensación, los resultados son muy buenos y prácticamente ambas trayectorias coinciden sin error prácticamente. Considero que este tipo de control es suficiente para tener buenos resultados y no requiere de mayor complejidad que el propio cálculo, siendo la implementación bastante simple.

Finalmente, tenemos el último control propuesto, el PD con método de par calculado. Con este control se han obtenido los mejores resultados en cuanto a precisión. Luego si la aplicación requiere de una precisión muy fina, este método de control sería mejor que un PID, aunque en mi opinión personal el control tipo PID ya tiene precisión suficiente. Sin embargo, tampoco parece mucho más costoso computacionalmente este método que el de implementación de un PID, por tanto quizás sí que es más interesante usar este tipo de control.

6.- Anexo (Códigos)

- **Startup_rvc:**

```
disp('Robotics, Vision & Control: (c) Peter Corke 1992-2011
http://www.petercorke.com')
tb = false;
rvcpath = fileparts( mfilename('fullpath') );

robotpath = fullfile(rvcpath, 'robot');
if exist(robotpath,'dir')
    addpath(robotpath);
    tb = true;
    startup_rtb
end

visionpath = fullfile(rvcpath, 'vision');
if exist(visionpath,'dir')
    addpath(visionpath);
    tb = true;
    startup_mvtb
end

if tb
    addpath(fullfile(rvcpath, 'common'));
    addpath(fullfile(rvcpath, 'simulink'));
end

clear tb rvcpath robotpath visionpath
```

- **Función MDH:**

```
% MDH       Obtención de la Matriz de transformación homogénea
%           a partir de parámetros de Denavit Hartenberg estandar.
%        DH = MDH(THETA, D, A, ALFA) devuelve la matriz de transformacion
%              homogénea 4 x 4 a partir de los parametros de Denavit-
Hartenberg
%           THETA,D, ALFA y A.
%

function dh=MDH(theta, d, a, alfa)
dh=[cos(theta)             -cos(alfa)*sin(theta)    sin(alfa)*sin(theta)
a*cos(theta);
    sin(theta)              cos(alfa)*cos(theta)   -sin(alfa)*cos(theta)
a*sin(theta);
           0                          sin(alfa)                 cos(alfa)
d;
           0                                  0                         0
1];
```

- **Código_MTH_Articulares y MCD:**

```
%Cinematica Directa Proyecto Robotica
%Brazo = c ; Muñeca = 2
%Definición parámetros dimensionales
syms L1 L2 L3 L4 real;
syms L1m L2m L3m L4m real;
syms q1 q2 q3 q4 q5 q6 real;
pi1 = sym(pi);
%Definición parametros DH para robot C2
theta1 = q1;        d1 = L1;          a1 = 0;        alfa1 = pi1/2;
theta2 = q2;        d2 = 0;           a2 = L2;       alfa2 = pi1/2;
theta3prima = 0;    d3prima = q3;     a3prima = 0;   alfa3prima = 0;
theta3 = 0;         d3 = L1m;         a3 = 0;        alfa3 = 0;
theta4 = q4;        d4 = L2m;         a4 = 0;        alfa4 = pi1/2;
theta5 = q5;        d5 = 0;           a5 = L3m;      alfa5 = 0;
theta6 = q6;        d6 = 0;           a6 = L4m;      alfa6 = 0;

%Matrices de Transformacion Homogeneas articulares
A01 = MDH(theta1, d1, a1, alfa1); % {0} a {1}
A12 = MDH(theta2, d2, a2, alfa2); % {1} a {2}
A23prima = MDH(theta3prima, d3prima, a3prima, alfa3prima); %{2} a {3'}
A3prima3 = MDH(theta3, d3, a3, alfa3); % {3'} a {3}
A34 = MDH(theta4, d4, a4, alfa4); % {3} a {4}
A45 = MDH(theta5, d5, a5, alfa5); % {4} a {5}
A56 = MDH(theta6, d6, a6, alfa6); % {5} a {6}

%MTH base(0)->desacople_muñeca(3')(brazo)
T03prima = simplify(A01*A12*A23prima);

%MTH desacople_muñeca(3')->garra(6)(muñeca)
T3prima6 = simplify(A3prima3*A34*A45*A56);

%MTH base(0)->garra(6)(robot_completo)
Trobot = simplify(T03prima*T3prima6);
```

- **Ecuaciones Prototipo (para cinemática inversa): https://core.ac.uk/download/pdf/157811138.pdf**
  (última página del enlace)

Apéndice A. Ecuaciones Prototipo

La siguiente lista recoge las principales expresiones trigonométricas prototipo utilizadas en este trabajo según pueden encontrarse en (Paul, 1981; Rieseler et al., 1990):

$$\sin(\theta) = a \Longrightarrow \theta = \operatorname{atan2}\left(a, \pm\sqrt{1-a^2}\right) \tag{A.1}$$

$$\cos(\theta) = a \Longrightarrow \theta = \operatorname{atan2}\left(\pm\sqrt{1-a^2}, a\right) \tag{A.2}$$

$$\left.\begin{array}{l}\sin(\theta) = a \\ \cos(\theta) = b\end{array}\right\} \Longrightarrow \theta = \operatorname{atan2}(a, b) \tag{A.3}$$

$$a\cos(\theta) + b\sin(\theta) = 0 \Longrightarrow \left\{\begin{array}{l}\theta = \operatorname{atan2}(-a, b) \\ \text{ó} \\ \theta = \operatorname{atan2}(a, -b)\end{array}\right. \tag{A.4}$$

$$a\cos(\theta) + b\sin(\theta) = c \Rightarrow \theta = \operatorname{atan2}(c, \pm\alpha) - \operatorname{atan2}(a, b) \tag{A.5}$$

donde $\alpha = \sqrt{a^2 + b^2 - c^2}$.

$$\left.\begin{array}{l}a\cos(\theta_1) + b\cos(\theta_2) = e \\ a\sin(\theta_1) + b\sin(\theta_2) = f\end{array}\right\} \Rightarrow$$

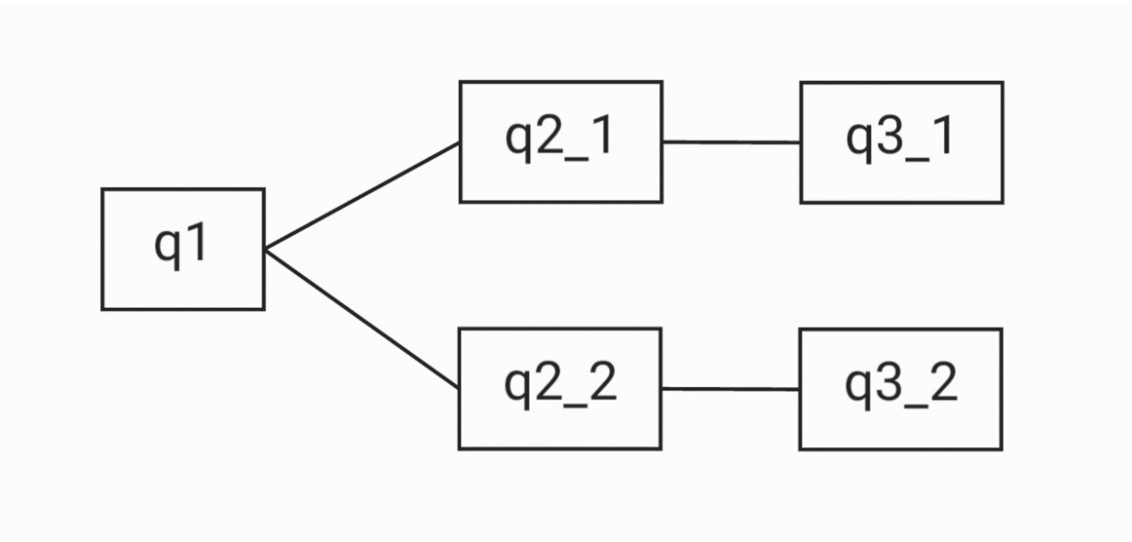
$$\Rightarrow \theta_1 = \operatorname{atan2}\left(\beta, \pm\sqrt{e^2 + f^2 - \beta^2}\right) \tag{A.6}$$

donde $\beta = \dfrac{a^2 + e^2 + f^2 - b^2}{2a}$.

- **Código_Cinemática_Inversa:**

```
%Cinematica inversa del Proyecto
%Parámetros dimensionales del robot(datos)
L1 = 0.8;
L2 = 0.4;
L3 = 0.6;
L4 = 0.4;
%Posición del efector final(Extremo del brazo sin la muñeca)
%Se parte de esta posición como dato para hallar q1, q2 y q3
px = -L3-L4;
py = 0;
pz = L1+L2;
%Expresiones de q1, q2 y q3
%Ruta de soluciones:
%-->tomando q1; 2 sol para q2: q2_1 y q2_2; 2 sol
%para q3: q3_1 y q3_2;
q1 = atan2(py,px);
b = pz-L1;
a = cos(q1)*px+sin(q1)*py;
c = L2;
q2_1 = atan2(L2,+sqrt(a^2+b^2-c^2))-atan2(a,b);
q2_2 = atan2(L2,-sqrt(a^2+b^2-c^2))-atan2(a,b);
q3_1 = sin(q2_1)*(py*sin(q1)+px*cos(q1))+cos(q2_1)*(L1-pz);
q3_2 = sin(q2_2)*(py*sin(q1)+px*cos(q1))+cos(q2_2)*(L1-pz);
```

Como vemos, hay varias ramas de soluciones, para clarificar cómo funcionan se adjunta también un esquema aclaratorio.



- **Código_Trayectoria_Circular:**

```
%Trayectoria Circular
L1 = 0.8; L2 = 0.4; L3 = 0.6; L4 = 0.4; %Dimensiones Físicas Robot
r = 0.2;                  %Radio
theta = 0:0.1:2*pi;       %Ángulo circunferencia
n = length(theta);
q1 = zeros(n,1); % Definir vectores de q1, q2 y q3 con longitud n puntos
q2 = zeros(n,1); % inicializados a 0, para en el bucle asignarles valores a las
3qs
q3 = zeros(n,1); % en función de los diferentes puntos de la circunferencia
(x,y,z) que se van recorriendo
origen = [0.2 0.8 1];
j=1:n;

for i=j
    x(i) = origen(1)+r*cos(theta(i));  % Se define las coordenadas xyz a partir
de las polares (r,theta)
    y(i) = origen(2)+r*sin(theta(i));  % con esto a lo largo del bucle vamos
consiguiendo los puntos xyz que definen
    z(i) = origen(3);                  % la circunferencia en el espacio y que
se entregan a la Cinemática Inversa

    q1(i) = atan2(y(i),x(i));      %Se van entregan los valores xyz que obtenemos
de la circunferencia a la CinInv
    b(i) = z(i)-L1;                 %para que esta calcule los valores q1,q2,q3
articulares y se consiga así las
    c(i) = L2;                     %actuaciones articulares por parte del robot
para realizar la trayectoria
    a(i) = cos(q1(i))*x(i)+sin(q1(i))*y(i);
    q2(i) = atan2(L2,-sqrt(a(i)^2+b(i)^2-c(i)^2))-atan2(a(i),b(i));
    q3(i) = sin(q2(i))*(y(i)*sin(q1(i))+x(i)*cos(q1(i)))+cos(q2(i))*(L1-z(i));
end
q = [q1,q2,q3];

subplot(3,1,1);plot(theta,q1);grid;
title('q1; Variable articular de rotación correspondiente a la Articulación
1');xlabel('theta [rad]');ylabel('q1 [rad]');legend('q1');
subplot(3,1,2);plot(theta,q2);grid;
title('q2; Variable articular de rotación correspondiente a la Articulación
2');xlabel('theta [rad]');ylabel('q2 [rad]');legend('q2');
subplot(3,1,3);plot(theta,q3);grid;
title('q3; Variable articular de traslación correspondiente a la Articulación
3');xlabel('theta [rad]');ylabel('q3 [m]');legend('q3');

clear B
%             th    d       a         alpha
B(1) = Link([ 0     L1      0          pi/2    0], 'standard');
B(2) = Link([ 0     0       L2         pi/2    0], 'standard');
B(3) = Link([ 0     0       0          0       1], 'standard');
Brazo = SerialLink(B, 'name', 'Brazo');
qr_brazo = [pi/2 pi/2 (L3+L4)]; %postura dibujo brazo ("Home")

figure;
plot3(x,y,z);grid;axis([-0.2 0.5 -0.2 1.2 -0.4 1.4]);
hold on;
Brazo.plot(q, 'loop');
%Brazo.plot(qr_brazo);
```

- **Jacobianos:**

```
%Calculo Jacobianos
L1 = 0.8;
L2 = 0.4;
L3 = 0.6;
L4 = 0.4;
syms q1 q2 q3 real;
pi1 = sym(pi);
px = cos(q1)*((2*cos(q2))/5 + q3*sin(q2));
py = sin(q1)*((2*cos(q2))/5 + q3*sin(q2));
pz = (2*sin(q2))/5 - q3*cos(q2) + 4/5;

J = [diff(px,q1)    diff(px,q2)    diff(px,q3);
     diff(py,q1)    diff(py,q2)    diff(py,q3);
     diff(pz,q1)    diff(pz,q2)    diff(pz,q3)]

Jinv = simplify(inv(J))

Determinante = simplify(det(J))

Singularidad = simplify(0 == Determinante);
% De la linea anterior sacamos esto:
% 2*cos(q2) + 5*q3*sin(q2) == 0 ||||| q3 == 0

% Se ve que q3 = 0 es una singularidad clara, de la otra condición
%podemos ver que para ciertas combinaciones también se dan
singularidades:
%q3 = 2/5  ; q2 = 3*pi/4 + n*pi
%q3 = -2/5  ; q2 = pi/4 + n*pi
% No sé si hay más combinaciones
```

- **Representación del robot con Toolbox:**

```
%Representación Robot Toolbox
syms L1 L2 L3 L4 L1m L2m L3m L4m real;
syms q1 q2 q3 q4 q5 q6;
clear L
%             th    d       a         alpha
L(1) = Link([ 0     L1      0          pi/2   0], 'standard');
L(2) = Link([ 0     0       L2         pi/2   0], 'standard');
L(3) = Link([ 0     0       0          0      1], 'standard');
L(4) = Link([ 0     L2m     0          pi/2   0], 'standard');
L(5) = Link([ 0     0       L3m          0      0], 'standard');
L(6) = Link([ 0     0       L4m          0      0], 'standard');

% Comentar o descomentar B y L para representar el robot de 6gdl entero
% o el brazo de solo 3gdl
clear B
%             th    d       a         alpha
B(1) = Link([ 0     L1      0          pi/2   0], 'standard');
B(2) = Link([ 0     0       L2         pi/2   0], 'standard');
B(3) = Link([ 0     0       0          0      1], 'standard');


Brazo = SerialLink(B, 'name', 'Brazo');
Robot_C2 = SerialLink(L, 'name', 'Robot "C2"');

Tentera = simplify(Robot_C2.fkine([q1 q2 q3 q4 q5 q6]));
Tbrazo = simplify(Brazo.fkine([q1 q2 q3]));

%
% some useful poses
%
qz = [0 0 0 0 0 0];
qr = [pi/2 pi/2 0 pi/2 pi/2 0]; %postura dibujo robot 6gdl
qr_brazo = [pi/2 pi/2 (L3+L4)]; %postura dibujo brazo

%Robot_C2.plot(qr)
%Brazo.plot(qr_brazo)
```

- **Cálculo de parámetros dinámicos:**

```
%  Datos robot RRP del Proyecto (Brazo_C2)
  L0=0;  % Altura de la base
  L1=0.8;
  L2=0.4;
  L3=0.6;
  L4=0.4;

% Jm1=0.025; Jm2=Jm1; Jm3=Jm1;
% Bm1=3.6e-5; Bm2=Bm1; Bm3=Bm1;
% Kt1=25; Kt2=20; Kt3=25;
% R1=25; R2=20; R3=15;

% Eslabones macizos y de densidad constante
% Todos eslabones tienen sección circular ; Radio de la seccion
circular = L1/20
R = L1/20; % Radio de la sección
% Área (sección) del tubo
A=pi*(R^2);

% Suponemos la densidad volumétrica del material como 6000 kg/m3
%rho = 6000; % kg/m3
% Densidad lineal del material
%rhoL = rho*A; % kg/m



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ESLABÓN 3 (Sin desdoble, 1 barra (longitud L4)) alineado con DH
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M3= 3.2; %kg
X3cdm=0;
Y3cdm=0;
Z3cdm= -L4/2;
Pcdm3 = [X3cdm;Y3cdm;Z3cdm];

I3xx=(1/12)*M3*(3*(R^2)+(L4)^2);
I3yy=I3xx;
I3zz=0.5*M3*R^2;
I3=[I3xx, 0, 0; 0, I3yy, 0; 0, 0, I3zz];

Jm3=min([I3(1,1),I3(2,2),I3(3,3)]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ESLABÓN 2 (Con desdoble, 3 barras (longitudes L3/2; L2; L3/2))
alineado
% con DH
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M2 = 4; %kg
L2A = L3/2;
L2B = L2;
L2C = L3/2;

%  ESLABÓN 2A
M2A= M2*(L2A/(L2A+L2B+L2C));
I2Axx=(1/12)*M2A*(3*(R^2)+(L2A)^2);
I2Ayy=I2Axx;
I2Azz=0.5*M2A*R^2;
I2A=[I2Axx, 0, 0; 0, I2Ayy, 0; 0, 0, I2Azz];
```

```
%  ESLABÓN 2B
M2B= M2*(L2B/(L2A+L2B+L2C));
I2Bxx=0.5*M2B*R^2;
I2Byy=(1/12)*M2B*(3*(R^2)+(L2B)^2);
I2Bzz=I2Byy;
I2B=[I2Bxx, 0, 0; 0, I2Byy, 0; 0, 0, I2Bzz];

%  ESLABÓN 2C
M2C= M2*(L2C/(L2A+L2B+L2C));
I2Cxx=(1/12)*M2C*(3*(R^2)+(L2C)^2);
I2Cyy=I2Cxx;
I2Czz=0.5*M2C*R^2;
I2C=[I2Cxx, 0, 0; 0, I2Cyy, 0; 0, 0, I2Czz];

% Centro de masas del eslabón
M2=M2A+M2B+M2C;
Pcdm2A=[-L2; 0; L3/4];
Pcdm2B=[-L2/2; 0; L3/2];
Pcdm2C = [0; 0; L3*(3/4)];
Pcdm2=(M2A*Pcdm2A+M2B*Pcdm2B+M2C*Pcdm2C)/(M2A+M2B+M2C);

% Vector desde cdm2A a cdm2
Pcdm2A_cdm2=Pcdm2-Pcdm2A;
% Inercia de parte 2A respecto a cdm2
%  rx=Pcdm2A_cdm2(1); ry=Pcdm2A_cdm2(2); rz=Pcdm2A_cdm2(3);
%  RSteiner2A=[ry^2+rz^2, -rx*ry, -rx*rz; -rx*ry, rx^2+rz^2, -ry*rz; -
rx*rz, -ry*rz, rx^2+ry^2];
% Pero más compacto
DespSteiner2A=norm(Pcdm2A_cdm2)^2*eye(3)-Pcdm2A_cdm2*Pcdm2A_cdm2';
I2Acdm2=I2A+M2A*DespSteiner2A;

% Vector desde cdm2B a cdm2
Pcdm2B_cdm2=Pcdm2-Pcdm2B;
% Inercia de parte 2B respecto a cdm2
%  rx=Pcdm2B_cdm2(1); ry=Pcdm2B_cdm2(2); rz=Pcdm2B_cdm2(3);
%  RSteiner2B=[ry^2+rz^2, -rx*ry, -rx*rz; -rx*ry, rx^2+rz^2, -ry*rz; -
rx*rz, -ry*rz, rx^2+ry^2];
% Pero más compacto
DespSteiner2B=norm(Pcdm2B_cdm2)^2*eye(3)-Pcdm2B_cdm2*Pcdm2B_cdm2';
I2Bcdm2=I2B+M2B*DespSteiner2B;

% Vector desde cdm2C a cdm2
Pcdm2C_cdm2=Pcdm2-Pcdm2C;
% Inercia de parte 2C respecto a cdm2
%  rx=Pcdm2C_cdm2(1); ry=Pcdm2C_cdm2(2); rz=Pcdm2C_cdm2(3);
%  RSteiner2C=[ry^2+rz^2, -rx*ry, -rx*rz; -rx*ry, rx^2+rz^2, -ry*rz; -
rx*rz, -ry*rz, rx^2+ry^2];
% Pero más compacto
DespSteiner2C=norm(Pcdm2C_cdm2)^2*eye(3)-Pcdm2C_cdm2*Pcdm2C_cdm2';
I2Ccdm2=I2C+M2C*DespSteiner2C;

% Inercia del eslabón 2 completo sobre cdm
I2cdm2=I2Acdm2+I2Bcdm2+I2Ccdm2;
Jm2=min([I2cdm2(1,1),I2cdm2(2,2),I2cdm2(3,3)]);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ESLABÓN 1 (Sin desdoble, 1 barra (longitud L1/2)) alineado con DH
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M1= 4.5; %kg
X1cdm= 0;
Y1cdm= -L1/4;
Z1cdm= 0;
Pcdm1 = [X1cdm;Y1cdm;Z1cdm];

I1xx=(1/12)*M1*(3*(R^2)+(L1/2)^2);
I1zz=I1xx;
I1yy=0.5*M1*R^2;
I1=[I1xx, 0, 0; 0, I1yy, 0; 0, 0, I1zz];

Jm1=min([I1(1,1),I1(2,2),I1(3,3)]);
```

- **Método de Newton-Euler para obtener el modelo dinámico:**

```
%NEWTON-EULER PARA ROBOT PROYECTO (C2) 3GDL
% Robot RRP
% Elegir entre R (rotación) y P (prismática)
Tipo_Q1 = 'R'; % A modo de ejemplo
Tipo_Q2 = 'R';
Tipo_Q3 = 'P';

if ( (Tipo_Q1 ~= 'R') & (Tipo_Q1 ~='P')); error('Elegir R o P para
Tipo_Q1'); end;
if ( (Tipo_Q2 ~= 'R') & (Tipo_Q2 ~='P')); error('Elegir R o P para
Tipo_Q2'); end;
if ( (Tipo_Q3 ~= 'R') & (Tipo_Q3 ~='P')); error('Elegir R o P para
Tipo_Q3'); end;


% Definición de variables simbólicas
syms T1 T2 T3 q1 qd1 qdd1 q2 qd2 qdd2 q3 qd3 qdd3 g real
PI = sym(pi); % Importante para cáculo simbólico

% DATOS CINEMÁTICOS DEL BRAZO DEL ROBOT
% Dimensiones (m)
  L0=0;  % Altura de la base
  L1=0.8;
  L2=0.4;
  L3=0.6;
  L4=0.4;

% Parámetros de Denavit-Hartenberg (utilizado en primera regla de
Newton-Euler)
% Eslabón base (no utilizado)
% PARTIENDO DEL ROBOT EN LA "POSICIÓN DE DIBUJO":
  theta0=0; d0=L0; a0=0; alpha0=0;
% Eslabón 1:
  theta1= q1+PI/2; d1= L1; a1=0; alpha1=PI/2;         %Offset q1
(+pi/2)
% Eslabón 2:
  theta2= q2+PI/2; d2=0; a2= L2; alpha2= PI/2;        %Offset q2
(+pi/2)
% Eslabón 3:
  theta3= 0; d3= q3+L3+L4; a3= 0; alpha3= 0;          %Offset q3
(L3+L4)
% Entre eslabón 3 y marco donde se ejerce la fuerza (a definir según
% experimento)
  theta4= 0; d4= 0; a4= 0; alpha4= 0;

% DATOS DINÁMICOS DEL BRAZO DEL ROBOT
% Eslabón 1
  m1= 4.5; % kg
  s11 = [ 0,  -0.2, 0]'; % m
  I11=[0.0618,   0,          0;...
       0,        0.0036,     0 ;...
       0,        0,          0.0618]; % kg.m2

% Eslabón 2
  m2= 4; % kg
  s22 = [ -0.2,  0, 0.3]'; % m
  I22=[0.0742,        0,          -0.072;...
       0,             0.1909,     0 ;...
```

```
       -0.072,        0,           0.1199]; % kg.m2

% Eslabón 3
  m3= 3.2; % kg
  s33 = [ 0,   0, -0.2]'; % m
  I33=[0.0439,  0,            0;...
       0,         0.0439,     0 ;...
       0,         0,           0.0026]; % kg.m2

% DATOS DE LOS MOTORES
% Inercias
  Jm1= 0.0036; Jm2=0.0742; Jm3=0.0026; % kg.m2
% Coeficientes de fricción viscosa
  Bm1= 3.6e-5; Bm2= 3.6e-5; Bm3= 3.6e-5; % N.m / (rad/s)
% Factores de reducción
  R1= 25; R2= 20; R3= 25;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ALGORITMO DE NEWTON-EULER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% wij : velocidad angular absoluta de eje j expresada en i
% wdij : aceleración angular absoluta de eje j expresada en i
% vij : velocidad lineal absoluta del origen del marco j expresada en
i
% vdij : aceleración lineal absoluta del origen del marco j expresada
en i
% aii : aceleración del centro de gravedad del eslabón i, expresado en
i?

% fij : fuerza ejercida sobre la articulación j-1 (unión barra j-1 con
j),
% expresada en i-1
%
% nij : par ejercido sobre la articulación j-1 (unión barra j-1 con
j),
% expresada en i-1

% pii : vector (libre) que une el origen de coordenadas de i-1 con el
de i,
% expresadas en i : [ai, di*sin(alphai), di*cos(alphai)] (a,d,alpha:
parámetros de DH)
%
% sii : coordenadas del centro de masas del eslabón i, expresada en el
sistema
% i

% Iii : matriz de inercia del eslabón i expresado en un sistema
paralelo al
% i y con el origen en el centro de masas del eslabón
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% N-E 1: Asignación a cada eslabón de sistema de referencia de acuerdo
con las normas de D-H.
  % Eslabón 1:
    p11 = [a1, d1*sin(alpha1), d1*cos(alpha1)]';
  % Eslabón 2:
    p22 = [a2, d2*sin(alpha2), d2*cos(alpha2)]';
  % Eslabón 3:
    p33 = [a3, d3*sin(alpha3), d3*cos(alpha3)]';
```

```
  % Entre eslabón 2 y marco donde se ejerce la fuerza (supongo que el
mismo
  % que el Z0
    p44 = [a4, d4*sin(alpha4), d4*cos(alpha4)]';

% N-E 2: Condiciones iniciales de la base
  w00=[0 0 0]';
  wd00 = [0 0 0]';
  v00 = [0 0 0]';
  vd00 = [0 0 g]'; % Aceleración de la gravedad en el eje Z0 negativo

% Condiciones iniciales para el extremo del robot
% PONER FUERZAS SALIENTES DEL ESLABÓN (= FUERZAS ENTRANTES CAMBIADAS
DE
% SIGNO)
  f44= [0 0 0]';
  n44= [0 0 0]';

% Definición de vector local Z
  Z=[0 0 1]';


% N-E 3: Obtención de las matrices de rotación (i)R(i-1) y de sus
inversas
  R01=[cos(theta1) -cos(alpha1)*sin(theta1) sin(alpha1)*sin(theta1);
      sin(theta1)  cos(alpha1)*cos(theta1)  -sin(alpha1)*cos(theta1);
      0            sin(alpha1)                cos(alpha1)           ];
  R10= R01';

  R12=[cos(theta2) -cos(alpha2)*sin(theta2) sin(alpha2)*sin(theta2);
      sin(theta2)  cos(alpha2)*cos(theta2)  -sin(alpha2)*cos(theta2);
      0            sin(alpha2)              cos(alpha2)           ];
  R21= R12';

  R23=[cos(theta3) -cos(alpha3)*sin(theta3) sin(alpha3)*sin(theta3);
      sin(theta3)  cos(alpha3)*cos(theta3)  -sin(alpha3)*cos(theta3);
      0            sin(alpha3)              cos(alpha3)           ];
  R32= R23';

  R34=[cos(theta4) -cos(alpha4)*sin(theta4) sin(alpha4)*sin(theta4);
      sin(theta4)  cos(alpha4)*cos(theta4)  -sin(alpha4)*cos(theta4);
      0            sin(alpha4)              cos(alpha4)           ];
  R43= R34';


%%%%%%% ITERACIÓN HACIA EL EXTERIOR (CINEMÁTICA) %%%%%%%

% N-E 4: Obtención de las velocidades angulares absolutas
 % Articulación 1
    if (Tipo_Q1=='R');
        w11= R10*(w00+Z*qd1);  % Si es de rotación
    else
        w11 = R10*w00;       % Si es de translación
    end
 % Articulación 2
    if (Tipo_Q2=='R');
        w22= R21*(w11+Z*qd2);  % Si es de rotación
    else
        w22 = R21*w11;       % Si es de translación
```

```
    end
 % Articulación 3
    if (Tipo_Q3=='R');
        w33= R32*(w22+Z*qd3);  % Si es de rotación
    else
        w33 = R32*w22;      % Si es de translación
    end

% N-E 5: Obtención de las aceleraciones angulares absolutas
 % Articulación 1
    if (Tipo_Q1=='R');
        wd11 = R10*(wd00+Z*qdd1+cross(w00,Z*qd1));  % si es de
rotación
    else
        wd11 = R10*wd00;                            % si es de
translación
    end
 % Articulación 2
     if (Tipo_Q2=='R');
         wd22 = R21*(wd11+Z*qdd2+cross(w11,Z*qd2));  % si es de
rotación
     else
         wd22 = R21*wd11;                            % si es de
translación
     end
 % Articulación 3
     if (Tipo_Q3=='R');
         wd33 = R32*(wd22+Z*qdd3+cross(w22,Z*qd3));  % si es de
rotación
     else
         wd33 = R32*wd22;                            % si es de
translación
     end

% N-E 6: Obtención de las aceleraciones lineales de los orígenes de
los
% sistemas
 % Articulación 1
     if (Tipo_Q1=='R');
         vd11 = cross(wd11,p11)+cross(w11,cross(w11,p11))+R10*vd00;  %
si es de rotación
     else
         vd11 =
R10*(Z*qdd1+vd00)+cross(wd11,p11)+2*cross(w11,R10*Z*qd1) +
cross(w11,cross(w11,p11));    % si es de translación
     end
 % Articulación 2
     if (Tipo_Q2=='R');
         vd22 = cross(wd22,p22)+cross(w22,cross(w22,p22))+R21*vd11;  %
si es de rotación
     else
         vd22 =
R21*(Z*qdd2+vd11)+cross(wd22,p22)+2*cross(w22,R21*Z*qd2) +
cross(w22,cross(w22,p22));    % si es de translación
     end
 % Articulación 3
     if (Tipo_Q3=='R');
         vd33 = cross(wd33,p33)+cross(w33,cross(w33,p33))+R32*vd22;  %
si es de rotación
     else
```

```
        vd33 =
R32*(Z*qdd3+vd22)+cross(wd33,p33)+2*cross(w33,R32*Z*qd3) +
cross(w33,cross(w33,p33));     % si es de translación
     end

% N-E 7: Obtención de las aceleraciones lineales de los centros de
gravedad
    a11 = cross(wd11,s11)+cross(w11,cross(w11,s11))+vd11;
    a22 = cross(wd22,s22)+cross(w22,cross(w22,s22))+vd22;
    a33 = cross(wd33,s33)+cross(w33,cross(w33,s33))+vd33;


%%%%%%% ITERACIÓN HACIA EL INTERIOR (DINÁMICA)

% N-E 8: Obtención de las fuerzas ejercidas sobre los eslabones
  f33=R34*f44+m3*a33;
  f22=R23*f33+m2*a22;
  f11=R12*f22+m1*a11;

% N-E 9: Obtención de los pares ejercidas sobre los eslabones
  n33 =
R34*(n44+cross(R43*p33,f44))+cross(p33+s33,m3*a33)+I33*wd33+cross(w33,
I33*w33);
  n22 =
R23*(n33+cross(R32*p22,f33))+cross(p22+s22,m2*a22)+I22*wd22+cross(w22,
I22*w22);
  n11 =
R12*(n22+cross(R21*p11,f22))+cross(p11+s11,m1*a11)+I11*wd11+cross(w11,
I11*w11);

% N-E 10: Obtener la fuerza o par aplicado sobre la articulación
  N3z = n33'*R32*Z;  % Si es de rotación
  N3  = n33'*R32;    % Para ver todos los pares, no solo el del eje Z
  F3z = f33'*R32*Z;  % Si es de translacion;
  F3  = f33'*R32;    % Para ver todas las fuerzas, no solo la del eje
Z
  N2z = n22'*R21*Z;  % Si es de rotación
  N2  = n22'*R21;    % Para ver todos los pares, no solo el del eje Z
  F2z = f22'*R21*Z;  % Si es de translacion;
  F2  = f22'*R21;    % Para ver todas las fuerzas, no solo la del eje
Z
  N1z = n11'*R10*Z;  % Si es de rotación
  N1  = n11'*R10;    % Para ver todos los pares, no solo el del eje Z
  F1z = f11'*R10*Z;  % Si es de translacion;
  F1  = f11'*R10;    % Para ver todas las fuerzas, no solo la del eje
Z

% Robot RRR o PPP
    if (Tipo_Q1=='R'); T1=N1z; else T1=F1z; end
    if (Tipo_Q2=='R'); T2=N2z; else T2=F2z; end
    if (Tipo_Q3=='R'); T3=N3z; else T3=F3z; end

%%% MANIPULACIÓN SIMBÓLICA DE LAS ECUACIONES %%%
% En ecuaciones matriciales (solo parte del brazo):
%
% T= M(q)qdd+V(q,qd)+G(q)  = M(q)qdd+VG(q,qd)
%

% Primera ecuación
% -----------------
```

```
% Cálculo de los términos de la matriz de inercia (afines a qdd)
M11 = diff(T1,qdd1);
Taux = simplify(T1 - M11*qdd1);
M12 = diff(Taux,qdd2);
Taux = simplify(Taux-M12*qdd2);
M13= diff(Taux,qdd3);
Taux = simplify(Taux-M13*qdd3);
% Taux restante contiene términos Centrípetos/Coriolis y Gravitatorios
% Términos gravitatorios: dependen linealmente de "g"
G1=diff(Taux,g)*g;
Taux=simplify(Taux-G1);
% Taux restante contiene términos Centrípetos/Coriolis
V1=Taux;

% Segunda ecuación
% -----------------
% Cálculo de los términos de la matriz de inercia (afines a qdd)
M21 = diff(T2,qdd1);
Taux = simplify(T2 - M21*qdd1);
M22 = diff(Taux,qdd2);
Taux = simplify(Taux-M22*qdd2);
M23 = diff(Taux,qdd3);
Taux = simplify(Taux-M23*qdd3);
% Taux restante contiene términos Centrípetos/Coriolis y Gravitatorios
% Términos gravitatorios: dependen linealmente de "g"
G2=diff(Taux,g)*g;
Taux=simplify(Taux-G2);
% Taux restante contiene términos Centrípetos/Coriolis
V2=Taux;

% Tercera ecuación
% -----------------
% Cálculo de los términos de la matriz de inercia (afines a qdd)
M31 = diff(T3,qdd1);
Taux = simplify(T3 - M31*qdd1);
M32 = diff(Taux,qdd2);
Taux = simplify(Taux-M32*qdd2);
M33 = diff(Taux,qdd3);
Taux = simplify(Taux-M33*qdd3);
% Taux restante contiene términos Centrípetos/Coriolis y Gravitatorios
% Términos gravitatorios: dependen linealmente de "g"
G3=diff(Taux,g)*g;
Taux=simplify(Taux-G3);
% Taux restante contiene términos Centrípetos/Coriolis
V3=Taux;

% Simplificación de expresiones
M11=simplify(M11); M12=simplify(M12); M13=simplify(M13);
M21=simplify(M21); M22=simplify(M22); M23=simplify(M23);
M31=simplify(M31); M32=simplify(M32); M33=simplify(M33);

V1=simplify(V1); V2=simplify(V2); V3=simplify(V3);
G1=simplify(G1); G2=simplify(G2); G3=simplify(G3);

% Apilación en matrices y vectores
M = [M11 M12 M13; M21 M22 M23; M31 M32 M33];
V = [V1; V2; V3];
G = [G1; G2; G3];
```

```
% Inclusión de los motores en la ecuación dinámica
%
% T= Ma(q)qdd+Va(q,qd)+Ga(q)
%
% Ma = M + R^2*Jm      Va=V + R^2*Bm*qd      Ga=G
%
R=diag([R1 R2 R3]);
Jm=diag([Jm1 Jm2 Jm3]);
Bm=diag([Bm1 Bm2 Bm3]);
% Kt=diag([Kt1 Kt2 Kt3]); % No utilizado

Ma=M+R*R*Jm;
Va=V+R*R*Bm*[qd1 ; qd2 ; qd3];
Ga = G;

% La función vpa del Symbolic Toolbox evalua las expresiones de las
% fracciones de una función simbólica, redondeándolas con la precisión
que podría pasarse como segundo
% argumento.
 Ma_ne=vpa(Ma,5);
 Va_ne=vpa(Va,5);
 Ga_ne=vpa(G,5);
```

- **Código para implementar la dinámica del Robot en Simulink con Robotics Toolbox (crea objeto SerialLink, con la dinámica de mi robot):**

```
% Dinámica Proyecto con Robotics Toolbox v9 para robot(C2) de 3GDL

% Elegir entre R (rotación) y P (prismática)
Tipo_Q1 = 'R'; % A modo de ejemplo
Tipo_Q2 = 'R';
Tipo_Q3 = 'P';

if ( (Tipo_Q1 ~= 'R') & (Tipo_Q1 ~='P')); error('Elegir R o P para
Tipo_Q1'); end;
if ( (Tipo_Q2 ~= 'R') & (Tipo_Q2 ~='P')); error('Elegir R o P para
Tipo_Q2'); end;
if ( (Tipo_Q3 ~= 'R') & (Tipo_Q3 ~='P')); error('Elegir R o P para
Tipo_Q3'); end;


% Defición genérica del reposo (no interviene en el modelo)
q1=0; q2=0; q3=0;

% DATOS CINEMÁTICOS DEL BRAZO DEL ROBOT
% Dimensiones (m)
  L0=0;  % Altura de la base
  L1=0.8;
  L2=0.4;
  L3=0.6;
  L4=0.4;


% Parámetros de Denavit-Hartenberg (utilizado en primera regla de
Newton-Euler)
% Eslabón base (no utilizado)
```

```
Offset_1 = pi/2;
Offset_2 = pi/2;
Offset_3 = L3+L4;
  theta0=0; d0=L0; a0=0; alpha0=0;
% Eslabón 1:
  theta1= q1; d1= L1; a1=0; alpha1=pi/2;
% Eslabón 2:
  theta2= q2; d2=0; a2= L2; alpha2= pi/2;
% Eslabón 3:
  theta3= 0; d3= q3; a3= 0; alpha3= 0;
% Entre eslabón 3 y marco donde se ejerce la fuerza (a definir según
% experimento)
  theta4= 0; d4= 0; a4= 0; alpha4= 0;

% DATOS DINÁMICOS DEL BRAZO DEL ROBOT
% Eslabón 1
  m1= 4.5; % kg
  s11 = [ 0,  -0.2, 0]'; % m
  I11=[0.0618,   0,           0;...
       0,        0.0036,      0 ;...
       0,        0,           0.0618]; % kg.m2

% Eslabón 2
  m2= 4; % kg
  s22 = [ -0.2,  0, 0.3]'; % m
  I22=[0.0742,        0,           -0.072;...
       0,             0.1909,      0 ;...
       -0.072,        0,           0.1199]; % kg.m2

% Eslabón 3
  m3= 3.2; % kg
  s33 = [ 0,  0, -0.2]'; % m
  I33=[0.0439,  0,           0;...
       0,        0.0439,     0 ;...
       0,        0,           0.0026]; % kg.m2


% DATOS DE LOS MOTORES
% Inercias
  Jm1= 0.0036; Jm2=0.0742; Jm3=0.0026; % kg.m2
% Coeficientes de fricción viscosa
  Bm1= 3.6e-5; Bm2= 3.6e-5; Bm3= 3.6e-5; % N.m / (rad/s)
% Factores de reducción
  R1= 25; R2= 20; R3= 25;

% Aceleración de la gravedad
g = 9.8;

% En caso de no ulilizar variables simbólicas
% Parámetros de Denavit-Hartemberg  DH = [THETA D A ALPHA SIGMA
OFFSET]
%                                                     SIGMA(0:R, 1:T)
%             theta_i  d_i  a_i  alpha_i  sigma offset   standard
% Articulación 1
 if (Tipo_Q1=='R')
     L(1) = Link([theta1   d1   a1   alpha1   0      Offset_1 ],
'standard');
 else
     L(1) = Link([theta1   d1   a1   alpha1   1      Offset_1 ],
'standard');
```

```
 end
 % Articulación 2
 if (Tipo_Q2=='R');
     L(2) = Link([theta2   d2   a2   alpha2   0      Offset_2 ],
'standard');
 else
     L(2) = Link([theta2   d2   a2   alpha2   1      Offset_2 ],
'standard');
 end
 % Articulación 3
 if (Tipo_Q3=='R');
     L(3) = Link([theta3   d3   a3   alpha3   0      Offset_3 ],
'standard');
 else
     L(3) = Link([theta3   d3   a3   alpha3   1      Offset_3 ],
'standard');
 end



% Definición de los parámetros dinámicos de los eslabones
% Masas
L(1).m = m1;
L(2).m = m2;
L(3).m = m3;
% Posición del centro de gravedad respecto al sistema de ref. local
%                 rx      ry      rz
L(1).r = s11'; % [0     0       0 ];  % Cuidado con el signo -
L(2).r = s22'; % [0     0       0 ];
L(3).r = s33'; % [0     0       0 ];

% Parámetros de inercia respecto al sistema de referencia local
%         Ixx     Iyy      Izz     Ixy     Iyz     Ixz
%L(1).I = [I1xx    I1yy     I1zz    I1xy    I1xz    I1yz];
% Tambien admite: L(1).I = [0 0 0; 0 0 0; 0 0 0.35];
L(1).I = I11;
L(2).I = I22;
L(3).I = I33;

% Inercia del actuador
L(1).Jm =  Jm1;
L(2).Jm =  Jm2;
L(3).Jm =  Jm3;

% Fricción viscosa del actuador
L(1).B =  Bm1;
L(2).B =  Bm2;
L(3).B =  Bm3;

% Relación de transformación de la reductora
L(1).G =  R1; % Accionamiento directo
L(2).G =  R2;
L(3).G =  R3;

% Definición del robot con los datos anteriores
R3GDL = SerialLink(L, 'name','ROBOT DE 3GDL');

R3GDL.gravity=[0 0 9.8]';
clear L %Borramos L ya que no lo necesitaremos más
```

- **Función “ModeloDinamico_R3GDL”:**

```
function [qdd] = ModeloDinamico_R3GDL(in)

% Variables de entrada en la funcion: [q(3)  qp(3)  Tau(3)]
q1         = in(1);
q2         = in(2);
q3         = in(3);
qd1        = in(4);
qd2        = in(5);
qd3        = in(6);
Tau1       = in(7);
Tau2       = in(8);
Tau3       = in(9);


% Matriz de Inercias
Ma = [5.12*q3*cos(q2)^2 - 1.336*sin(2.0*q2) + 1.7316*cos(q2)^2 -
1.28*q3*sin(2.0*q2) + 3.2*q3^2*cos(q2)^2 + 3.0481,
0,      0;...

0, (16*q3^2)/5 + (128*q3)/25 + 82487/2500, -32/25;...

0,                                   -32/25, 193/40];

% Matriz de aceleraciones centrípetas y de Coriolis
Va = [0.0225*qd1 - 0.0004*qd1*(6680.0*qd2*cos(2.0*q2) - 8000.0*q3*qd3
- 6400.0*qd3 - 6400.0*qd3*cos(2.0*q2) + 4329.0*qd2*sin(2.0*q2) +
3200.0*qd3*sin(2.0*q2) + 8000.0*q3^2*qd2*sin(2.0*q2) +
6400.0*q3*qd2*cos(2.0*q2) - 8000.0*q3*qd3*cos(2.0*q2) +
12800.0*q3*qd2*sin(2.0*q2));...
                                        0.0144*qd2 + 5.12*qd2*qd3 +
1.336*qd1^2*cos(2.0*q2) + 0.8658*qd1^2*sin(2.0*q2) +
1.28*q3*qd1^2*cos(2.0*q2) + 2.56*q3*qd1^2*sin(2.0*q2) +
1.6*q3^2*qd1^2*sin(2.0*q2) + 6.4*q3*qd2*qd3;...

0.0225*qd3 - 2.56*qd1^2*cos(q2)^2 - 3.2*q3*qd2^2 +
0.64*qd1^2*sin(2.0*q2) - 2.56*qd2^2 - 3.2*q3*qd1^2*cos(q2)^2];


% Par gravitatorio
g = 9.81;
Ga =      [0;...
g*(3.76*cos(q2) - 2.08*sin(q2) + 3.2*q3*cos(q2));...
                                         3.2*g*sin(q2)];

% Ecuación del robot
%     Tau = M*qpp + V + G
  Tau=[Tau1;Tau2;Tau3];

% Por lo que:
% Aceleraciones
  qdd = inv(Ma)*(Tau-Va-Ga);
```

- **Código cinemática directa_GTCL:**

```
function xyz = CinematicaDirecta(in)

q1       = in(1);           % Posición articular
q2       = in(2);           %
q3       = in(3);

  % A rellenar por el alumno

  L1 = 0.8;
  L2 = 0.4;
  L3 = 0.6;
  L4 = 0.4;

  x = cos(q1)*(L2*cos(q2)+q3*sin(q2));
  y = sin(q1)*(L2*cos(q2)+q3*sin(q2));
  z = L1+L2*sin(q2)-q3*cos(q2);


xyz=[x;y;z];
```

- **Código cinemática inversa_GTCL:**

```
function q = CinematicaInversa(in)

x       = in(1);           % Posición cartesianas
y       = in(2);           %
z       = in(3);           %

% A rellenar por el alumno
% Al tener mi cinemática inversa 2 ramas de soluciones he tomado
% arbitrariamente una de ellas. Tomando como referencia el esquema del
% anexo se toma la rama q1 --> q2_1 --> q3_1;
L1 = 0.8;
L2 = 0.4;
L3 = 0.6;
L4 = 0.4;

q1 = atan2(y,x);
b = z-L1;
a = cos(q1)*x+sin(q1)*y;
c = L2;
q2 = atan2(L2,+sqrt(a^2+b^2-c^2))-atan2(a,b);
q3 = sin(q2)*(y*sin(q1)+x*cos(q1))+cos(q2)*(L1-z);

q=[q1;q2;q3];
```

- **Códigos gráficas apartado GTCL:**

```
% Apartado GTCL
   %    Trayectorias 3D
%  plot3(xr,yr,zr,[Xini Xfin],[Yini Yfin],[Zini
Zfin],'LineWidth',1.5);title('Trayectoria Deseada (alimentada al GTCL)
VS Trayectoria Realizada (mediante MCD) (con 500 puntos
intermedios)');xlabel('Eje-X [m]');ylabel('Eje-Y [m]');zlabel('Eje-Z
[m]');legend('Trayectoria Realizada','Trayectoria Deseada');
%  grid on;

    %   Trayectorias 2D plano XY
% plot3(xr,yr,zr,[Xini Xfin],[Yini Yfin],[Zini Zfin],'LineWidth',1.5);
view(0,90);title('Trayectoria Deseada (alimentada al GTCL) VS
Trayectoria Realizada (mediante MCD) (con 500 puntos
intermedios)');xlabel('Eje-X [m]');ylabel('Eje-Y [m]');zlabel('Eje-Z
[m]');legend('Trayectoria Realizada','Trayectoria Deseada');
% grid on;

    %   plot 2D real para zoom
% plot(xr,yr,[Xini Xfin],[Yini Yfin]);title('Trayectoria Deseada
(alimentada al GTCL) VS Trayectoria Realizada (mediante MCD) (con 500
puntos intermedios)');xlabel('Eje-X [m]');ylabel('Eje-Y
[m]');zlabel('Eje-Z [m]');legend('Trayectoria Realizada','Trayectoria
Deseada');
% grid on;

    %   Subplot trayectorias 3D por ejes
% subplot(3,1,1);plot(tout,xr,[Xini Xfin]);grid;title('Diferencia
Trayectoria deseada VS realizada en el eje
X');xlabel('tiempo(s)');ylabel('Eje-X [m]');legend('Trayectoria
Realizada','Trayectoria Deseada');
% subplot(3,1,2);plot(tout,yr,[Yini Yfin]);grid;title('Diferencia
Trayectoria deseada VS realizada en el eje
Y');xlabel('tiempo(s)');ylabel('Eje-Y [m]');legend('Trayectoria
Realizada','Trayectoria Deseada');
% subplot(3,1,3);plot(tout,zr,[Zini Zfin]);grid;title('Diferencia
Trayectoria deseada VS realizada en el eje
Z');xlabel('tiempo(s)');ylabel('Eje-Z [m]');legend('Trayectoria
Realizada','Trayectoria Deseada');

    %   Gráfica error trayectoria 3D (esta forma necesita que el punto
vaya de "menor a mayor" en sus coordenadas: Xini<Xfin; Yini<Yfin;
Yini<Yfin)
% x_des = [Xini:((Xfin-Xini)/(length(xr)-1)):Xfin]';
% y_des = [Yini:((Yfin-Yini)/(length(yr)-1)):Yfin]';
% z_des = [Zini:((Zfin-Zini)/(length(zr)-1)):Zfin]';
% subplot(3,1,1);plot(tout,xr,tout,x_des);grid;title('Diferencia
Trayectoria deseada VS realizada en el eje
X');xlabel('tiempo(s)');ylabel('Posición [m]');legend('Diferencia-
EjeX');
% subplot(3,1,2);plot(tout,yr,tout,y_des);grid;title('Diferencia
Trayectoria deseada VS realizada en el eje
Y');xlabel('tiempo(s)');ylabel('Posición [m]');legend('Diferencia-
EjeY');
% subplot(3,1,3);plot(tout,zr,tout,z_des);grid;title('Diferencia
Trayectoria deseada VS realizada en el eje
Z');xlabel('tiempo(s)');ylabel('Posición [m]');legend('Diferencia-
EjeZ');
```

```
    %   Variables articulares q,qd,qdd
% figure;
% subplot(3,1,1);plot(tout,qr);grid;title('q_i (posiciones articulares)');xlabel('tiempo(s)');ylabel('q1,q2 [rad] ; q3 [m]');legend('q1','q2','q3');
% subplot(3,1,2);plot(tout,qdr);grid;title('qd_i (velocidades articulares)');xlabel('tiempo(s)');ylabel('q1,q2 [rad/s] ; q3 [m/s]');legend('qd1','qd2','qd3');
% subplot(3,1,3);plot(tout,qddr);grid;title('qdd_i (aceleraciones articulares)');xlabel('tiempo(s)');ylabel('q1,q2 [rad/s^2] ; q3 [m/s^2]');legend('qdd1','qdd2','qdd3');
```

- **Código control PD descentralizado con cancelación de dinámica:**

```
     % Cálculo de controladores PD con cancelación

% Funciones de transferencia
% Gii(s) = (1)/((ai*s+bi)*s)

a1 = 13.0997;  b1 = 0.0225;
a2 = 41.315;   b2 = 0.0144;
a3 = 4.825;    b3 = 0.0225;

% PD con cancelación
tsd = 0.01; % Tiempo de subida deseado

% Teóricamente buscamos Kci y Tdi, pero para Simulink (implementación)
% necesitamos Kpi y Kdi, obteniéndose ambos a continuación:

if(1)
    Kc1 = 3*b1/tsd;
    Td1 = a1/b1;
    Kp1 = Kc1;
    Kd1 = Kc1*Td1;

    Kc2 = 3*b2/tsd;
    Td2 = a2/b2;
    Kp2 = Kc2;
    Kd2 = Kc2*Td2;

    Kc3 = 3*b3/tsd;
    Td3 = a3/b3;
    Kp3 = Kc3;
    Kd3 = Kc3*Td3;

    Ki1 = 0;
    Ki2 = 0;     %Para anular el término integral en Simulink
    Ki3 = 0;
end
```

- **Código control PID descentralizado con cancelación de dinámica:**

```
% Cálculo de controladores PID con cancelación

% Funciones de transferencia
% Gii(s) = (1)/((ai*s+bi)*s)

a1 = 13.0997;  b1 = 0.0225;
a2 = 41.315;   b2 = 0.0144;
a3 = 4.825;    b3 = 0.0225;

% PID con cancelación
tsd = 0.01; % Tiempo de subida deseado

if(1)
    %Articulación 1
    tau1_1 = a1/b1;
    tau2_1 = tsd/3;
    Kc1 = (36*b1)/(tsd^2);
    Ki1 = Kc1;
    Kd1 = Ki1*tau1_1*tau2_1;
    Kp1 = (tau1_1+tau2_1)*Ki1;

    %Articulación 2
    tau1_2 = a2/b2;
    tau2_2 = tsd/3;
    Kc2 = (36*b2)/(tsd^2);
    Ki2 = Kc2;
    Kd2 = Ki2*tau1_2*tau2_2;
    Kp2 = (tau1_2+tau2_2)*Ki2;

    % Articulación 3
    tau1_3 = a3/b3;
    tau2_3 = tsd/3;
    Kc3 = (36*b3)/(tsd^2);
    Ki3 = Kc3;
    Kd3 = Ki3*tau1_3*tau2_3;
    Kp3 = (tau1_3+tau2_3)*Ki3;
end
```

- **Código control PD para Par Calculado:**

```
% Cálculo de controladores PD con par calculado

% Funciones de transferencia
% Gii(s) = (1)/((ai*s+bi)*s)

a1 = 13.0997;  b1 = 0.0225;
a2 = 41.315;   b2 = 0.0144;
a3 = 4.825;    b3 = 0.0225;

tsd = 0.01; % Tiempo de subida deseado

% Cálculo de PD para par calculado
if(1)
    Kp1 = 36/tsd^2;
    Kp2 = Kp1;
    Kp3 = Kp1;

    Kd1 = 12/tsd;
    Kd2 = Kd1;
    Kd3 = Kd1;

    Ki1 = 0;
    Ki2 = 0;
    Ki3 = 0;
end
```

- **Código bloque “Compensador” para controlador PD para Par Calculado:**

```
function [Tau_add] = Compensador(in)

% Variables de entrada en la funcion: [q(3)  qp(3)  Tau(3)]
q1        = in(1);
q2        = in(2);
q3        = in(3);
qd1       = in(4);
qd2       = in(5);
qd3       = in(6);
qddr1     = in(7);
qddr2     = in(8);
qddr3     = in(9);
u1     = in(10);
u2     = in(11);
u3     = in(12);


  % Matriz de Inercias
 Ma = [5.12*q3*cos(q2)^2 - 1.336*sin(2.0*q2) + 1.7316*cos(q2)^2 -
1.28*q3*sin(2.0*q2) + 3.2*q3^2*cos(q2)^2 + 3.0481,
0,      0;...

0, (16*q3^2)/5 + (128*q3)/25 + 82487/2500, -32/25;...

0,                                -32/25, 193/40];

% Matriz de aceleraciones centrípetas y de Coriolis
```

```
  Va = [0.0225*qd1 - 0.0004*qd1*(6680.0*qd2*cos(2.0*q2) -
8000.0*q3*qd3 - 6400.0*qd3 - 6400.0*qd3*cos(2.0*q2) +
4329.0*qd2*sin(2.0*q2) + 3200.0*qd3*sin(2.0*q2) +
8000.0*q3^2*qd2*sin(2.0*q2) + 6400.0*q3*qd2*cos(2.0*q2) -
8000.0*q3*qd3*cos(2.0*q2) + 12800.0*q3*qd2*sin(2.0*q2));...
                                        0.0144*qd2 + 5.12*qd2*qd3 +
1.336*qd1^2*cos(2.0*q2) + 0.8658*qd1^2*sin(2.0*q2) +
1.28*q3*qd1^2*cos(2.0*q2) + 2.56*q3*qd1^2*sin(2.0*q2) +
1.6*q3^2*qd1^2*sin(2.0*q2) + 6.4*q3*qd2*qd3;...

0.0225*qd3 - 2.56*qd1^2*cos(q2)^2 - 3.2*q3*qd2^2 +
0.64*qd1^2*sin(2.0*q2) - 2.56*qd2^2 - 3.2*q3*qd1^2*cos(q2)^2];



% Par gravitatorio
  g = 9.81;
Ga =      [0;...
g*(3.76*cos(q2) - 2.08*sin(q2) + 3.2*q3*cos(q2));...
                                         3.2*g*sin(q2)];


% Ecuación del robot
%    Tau = M*qpp + V + G
%  Tau=[Tau1;Tau2;Tau3];

% Compensación gravedad
%Tau_add=Ga;% Ma*[qddr1;qddr2;qddr3];

% Control con precompensación dinámica
% qddr = [qddr1; qddr2; qddr3];
% Tau_add = Ma*qddr+Va+Ga;

% Control por par calculado
qddr = [qddr1; qddr2; qddr3];
u = [u1; u2; u3];   % Par que genera el PD
Tau_add = Ma*(qddr+u) + Va + Ga;
```

- **Código genérico gráficas apartado controladores (se adjunta el de PD con par calculado pero son similares para los otros dos):**

```
% Códigos genéricos para graficar resultados de controladores

%%%%%%% CAMBIAR TITULOS Y LEYENDAS ENTRE PD, PID Y PAR CALCULADO
%%%%%%%%%%

% Comparativa de trayectorias referencia VS real (control)
%  plot3(xr,yr,zr,x,y,z);title('Trayectoria referencia VS Trayectoria
real para controlador PD con método de Par Calculado (ts =
0.01s)');xlabel('X(m)');ylabel('Y(m)');zlabel('Z(m)');legend('Ref','Re
al');grid

% Errores en cartesianas para xr-x,yr-y,zr-z
% subplot(3,1,1);plot(t,(xr-x).^2);title('Error cuadrático entre Xr y
X');xlabel('tiempo(s)');ylabel('Magnitud del Error
Cuadrático');legend('Error-X');grid;
```

```
% subplot(3,1,2);plot(t,(yr-y).^2);title('Error cuadrático entre Yr y Y');xlabel('tiempo(s)');ylabel('Magnitud del Error Cuadrático');legend('Error-Y');grid;
% subplot(3,1,3);plot(t,(zr-z).^2);title('Error cuadrático entre Zr y Z');xlabel('tiempo(s)');ylabel('Magnitud del Error Cuadrático');legend('Error-Z');grid;

% Pares aplicados a las articulaciones
%  plot(t,Tau(:,1),t,Tau(:,2),t,Tau(:,3));title('Pares aplicados a cada articulación (tau_i)');xlabel('tiempo(s)');ylabel('Par(Nm)');legend('tau1','tau2','tau3');grid

% Errores cuadráticos para q1,q2,q3
% subplot(3,1,1);plot(t,(qr(:,1)-q(:,1)).^2);title('Error cuadrático entre q1ref(GTCL) y q1real(ModeloDinamico)');xlabel('tiempo(s)');ylabel('q1(rad)');legend('Error-q1');grid;
% subplot(3,1,2);plot(t,(qr(:,2)-q(:,2)).^2);title('Error cuadrático entre q2ref(GTCL) y q2real(ModeloDinamico)');xlabel('tiempo(s)');ylabel('q2(rad)');legend('Error-q2');grid;
% subplot(3,1,3);plot(t,(qr(:,3)-q(:,3)).^2);title('Error cuadrático entre q3ref(GTCL) y q3real(ModeloDinamico)');xlabel('tiempo(s)');ylabel('q3(m)');legend('Error-q3');grid;

%Comparación q1,q2,q3
% subplot(3,1,1);plot(t,qr(:,1),t,q(:,1));grid;title('q1(R)');xlabel('tiempo(s)');ylabel('q1 [rad]');legend('q1-ref','q1-real');
% subplot(3,1,2);plot(t,qr(:,2),t,q(:,2));grid;title('q2(R)');xlabel('tiempo(s)');ylabel('q2 [rad]');legend('q2-ref','q2-real');
% subplot(3,1,3);plot(t,qr(:,3),t,q(:,3));grid;title('q3(P)');xlabel('tiempo(s)');ylabel('q3 [m]');legend('q3-ref','q3-real');
% figure;
%Comparación qd1,qd2,qd3
% subplot(3,1,1);plot(t,qdr(:,1),t,qd(:,1));grid;title('qd1(R)');xlabel('tiempo(s)');ylabel('qd1 [rad/s]');legend('qd1-ref','qd1-real');
% subplot(3,1,2);plot(t,qdr(:,2),t,qd(:,2));grid;title('qd2(R)');xlabel('tiempo(s)');ylabel('qd2 [rad/s]');legend('qd2-ref','qd2-real');
% subplot(3,1,3);plot(t,qdr(:,3),t,qd(:,3));grid;title('qd3(P)');xlabel('tiempo(s)');ylabel('qd3 [m/s]');legend('qd3-ref','qd3-real');
% figure;
%Comparación qdd1,qdd2,qdd3
% subplot(3,1,1);plot(t,qddr(:,1),t,qdd(:,1));grid;title('qdd1(R)');xlabel('tiempo(s)');ylabel('qdd1 [rad/s^2]');legend('qdd1-ref','qdd1-real');
% subplot(3,1,2);plot(t,qddr(:,2),t,qdd(:,2));grid;title('qdd2(R)');xlabel('tiempo(s)');ylabel('qdd2 [rad/s^2]');legend('qdd2-ref','qdd2-real');
% subplot(3,1,3);plot(t,qddr(:,3),t,qdd(:,3));grid;title('qdd3(P)');xlabel('tiempo(s)');ylabel('qdd3 [m/s^2]');legend('qdd3-ref','qdd3-real');
```