

Trabajo de Curso 2024/25

Fundamentos de Robótica



Alejandro Roche Aniento

Índice:

1. Introducción	4
2. Análisis Cinemático.....	4
2.1. Algoritmo Denavit-Hartenberg y tabla DH.....	5
2.2. Matrices de Transformación Homogéneas (MTH).....	7
2.3. Ecuaciones del modelo cinemático directo.....	9
2.4. Cinemática Inversa.....	12
2.5. Control Cinemático.....	18
2.6. Jacobianos.....	19
2.7. Información Complementaria.....	20
3. Análisis Dinámico	23
3.1. Dinámica Inversa.....	24
3.2. Simulador de la dinámica y del modelo de referencia	33
3.3. Resultado Modelo Dinámico	34
4. Control Cinemático	36
5. Control Dinámico.....	40
5.1. Diseño de controladores.....	42
5.1.1. PD descentralizado.....	42
5.1.2. PD descentralizado con compensación de gravedad...	44
5.1.3. PID descentralizado	45
5.1.4. De par calculado.....	47
5.2. Simulación de controladores.....	49
5.2.1. PD descentralizado.....	50
5.2.2. PD descentralizado con compensación de gravedad...	53

5.2.3. PID descentralizado	56
5.2.4. De par calculado.....	59
5.3. Análisis de robustez.....	62
5.3.1. PD descentralizado.....	63
5.3.2. PD descentralizado con compensación de gravedad...	66
5.3.3. PID descentralizado	69
5.3.4. De par calculado.....	72
6.Verificacion.....	75

1.- Introducción.

En primer lugar, se intenta plasmar los conocimientos adquiridos en teoría sobre análisis cinemático y análisis dinámico para ello se proporciona los parámetros cinemáticos de un robot de 6 grados de libertad con el que se trabajará en los siguientes apartados; para posteriormente reducir su complejidad a un robot de 3 grados, simplificando el contenido del contenido de las partes.

Una vez simplificado, se desarrollará el control cinemático y control dinámico del robot con el objetivo de mostrar la veracidad de la teoría aplicada en un robot ideal.

2.- Análisis Cinemático

El objetivo de este análisis es estudiar la relación entre la localización del extremo del robot y los valores de sus articulaciones además de una descripción analítica del movimiento espacial en función del tiempo.

Para ello se establece un estudio del movimiento del robot con respecto a un sistema de referencia sin tener en cuenta las fuerzas que puedan intervenir.

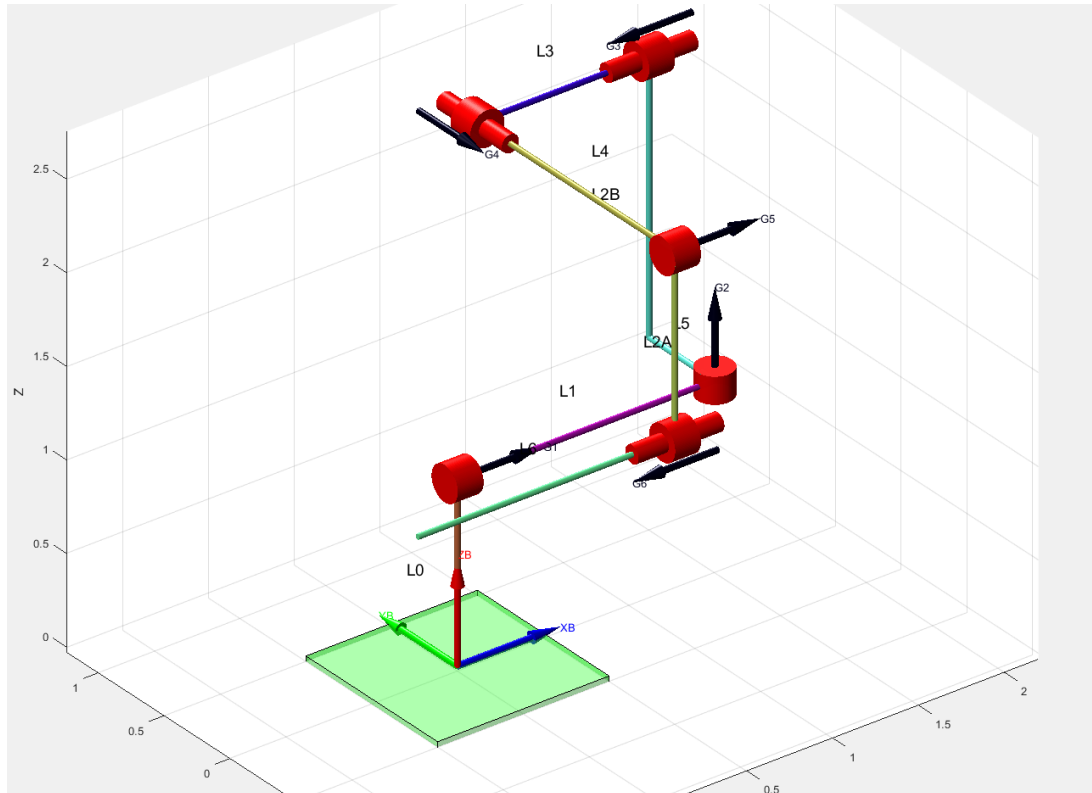
Para empezar, se plantea el problema cinemático directo en el que se determinará la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas de referencia, teniendo en cuenta los valores de sus articulaciones (q_1, q_2, \dots, q_6) y los parámetros geométricos de los elementos del robot (L_0, L_1, \dots, L_6).

Posteriormente, se hará un estudio del problema cinemático inverso donde se va a determinar la configuración que debe adoptar el robot para alcanzar una posición y orientación del extremo conocidas.

Más allá, se planteará el estudio de una trayectoria circular en el plano XY que pretende que el propio brazo robótico alcance a trazar.

Por último, se hará un estudio de la jacobiana del robot con el fin de poder determinar los puntos singulares del extremo del mismo.

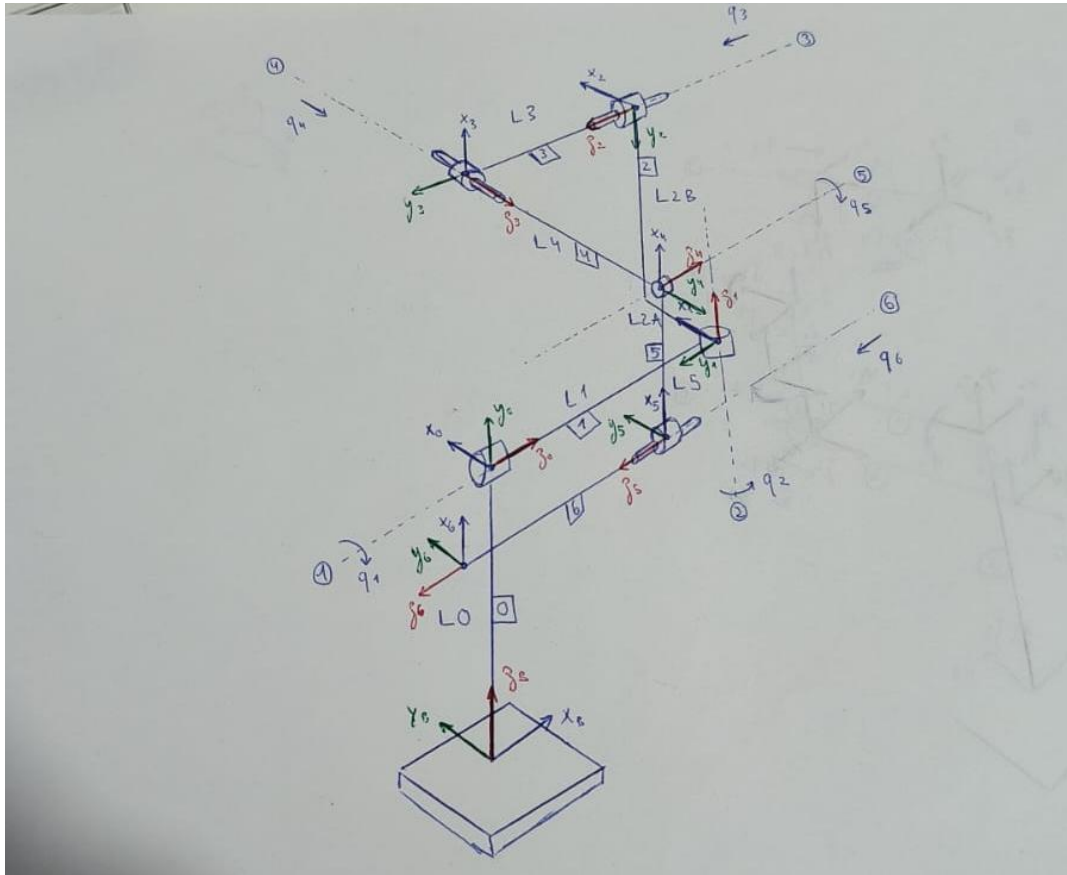
El robot de 6 grados de libertad con el que se va a trabajar es el aquí expuesto:



2.1. Algoritmo Denavit-Hartenberg y tabla D-H

Se plantea un método basado en cambios de base mediante MTH, con el que se sistematiza la selección de los sistemas de coordenadas a partir de movimientos simples como rotaciones o traslaciones entorno a ejes concretos.

Siguiendo el procedimiento de Denavit-Hartenberg, se obtiene:



Se puede finalmente completar la tabla D-H, y se obtiene:

```
%      theta_i  d_i   a_i   alpha_i
TDH = [ PI/2,    L0,    0,    PI/2           ;... % Link i=0
        q1,      L1,    0,    PI/2           ;... % Link i=1
        q2,      L2B,   L2A, -PI/2          ;... % Link i=2
        PI/2,    L3+q3, 0,    PI/2           ;... % Link i=3
        0,       L4+q4, 0,    PI/2           ;... % Link i=4
        q5,      0,     -L5,  PI            ;... % Link i=5
        0,       L6+q6, 0,    0              ]; % Link i=6

end|
```

2.2.-Matrices de transformación Homogéneas (MTH)

Una vez obtenido los parámetros de la tabla D-H, se procede a encontrar las MTH que definan el cambio de base entre sistemas.

Se busca encontrar las matrices que permitan representar la orientación y posición de los eslabones del brazo robótico.

A partir de los parámetros de la tabla D-H, se pueden obtener las matrices de transformación las cuales representan la transformación del sistema S_{i-1} al sistema S_i

Estas matrices se calculan como:

$${}^{i-1}\mathbf{A}_i = \mathbf{Rotz}(\theta_i) \mathbf{T}(0,0,d_i) \mathbf{T}(a_i,0,0) \mathbf{Rotx}(\alpha_i)$$

Siendo θ_i una rotación alrededor del eje z_{i-1} ; la distancia d_i una traslación a lo largo de z_i ; la distancia a_i una traslación a lo largo de x_i ; y el ángulo α una rotación alrededor del eje x_i .

TB0 =

```
[0, 0, 1, 0]
[1, 0, 0, 0]
[0, 1, 0, L0]
[0, 0, 0, 1]
```

T01 =

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

T12 =

```
[cos(q2), 0, -sin(q2), L2A*cos(q2)]
[sin(q2), 0, cos(q2), L2A*sin(q2)]
[0, -1, 0, L2B]
[0, 0, 0, 1]
```

T23 =

```
[0, 0, -1, 0]
[-1, 0, 0, 0]
[0, 1, 0, L3 + q3]
[0, 0, 0, 1]
```

T34 =

```
[1, 0, 0, 0]
[0, 0, -1, 0]
[0, 1, 0, L4 + q4]
[0, 0, 0, 1]
```

T45 =

```
[cos(q5), sin(q5), 0, -L5*cos(q5)]
[sin(q5), -cos(q5), 0, -L5*sin(q5)]
[0, 0, -1, 0]
[0, 0, 0, 1]
```

T56 =

```
[1, 0, 0, 0]
[0, 1, 0, 0]
[0, 0, 1, L6 + q6]
[0, 0, 0, 1]
```

Para verificar si la configuración cinemática es correcta, se remplace todas las variables articulares por 0 para poner el brazo robótico en su posición de origen (HOME). Luego, en la matriz de tarea se puede observar en el vector posición que las coordenadas coinciden con las del extremo del robot.

Para calcular la matriz tarea: $T_n = {}^0A_1 {}^1A_2 \dots {}^{n-1}A_n$

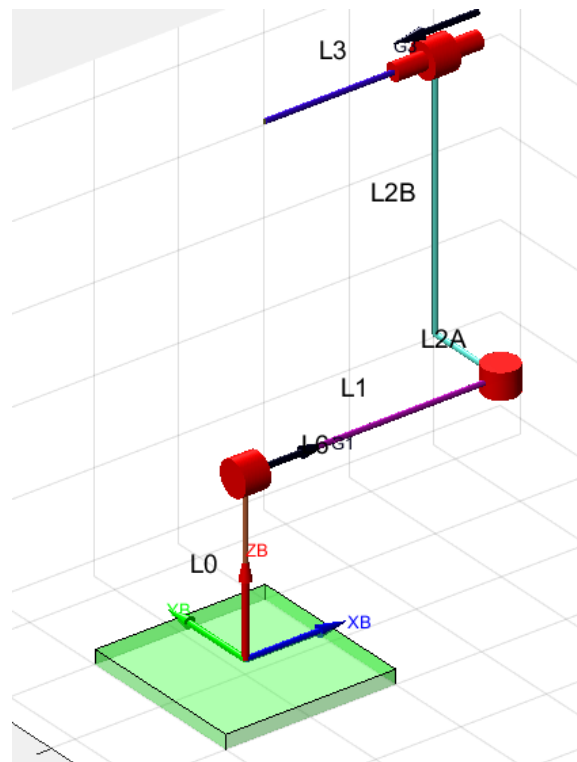
Se obtiene:

$$TB6 = \begin{bmatrix} 0 & 0 & -1 & L1 - L3 - L6 \\ 0 & 1 & 0 & L2A - L4 \\ 1 & 0 & 0 & L0 - L5 + L2B \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

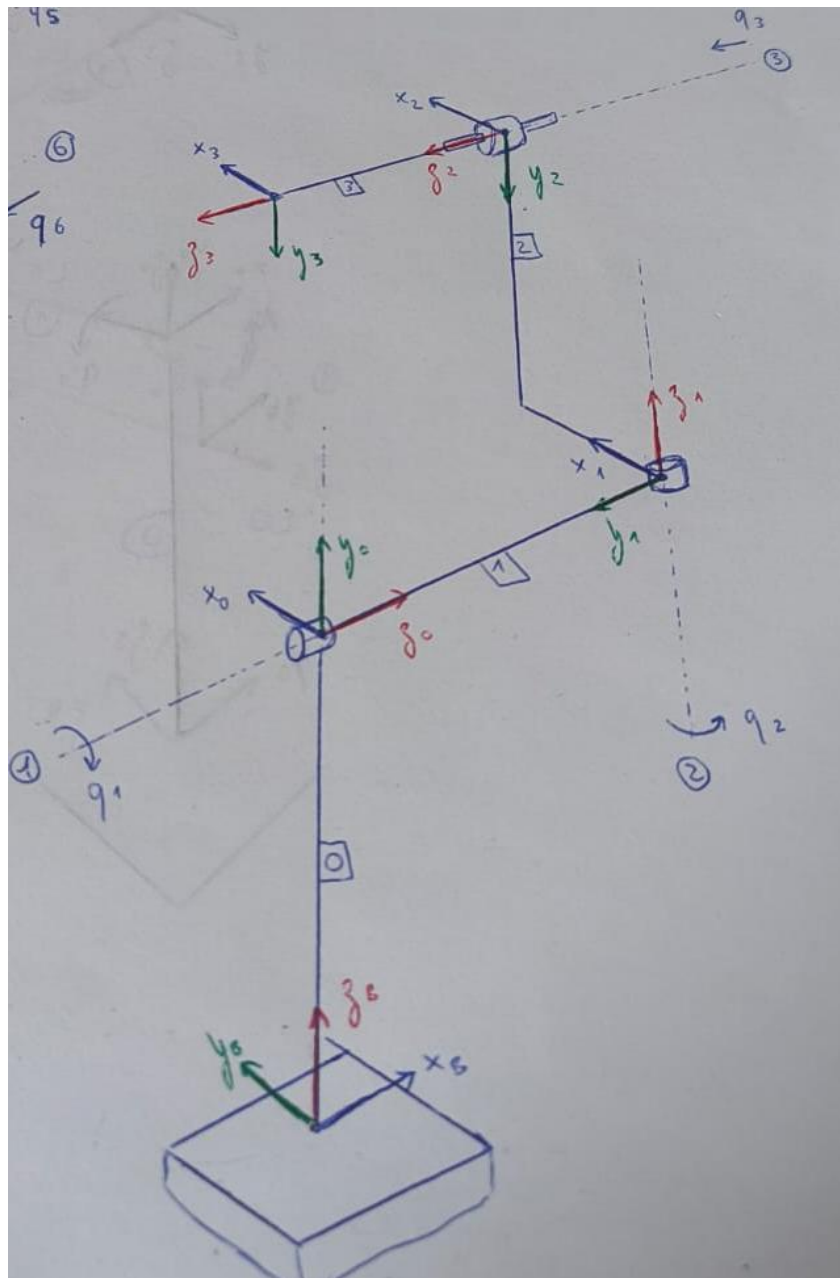
Donde en la sub matriz de traslación se puede verificar la posición del extremo del robot referido a la base.

2.3 - Ecuaciones del modelo cinemático directo

Para este apartado se trabaja con el robot reducido considerando solo los 3 primeros grados de libertad.



Se obtiene por tanto este brazo robótico, teniendo en cuenta que hay que cambiar el marco de referencia del extremo del brazo:



En cuanto a las expresiones simbólicas de posición(x,y,z) se obtienen estas ecuaciones las cuales representan la posición del nuevo extremo del brazo robótico en función de la base de referencia:

```
% Posición del extremo de la articulación 3
x = L1 - cos(q2)*(L3 + q3) - L2A*sin(q2);
y = L2A*cos(q1)*cos(q2) - cos(q1)*sin(q2)*(L3 + q3) - L2B*sin(q1);
z = L0 + L2B*cos(q1) - sin(q1)*sin(q2)*(L3 + q3) + L2A*cos(q2)*sin(q1);
```

En cuanto a las expresiones numéricas de posición y orientación lo que se hace es sacar los ángulos de Euler mediante la combinación ZXZ.

Los ángulos de Euler son 3 ángulos (phi,theta,psi) que describen la orientación de un cuerpo rígido en el espacio tridimensional con respecto a un sistema de referencia. En cuanto al convenio ZXZ, esta combinación es una forma de representar la orientación del sólido rígido mediante 3 rotaciones, obteniendo una matriz de rotación:

$$\mathbf{R} = \mathbf{R}_{z,\phi} \mathbf{R}_{u',\theta} \mathbf{R}_{w'',\psi} = \begin{bmatrix} C\phi & -S\phi & 0 \\ S\phi & C\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} C\phi C\psi - S\phi C\theta S\psi & -C\phi S\psi - S\phi C\theta C\psi & S\phi S\theta \\ S\phi C\psi + C\phi C\theta S\psi & -S\phi S\psi + C\phi C\theta C\psi & -C\phi S\theta \\ S\theta S\psi & S\theta C\psi & C\theta \end{bmatrix}$$

Una vez sabido la posición y orientación del extremo del robot, se representa la matriz orientación en términos de ángulos de Euler y se compara con la matriz de orientación del robot

A partir de este momento, podemos despejar los ángulos por comparación:

-Para sacar θ : $ZXZ_{31}/ZXZ_{33}=TB_{331}/TB_{333}$

-Para sacar ψ : $ZXZ_{32}/ZXZ_{31}=TB_{331}/TB_{333}$

-Para sacar ϕ : $ZXZ_{31}/ZXZ_{23}=TB_{331}/TB_{323}$

Obteniendo:

```
phi=atan2(-cos(q2),cos(q1)*sin(q2));  
psi=atan2(cos(q2)*sin(q1),-cos(q1));  
theta=atan2(cos(q2),-sin(psi)*sin(q2));
```

Para verificar si los ángulos de Euler han sido bien calculados, verificamos en la posición HOME si el vector orientación es correcto:

```
>> [pos,orient]=CinematicaDirecta([0,0,0])  
  
pos =  
  
    0.5000  
    0.5000  
    2.5000  
  
orient =  
  
   -1.5708  
    1.5708  
    3.1416
```

Se observa que en la posición HOME; $\phi = -\pi/2$, $\theta = \pi/2$ y $\psi = \pi$; lo cual coincide con la orientación del extremo del robot respecto a la base.

2.4 – Cinemática Inversa

Una vez obtenida la cinemática directa, se plantea el problema inverso. El objetivo de este apartado es el de encontrar los valores que deben adoptar las coordenadas articulares del robot (q_1, q_2 y q_3) para que el extremo del robot se posicione y oriente según una determinada localización espacial [noap].

Para proceder con el despeje de las variables articulares, primero se crea una matriz simbólica que representa la matriz de transformación del robot:

$T_d =$

$$\begin{bmatrix} nx & ox & ax & x \\ ny & oy & ay & y \\ nz & oz & az & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad AB0 \cdot A01 \cdot A12 \cdot A23 = T_d$$

El contenido de la matriz son datos ya sabidos, lo único que queda es ir creando sistemas de ecuaciones en los que sea posible despejar las variables articulares en función de x, y, z .

En primer lugar, se obtiene el MCD de la siguiente manera:

$$A12 \cdot A23 = A01^{-1} \cdot AB0^{-1} \cdot T_d$$

Por tanto, se obtiene un sistema de ecuaciones que queda:

ans =

$$L2A \cdot \cos(q2) - \sin(q2) \cdot (L3 + q3) == y \cdot \cos(q1) - L0 \cdot \sin(q1) + z \cdot \sin(q1)$$

ans =

$$\cos(q2) \cdot (L3 + q3) + L2A \cdot \sin(q2) == L1 - x$$

ans =

$$L2B == z \cdot \cos(q1) - L0 \cdot \cos(q1) - y \cdot \sin(q1)$$

El siguiente paso sería aislar una de las variables articulares, en este caso se elige aislar q_1 mediante cambio de coordenadas cartesianas a polares.

Para realizar el cambio se utiliza:

$$A \cdot \text{sen}(q_1) + B \cdot \text{cos}(q_1) = C$$

Teniendo en cuenta que en polares:

$$A = R \cdot \text{cos}(\alpha) \qquad B = R \cdot \text{sen}(\alpha)$$

Siendo: $R = \sqrt{A^2 + B^2} \qquad \alpha = \tan\left(\frac{A}{B}\right)$

Ahora la ecuación quedaría como:

$$R \cdot \text{cos}(\alpha) \cdot \text{sen}(q_1) + R \cdot \text{sen}(\alpha) \cdot \text{cos}(q_1) = C$$

Se puede apreciar que la ecuación resultante se puede expresar como el seno de la suma, quedando:

$$R \cdot \text{sen}(\alpha + q_1) = C$$

De aquí ya se puede despejar el seno y por tanto el coseno:

$$\text{sen}(\alpha + q_1) = \frac{C}{R} \qquad \text{cos}(\alpha + q_1) = \sqrt{1 - \text{sen}(\alpha + q_1)^2}$$

A nivel matemático, el seno tiene que estar siempre entre -1 y 1, por tanto, el seno recién calculado tiene que cumplir la condición donde C no pueda ser superior a R.

Esto se debe ya que en caso de que $C > R$ no existiría solución posible y se debería devolver que esa variable articular vale 0 (Fuera de rango).

Esta condición se cumple gracias a este bucle "if" :

```
if abs(C) > abs(R)
    q = [0;0;0];
    return;
end
```

Mediante el comando de Matlab "atan2(sen,cos)", se obtiene el valor de q_1 ; teniendo en cuenta que hay que restarle "alpha"; queda finalmente:

$$q_{1a} = \text{atan2}(\text{sen}(\alpha + q_1), \text{cos}(\alpha + q_1)) - \alpha$$

$$q_{1b} = \text{atan2}(\text{sen}(\alpha + q_1), -\text{cos}(\alpha + q_1)) - \alpha$$

Finalmente, hay que tener en cuenta que a la hora de calcular el coseno se realiza una raíz cuadrada, por tanto, se obtienen 2 soluciones de q_1 .

Continuando con el proceso, se decide aislar posteriormente q_3 .

En este caso en vez de utilizar la conversión a polares, se va a solucionar por comparación.

En primer lugar; del MCD calculado previamente; se pueden observar varias relaciones que pueden simplificar los cálculos para aislar q_3 . En este caso, se decide elevar al cuadrado tanto la primera como la segunda fila de nuestro sistema de ecuaciones y sumarlas posteriormente para quitarnos de encima la variable articular q_2 y obtener una ecuación bastante más reducida donde es significativamente más sencillo aislar q_3 :

$$(y \cdot \cos(q_1) - L_0 \cdot \sin(q_1) + z \cdot \sin(q_1))^2 + (L_1 - x)^2 = L_3^2 + 2 \cdot L_3 \cdot q_3 + L_2^2 + q_3^2$$

Se puede observar que la ecuación solo depende de q_3 ya que la única otra variable que aparece es q_1 que ha sido anteriormente calculada.

Además, la ecuación es una cuadrática, solo basta con resolver sus raíces para obtener las 2 soluciones de q_3 , teniendo en cuenta que han sido calculadas otras 2 soluciones de q_1 , lo que hace un total de 4 soluciones de q_3 (2 por cada raíz).

A la hora de resolver la cuadrática, para simplificar el cálculo se dan nombre a las diferentes partes de la ecuación:

```
N=((y*cos(q1b) - L0*sin(q1b) + z*sin(q1b))^2 + (L1 - x)^2 - L3^2 - L2^2);

%a*q3^2 + b*q3 + c = 0
a=1;
b=2*L3;
c=-N;
```

Por tanto, se obtienen las siguientes soluciones de q_3 :

$$q_{3a} = \frac{(-b - \sqrt{b^2 - 4 \cdot a \cdot c})}{(2 \cdot a)} \quad q_{3b} = \frac{(-b + \sqrt{b^2 - 4 \cdot a \cdot c})}{(2 \cdot a)}$$

Por último, para calcular la variable articular q_2 , se hará el mismo procedimiento que para q_1 , es decir, pasar de coordenadas cartesianas a polares.

La ecuación con la que se trabaja es:

$$\cos(q_2) \cdot (L_3 + q_3) + L_2A \cdot \sin(q_2) = L_1 - x$$

Donde se puede observar que solo depende de q_2 ya que q_3 es dato.

Se obtienen, por tanto 2 soluciones de q_2 :

$$q_{2a} = \text{atan2}(\sin(\alpha_2 + q_2), \cos(\alpha_2 + q_2)) - \alpha_2$$

$$q_{2b} = \text{atan2}(\sin(\alpha_2 + q_2), -\cos(\alpha_2 + q_2)) - \alpha_2$$

Siendo “ α_2 ” el nuevo ángulo “Alpha” calculado, pero con los datos de la ecuación anterior.

Sin embargo, hay que tener en cuenta que q_3 tenía 4 soluciones, y que, por tanto, q_2 ; al contar con 2 soluciones por la raíz cuadrada; tendrá un total de 8 soluciones posibles

Estas 8 soluciones se recogen en una matriz para verificar cual de las 8 combinaciones son las conveniente a elegir y poder terminar con la cinemática Inversa del problema.

Para verificar que combinaciones son las correctas, basta con enviar a la función “CinemáticaInversa(xyz)” la posición HOME del robot y observar el valor de las variables articulares. Según el enunciado, en la posición HOME las variables articulares están a 0.

Al llamar a la función introduciendo el vector HOME=[0.5,0.5,2.5] se recibe el vector q_{aux} con todas las combinaciones posibles, apreciando que hay solo una solución correcta, caso en el que las variables articulares están todas a 0.

Sin embargo, todas las demás soluciones son igual de válidas solo que no corresponden con el enunciado del ejercicio que quiere que las variables articulares en el origen sean nulas. Es decir que si le enviamos cualquier otro punto que no tenga ninguna condición especial debería devolver todas las soluciones posibles

Para hacer la distinción entre soluciones se implementa una condición que elimina toda combinación que tenga la variable q_3 fuera de su rango, ya que q_3 solo puede moverse entre 0 y 1.5.

2.5 – Control Cinemático

El objetivo de esta parte es establecer cuales son las trayectorias que debe seguir cada articulación del robot a lo largo del tiempo para conseguir los objetivos fijados por el usuario, en este caso se quiere dibujar una circunferencia en el plano XY (teniendo en cuenta las restricciones físicas).

Para este apartado se dibuja la circunferencia con centro en (1, 0.5, 3) y un radio de $R=1$, teniendo en cuenta que q_3 esta reducido a $[0, 1.5]$ y que q_1 y q_2 no pueden girar más de $[-\pi, \pi]$.

Se decide hacer la trayectoria con 72 puntos (de 1 a 73). Para recorrer los 72 puntos del círculo, se decide crear un vector que genera 72 puntos equiespaciados en una circunferencia que gira de 0 a 2π :

```
theta=linspace(0,2*pi,73)
```

La trayectoria se genera a partir de un bucle “for” que irá recogiendo tanto las coordenadas de los puntos como los valores de las variables articulares en cada punto de los 72:

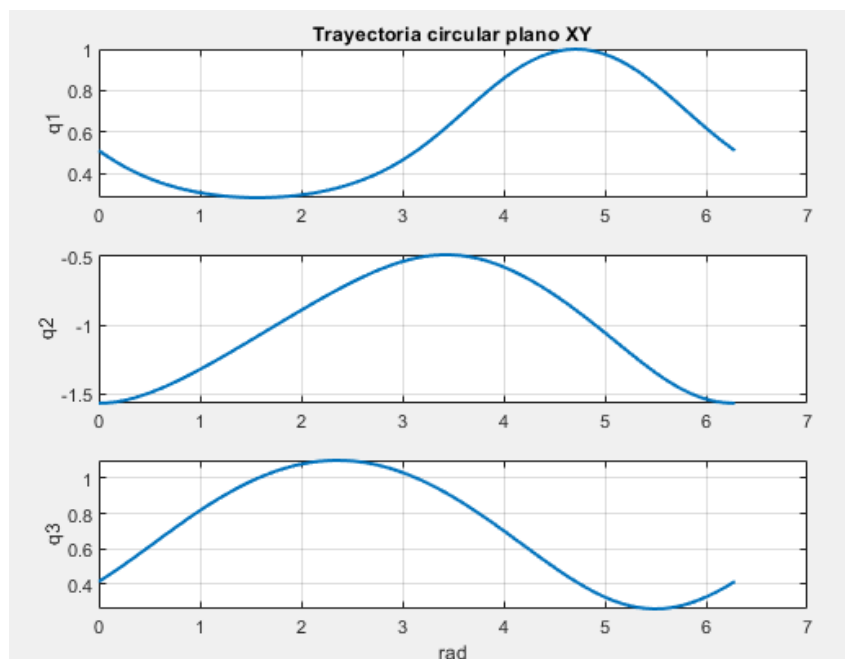
```
for i= 1:73
    x(i)=xc+R*cos(theta(i));           %x=xc+r*cos(theta)
    y(i)=yc+R*sin(theta(i));           %y=yc+r*sen(theta)
    z(i)=zc;                           %constante-->plano XY

    ver=CinematicaInversa([x(i),y(i),z(i)]);
    q1(i)=ver(1,2);
    q2(i)=ver(2,2);
    q3(i)=ver(3,2);
```

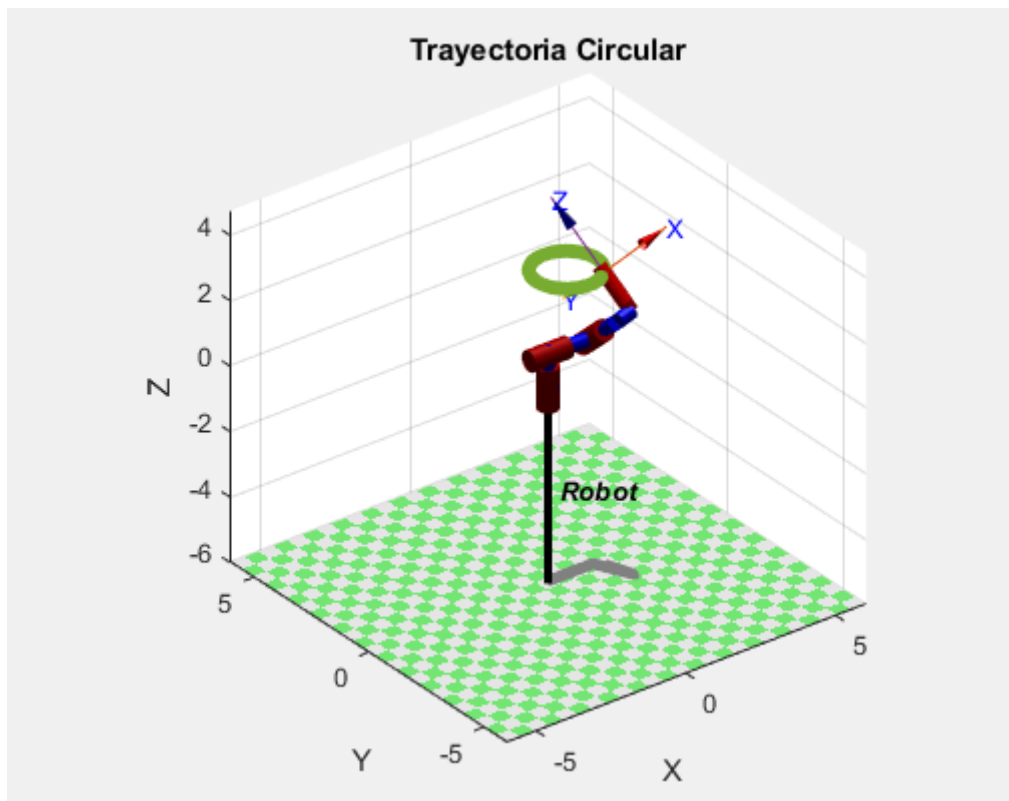
end

Se expresa la circunferencia en el plano XY usando su forma paramétrica, con centro (xc,yc,zc) y con ángulo parametrizado de forma que cada incremento de i suma 5° para que $i=73$ se comporte como una vuelta completa.

Además, se hace llamada a la función “CinematicaInversa(xyz)” para ir recogiendo los valores de las variables articulares de cada punto y poder dibujar su trayectoria:



A esta gráfica se le añade una animación que ayuda a visualizar mejor la trayectoria del robot la cual se encuentra en el archivo "PROYECTO" en el final de la parte 6:



2.6 - Jacobianos

Los Jacobianos establecen la relación entre las velocidades (o incrementos) de $(q_1, q_2 \text{ y } q_3)$ y $(x, y, z, \phi, \theta, \psi)$.

Si el Jacobiano es singular ($\det(J) = 0$), el robot está en una configuración singular, lo que puede causar pérdida de movilidad.

En primer lugar, se estudia el Jacobiano directo de modo que se puedan relacionar las velocidades articulares con las velocidades del extremo del robot, quedando:

Jdir =

$$\begin{bmatrix} 0, & \sin(q_2)*(L_3 + q_3) - L_2A*\cos(q_2), & -\cos(q_2) \\ \sin(q_1)*\sin(q_2)*(L_3 + q_3) - L_2B*\cos(q_1) - L_2A*\cos(q_2)*\sin(q_1), & -\cos(q_1)*\cos(q_2)*(L_3 + q_3) - L_2A*\cos(q_1)*\sin(q_2), & -\cos(q_1)*\sin(q_2) \\ L_2A*\cos(q_1)*\cos(q_2) - \cos(q_1)*\sin(q_2)*(L_3 + q_3) - L_2B*\sin(q_1), & -\cos(q_2)*\sin(q_1)*(L_3 + q_3) - L_2A*\sin(q_1)*\sin(q_2), & -\sin(q_1)*\sin(q_2) \end{bmatrix}$$

En segundo lugar, se estudia el Jacobiano inverso lo que permite encontrar las velocidades articulares necesarias para alcanzar una velocidad deseada en el extremo. Calcularlo con Matlab se reduce a hacer la inversa del Jacobiano directo:

$$J_{inv} = \text{inv}(J_{dir})$$

Para extraer los puntos singulares del manipulador donde pueden ocurrir singularidades, igualamos la matriz del jacobiano a cero y se despejan las expresiones de los parámetros articulares:

DetJ=simplify(det(Jdir))

$$(L_3 + q_3)*(L_3*\sin(q_2) - L_2A*\cos(q_2) + q_3*\sin(q_2)) == 0$$

Se pueden obtener por tanto 2 soluciones posibles:

- $L_3 + q_3 == 0$ No podría ser posible ya que para que esta igualdad se cumpla q_3 debería ser negativa, y se ha impuesto que q_3 se mueve de 0 a 1.5.

- $L_3*\sin(q_2) - L_2A*\cos(q_2) + q_3*\sin(q_2) == 0$ Despejando la ecuación se obtiene:

$$\tan(q_2) == L_2A / (L_3 + q_3)$$

Se observa que la tangente de q_2 depende de q_3 , por tanto, los puntos singulares del extremo del robot serán todos los puntos que cumplan tal igualdad.

Es decir que tenemos un rango de singularidad de q_2 : de forma simbólica $[L2A/L3, L2A/L3+1.5]$ o de forma numérica $[0.2, 1]$. Es decir, para cualquier valor de q_3 que q_2 esté dentro de ese rango, el extremo del robot perderá movilidad

2.7 – Información complementaria

Una vez verificado todos los resultados, al llamar a la función “verificador.p” se obtiene:

```
Comprobando ficheros de verificación
-----
- Fichero tablaDH.m -----> OK
- Fichero matricesTH.m -----> OK
- Fichero CinematicaDirectaSimbolica.m --> OK
- Fichero CinematicaDirecta.m -----> OK
- Fichero CinematicaInversaSimbolica.m --> OK
- Fichero CinematicaInversa.m -----> OK
- Fichero trayectoriaCircular.m -----> OK
- Fichero jacobiano.m -----> OK
- Fichero parametrosDinamicos.m -----> OK (Atención: No ha modificado el fichero original).
- Fichero ModeloDinamico_R3GDL.m -----> OK (Atención: No ha modificado el fichero original).
-----
Todos los ficheros de verificación son sintácticamente correctos
```

Por tanto, todos los archivos de verificación funcionan correctamente.

Además, en la carpeta de verificación se adjunta tanto un archivo MATLAB llamado "PROYECTO" como las diferentes funciones de rotación en cada eje, traslación y cálculo de matrices de transformación homogénea.

En cuanto el contenido del archivo "PROYECTO", dentro se han realizado todos los cálculos de los diferentes apartados del trabajo, con pequeñas descripciones sobre cada calculo o ecuación para poder entender el procedimiento de cada apartado.

En dichos apartados se realiza el procedimiento más amplio de cada parte para poder facilitar la comprensión del lector y además añadir información extra que puede ayudar a visualizar mejor cada problema.

3 – Análisis Dinámico

El objetivo de esta parte es conocer la relación entre el movimiento del robot y las fuerzas implicadas en el mismo. Se establecerán diversas relaciones matemáticas entre la localización del robot con su velocidad y aceleración; las fuerzas y pares aplicados en las articulaciones...

Estas relaciones se crean a partir de un modelo dinámico que permite simular el movimiento del robot, diseñar la estructura mecánica, dimensionar actuadores, diseñar el control dinámico del robot...

Para crear dicho modelo se pueden usar diversas formulaciones, en este caso se usará la formulación de Newton-Euler.

En primer lugar, se sacarán las ecuaciones de la dinámica inversa.

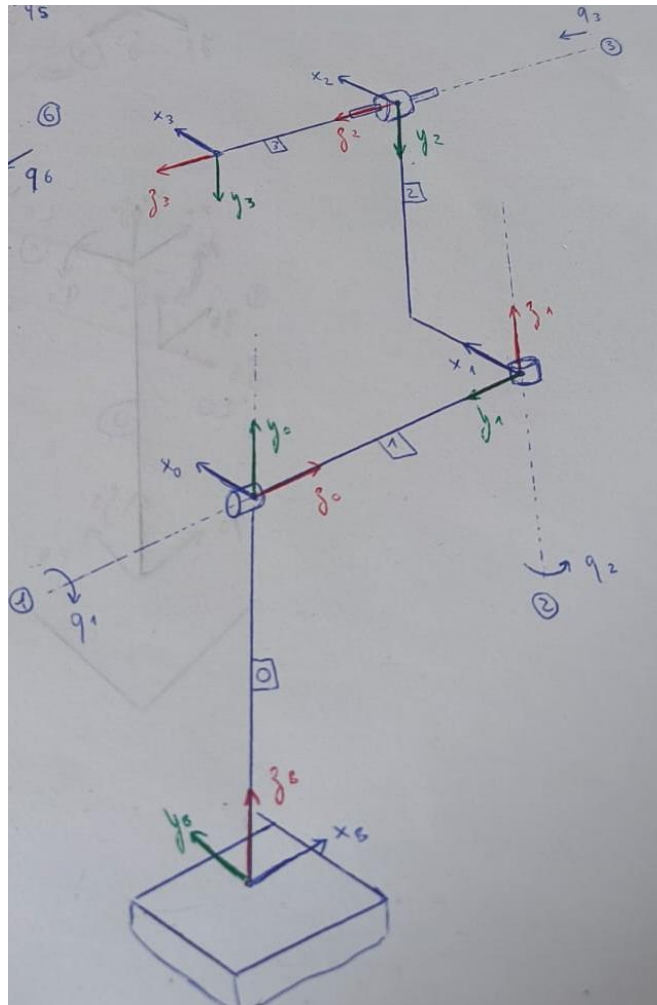
En segundo lugar, a partir de dichas ecuaciones se creará una simulación de la dinámica del robot.

Por último, se comprobará dicho simulador si corresponde con el modelo que proporciona el enunciado para verificar el buen funcionamiento del modelo creado.

Cabe a destacar que todos los cálculos serán desarrollados en un archivo Matlab llamado "PROYECTO_B.m" con el fin de mostrar con más precisión cada cálculo realizado.

3.1 – Dinámica Inversa

Para el cálculo de la dinámica inversa se considera de nuevo solo los 3 primeros grados de libertad del robot (ya que para robots con >3 gdl la deducción analítica se hace excesivamente compleja):



El Modelo de la dinámica inversa representa la evolución temporal de las fuerzas y pares del robot en función de las coordenadas articulares:

$$\tau(t) = f(q(t))$$

Se busca expresar la ecuación dinámica de un robot de 3 grados de libertad mediante la formulación de Newton-Euler:

$$\tau = \mathbf{D}(q)\ddot{q} + \mathbf{H}(q, \dot{q}) + \mathbf{C}(q)$$

Con:

q : vector de coordenadas articulares.

τ : vector de fuerzas o pares que se aplica a cada articulación.

$D(q)$: la matriz de inercias, de dimensión (nxn), cuyos elementos son función de q .

$H(q, \dot{q})$: matriz (nx1) de fuerzas de Coriolis, dependiente de q y \dot{q} .

$C(q)$: matriz (nx1) de fuerzas de gravedad, dependiente de q .

n : número de grados de libertad del robot.

Se proporcionan estos parámetros dinámicos:

	Articulación 0	Articulación 1	Articulación 2	Articulación 3
Masa (Kg)	$m_0 = 65.5965$	$m_1 = 98.3947$	$m_2 = 131.193$	$m_3 = 65.5965$
Longitud (m)	$L_0 = 1$	$L_1 = 1.5$	$L_{2A} = 0.5, L_{2B} = 1.5$	$L_3 = 1$
Inercias motores (kg m ²)	-	$J_1 = 0.00394351$	$J_2 = 0.0029685$	$J_3 = 0.00348257$
Fricciones motores(Nm/(rad/s))	-	$B_1 = 0.000101598$	$B_2 = 0.00022592$	$B_3 = 0.00015022$
Reductor	-	$R_1 = 20$	$R_2 = 25$	$R_3 = 35$

Los eslabones son todos cilíndricos huecos con: $r_{int} = 0.08$ m y $r_{ext} = 0.1$ m.

Y tienen una densidad constante $\rho = 5800 \text{ kg/m}^3$

Se puede calcular tanto el área como la densidad de cada barra:

$$A = \pi * (r_{ext}^2 - r_{int}^2) \quad \rho_L = \rho * A$$

Para comenzar con la formulación del método de Newton-Euler (método que se basa en la 2da ley de Newton) se asigna a cada eslabón un sistema de referencia de acuerdo con las normas de D-H además de las diferentes propiedades geométricas de cada eslabón.



Para continuar con el proceso se necesita el cálculo del tensor de inercia y coordenadas del centro de masa de cada eslabón.

En cuanto al tensor de inercia solo interesa los elementos diagonales de la matriz los cuales son llamados los momentos de inercia:

$$I = \sum_i m_i R_i^2$$

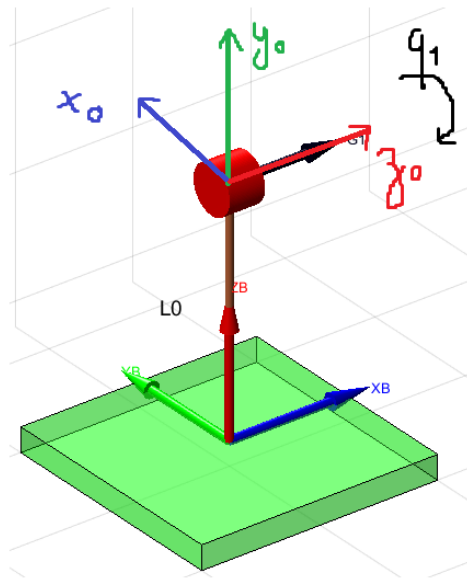
Donde R_i es la distancia de una partícula m_i al eje.

Cabe destacar que hay que hacer la diferencia entre un eslabón con 2 masas y un eslabón con 1 masa. Además, dependiendo de la dirección del eje que se estudia se obtendrá un momento de inercia u otro:

Eje	Momento de inercia
 El del cilindro	$\frac{1}{2}M (R_1^2 + R_2^2)$
 Perpendicular por el centro	$\frac{1}{12}MH^2$

En cuanto a las coordenadas del centro de masas, hay que tener en cuenta que todas las masas de las barras son homogéneas.

Eslabón 0:

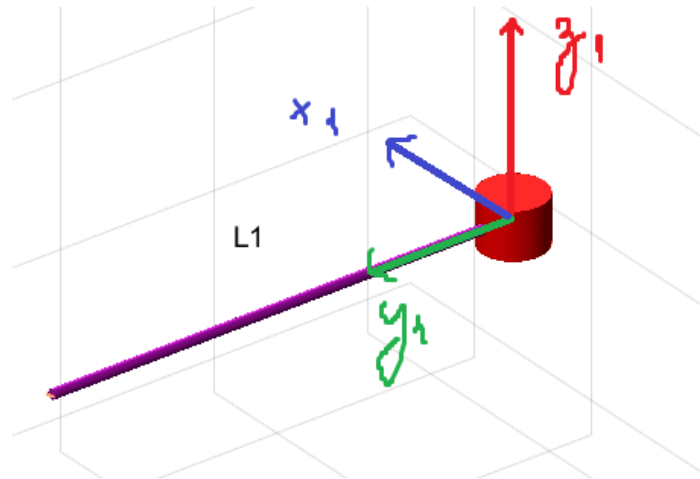


En este caso el eslabón tiene una única masa, por tanto, el centro de gravedad se encuentra en el centro de la barra:

$$s_{00} = [0, -L_0/2, 0]^T \quad m_0 = 65.5965; \text{ \% kg}$$

En cuanto al tensor de inercias, el eslabón es fijo, por tanto, no hace falta especificarlo porque no se mueve respecto al sistema Base B.

Eslabón 1:



En este caso el eslabón tiene una única masa, por tanto, el centro de gravedad se encuentra en el centro de la barra:

$$s_{11} = [0, L1/2, 0]' \quad m1 = 98.3947; \% \text{ kg}$$

En cuanto al tensor de inercias, podemos calcular los momentos de inercia teniendo en cuenta que el eje 'y' está en dirección al eje de la barra:

$$I_{xx} = \frac{1}{12} \cdot m1 \cdot L1^2$$

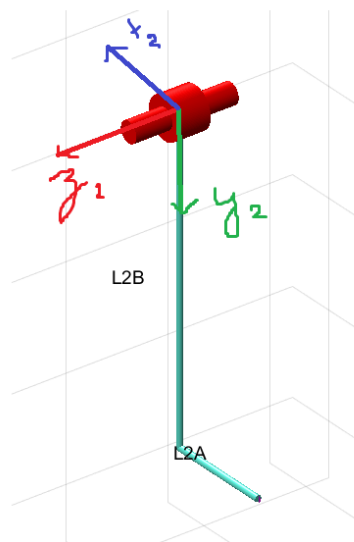
$$I_{yy} = \frac{1}{2} \cdot m1 \cdot (r_{ext}^2 + r_{int}^2)$$

$$I_{zz} = \frac{1}{12} \cdot m1 \cdot L1^2$$

Siendo el tensor de inercias:

$$I_{11} = \begin{pmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{pmatrix}$$

Eslabón 2:



En este caso el eslabón tiene 2 masas, por tanto, hay que calcular el centro de masas, así como las masas de ambas barras:

$$\begin{aligned} sA2 &= [-L2A/2, L2B, 0]' \\ sB2 &= [0, L2B/2, 0]' \end{aligned}$$

$$\begin{aligned} m2 &= 131.193; \% \text{ kg} \\ mA &= (m2 * L2A) / (L2A + L2B) \\ mB &= (m2 * L2B) / (L2A + L2B) \end{aligned}$$

Calculando las masas de manera proporcional según la longitud de cada barra.

Una vez calculado el centro de masas de cada barra podemos calcular el centro de masas del eslabón completo, así como la posición de cada barra respecto al centro de masas del eslabón:

$$s_{22} = \frac{(m_A \cdot s_{A2} + m_B \cdot s_{B2})}{(m_A + m_B)} \quad \begin{array}{l} r_{2A} = s_{22} - s_{A2} \\ r_{2B} = s_{22} - s_{B2} \end{array}$$

Para hallar el tensor de inercias del eslabón se calcula en un primer momento el tensor de inercias de cada barra:

- Tensor de Inercias barra A:

$$\begin{aligned} I_{2Axx} &= \frac{1}{2} \cdot m_A \cdot (r_{ext}^2 + r_{int}^2) \\ I_{2Ayy} &= \frac{1}{12} \cdot m_B \cdot L_{2A}^2 & I_{2Azz} &= \frac{1}{12} \cdot m_B \cdot L_{2A}^2 \\ I_{2A} &= \begin{pmatrix} I_{2Axx} & 0 & 0 \\ 0 & I_{2Ayy} & 0 \\ 0 & 0 & I_{2Azz} \end{pmatrix} \end{aligned}$$

- Tensor de Inercias barra B:

$$\begin{aligned} I_{2Bxx} &= \frac{1}{12} \cdot m_B \cdot L_{2B}^2 \\ I_{2Byy} &= \frac{1}{2} \cdot m_B \cdot (r_{ext}^2 + r_{int}^2) & I_{2Bzz} &= \frac{1}{12} \cdot m_B \cdot L_{2B}^2 \\ I_{2B} &= \begin{pmatrix} I_{2Bxx} & 0 & 0 \\ 0 & I_{2Byy} & 0 \\ 0 & 0 & I_{2Bzz} \end{pmatrix} \end{aligned}$$

En nuestro caso el momento de inercia del eslabón es respecto a un eje que no pasa por su centro de masa, pero es paralelo a uno que sí lo hace. Por tanto, se puede calcular la aportación que hace cada barra al tensor de inercias del cuerpo gracias al teorema de Steiner:

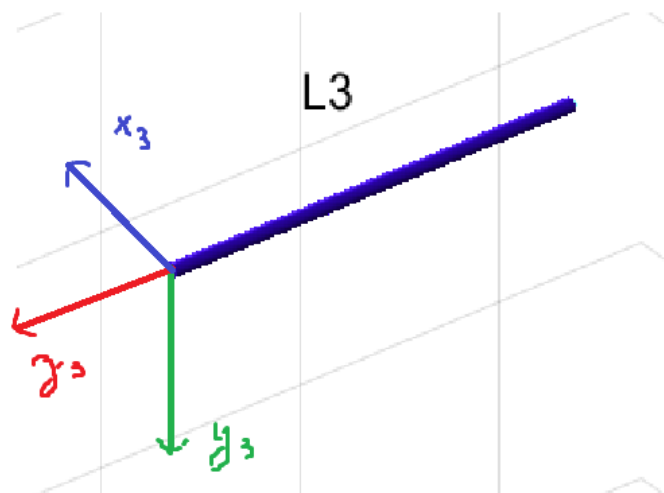
$$Steiner_i = norm(r_i)^2 \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - r_i \cdot r_i^T$$

- Barra A: $I2A_{cdm} = I2A + m_A * Steiner2A$
- Barra B: $I2B_{cdm} = I2B + m_B * Steiner2B$

El tensor de inercias del cuerpo queda como la suma de la aportación de cada barra:

$$I22 = I2A_{cdm} + I2B_{cdm}$$

Eslabón 3:



En este caso el eslabón tiene una única masa, por tanto, el centro de gravedad se encuentra en el centro de la barra:

$$s_{33} = [0, 0, -L_3/2]'; \quad m_3 = 65.5965 \text{ ; \% kg}$$

En cuanto al tensor de inercias, podemos calcular los momentos de inercia teniendo en cuenta que el eje 'z' está en dirección al eje de la barra:

$$I_{3xx} = \frac{1}{12} \cdot m_3 \cdot L_3^2$$

$$I_{3yy} = \frac{1}{12} \cdot m_3 \cdot L_3^2$$

$$I_{3zz} = \frac{1}{2} \cdot m_3 \cdot (r_{ext}^2 + r_{int}^2)$$

Siendo el tensor de inercias:

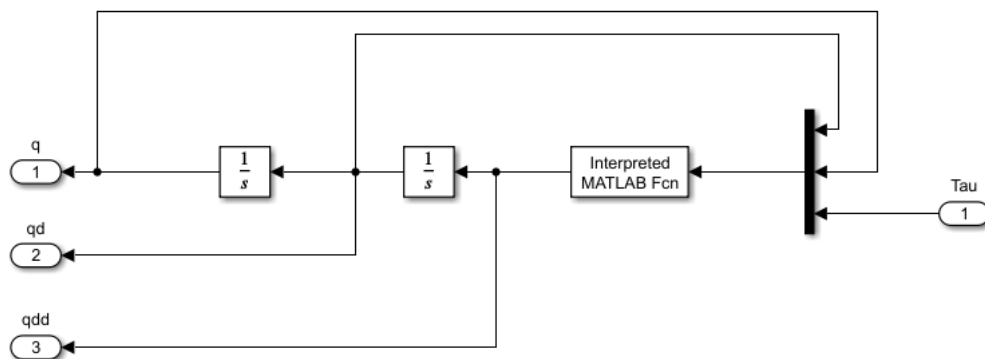
$$I_3 = \begin{pmatrix} I_{3xx} & 0 & 0 \\ 0 & I_{3yy} & 0 \\ 0 & 0 & I_{3zz} \end{pmatrix}$$

3.2 - Simulador de la dinámica y del modelo de referencia

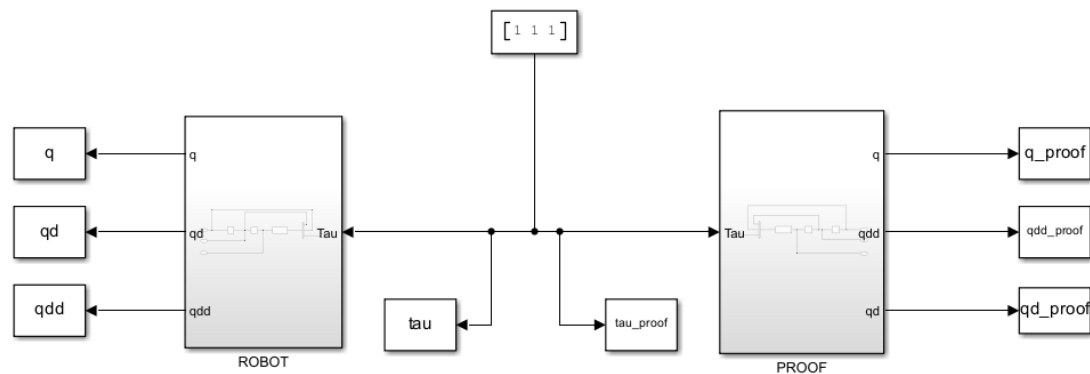
Para esta parte se pide crear un simulador ("Simulador_Dinamica.slx") con el fin de poder comprobar el funcionamiento de la dinámica del robot recién calculada. Para ello se proporciona un modelo dinámico de referencia para poder hacer las apropiadas comparaciones y verificar el correcto funcionamiento de los cálculos.

Por tanto, se decide crear un simulador el cual implementa diferentes funciones. Por un lado, se tiene la dinámica calculada previamente (matriz de inercias, matriz de Coriolis y matriz de fuerzas de gravedad). Por otro lado, se tiene el modelo proporcionado por el profesor.

En ambos modelos se obtienen las diferentes velocidades, aceleraciones y posición de las variables articulares. Lo único que queda por añadir es un torque genérico a cada modelo para poder obtener estas variables articulares y graficarlas con el fin de compararlas y verificar su correcto cálculo. En este caso se añade un torque de 1.

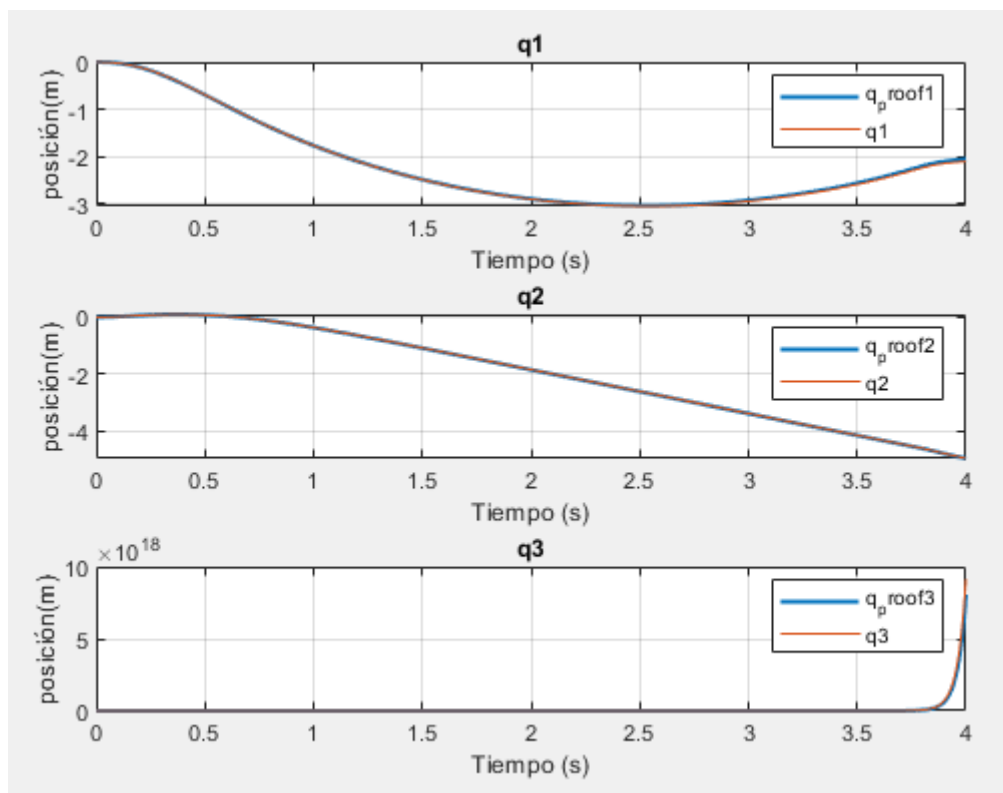


El modelo se basa en diferentes integradores concatenados para poder sacar los valores buscados. A la función que se use (el modelo es el mismo tanto para el modelo calculado como para el modelo de referencia) le entra tanto el torque como la posición como la velocidad de las variables articulares del robot de 3 grados de libertad.

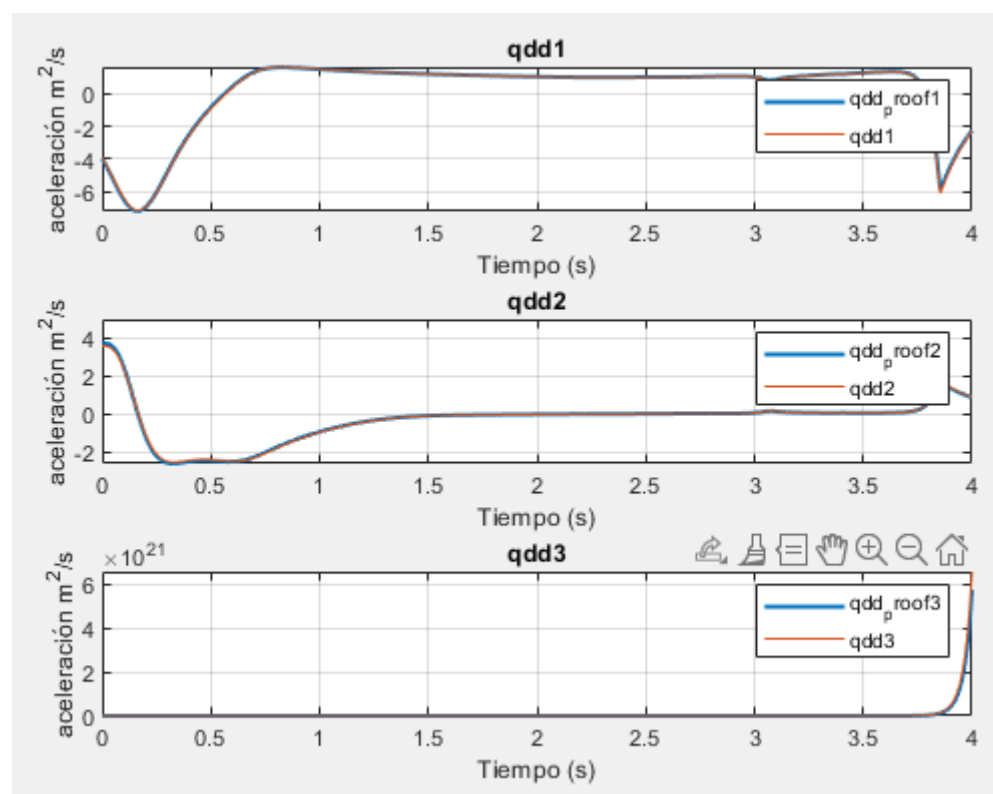
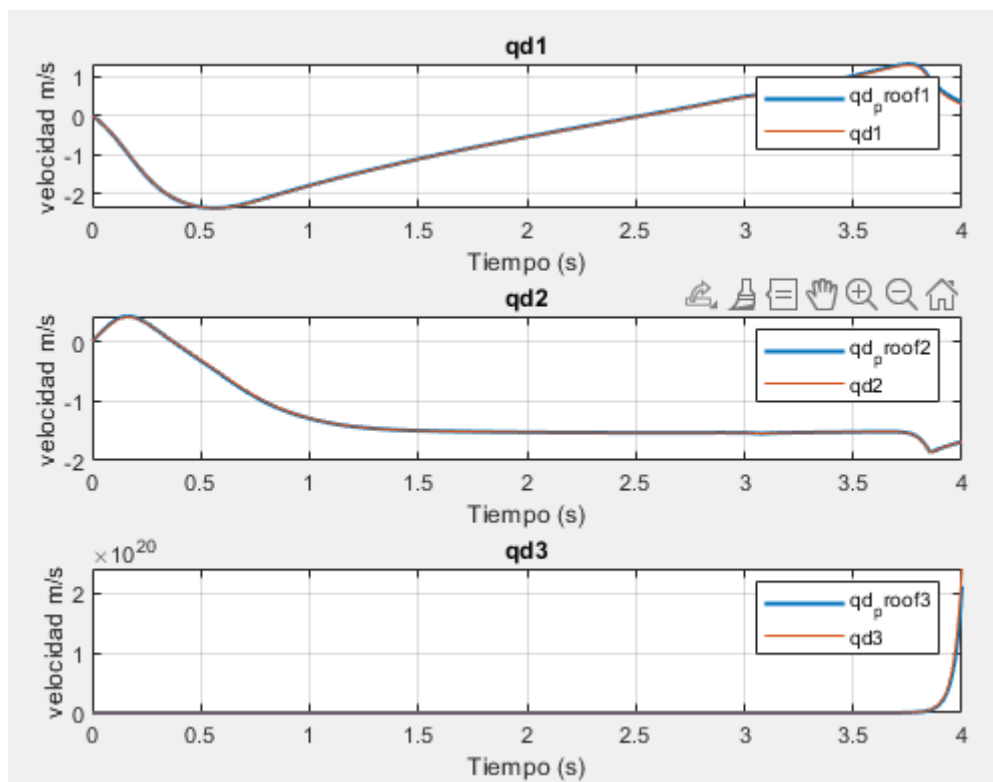


3.3 – Resultado Modelo Dinámico

Se recuperan las variables articulares en el archivo Matlab de esta parte del proyecto obteniendo estas gráficas:



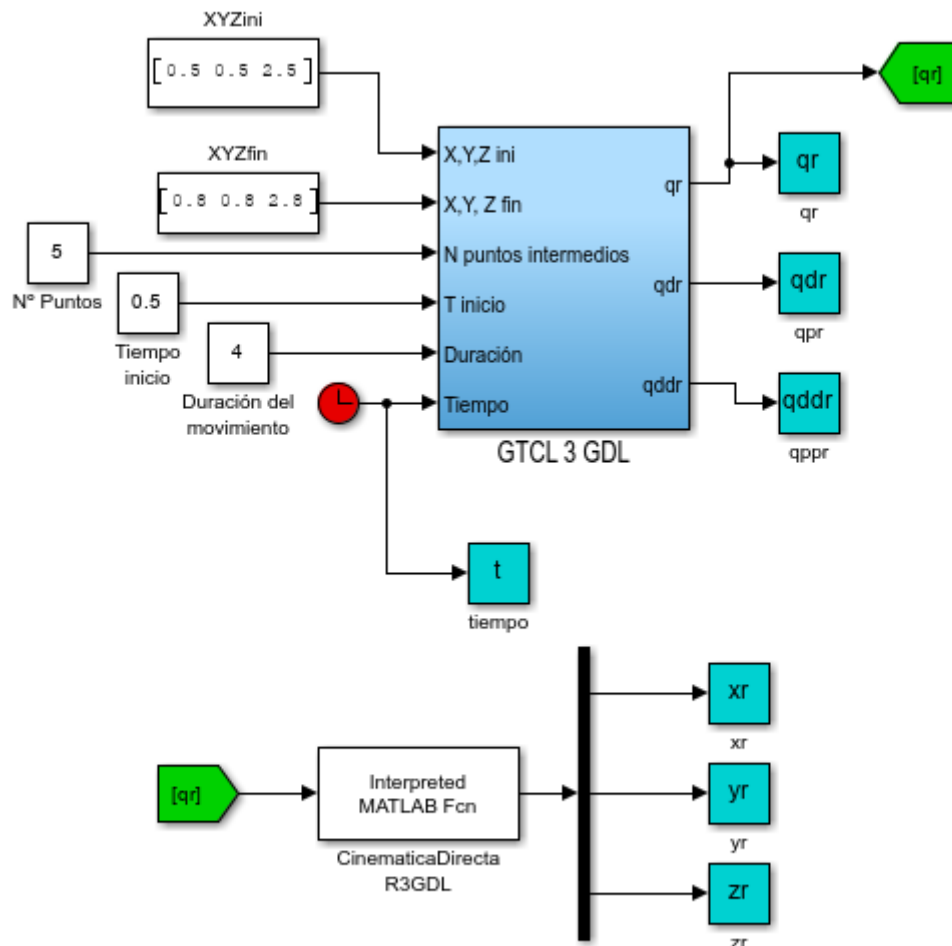
De los resultados obtenidos, hay que tener en cuenta que q_3 está limitada entre 0 y 1.5 por tanto nunca llegará a valores tan altos en la posición como se observa al final de la simulación.



4 – Control Cinemático

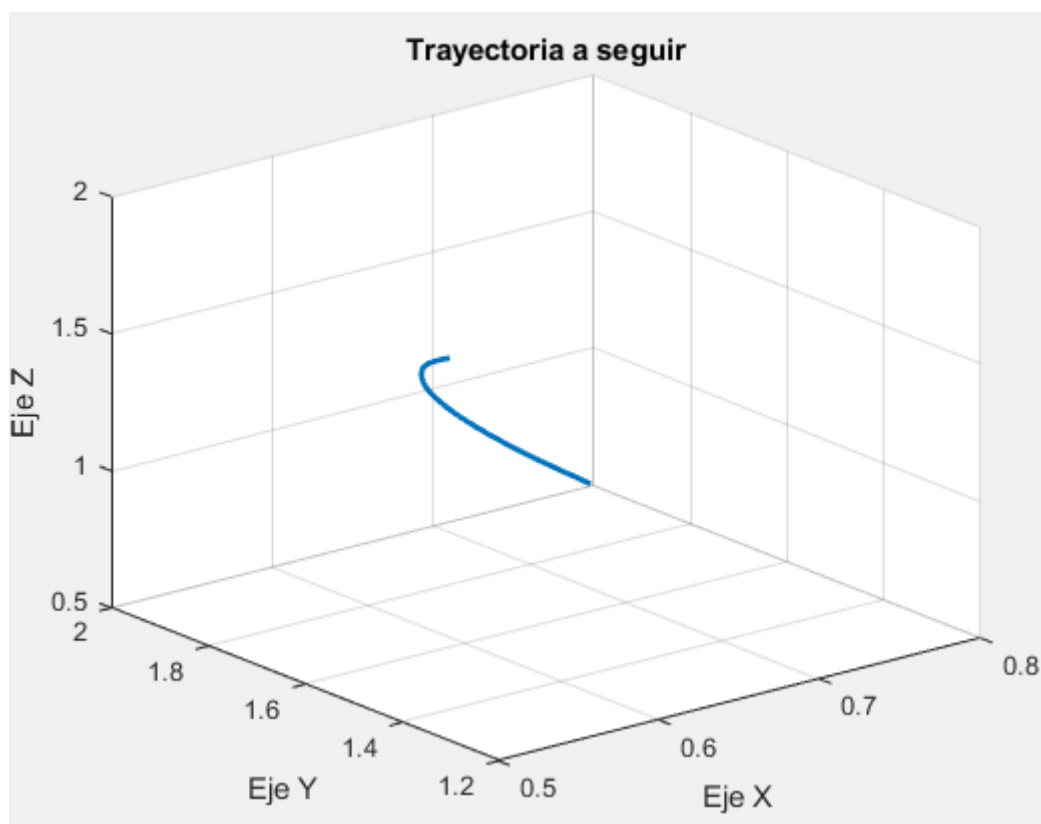
En cuanto a el esquema de control, se establece un simulador “Control_Cinematico.mdl” el cual permite generar una trayectoria referencia a seguir por el robot. En este caso desde la posición HOME [0.5; 0.5; 2.5] hasta un incremento de 0.3 en coordenadas cartesianas [0.8; 0.8; 2.8].

En cuanto al funcionamiento del simulador; se genera una trayectoria a partir de: el punto inicial HOME, como del final [0.8; 0.8; 2.8], el n° de puntos a dibujar (5 puntos), el tiempo de inicio en el que el robot se empezará a mover (0.5 segundos) y la duración del movimiento (4 segundos).

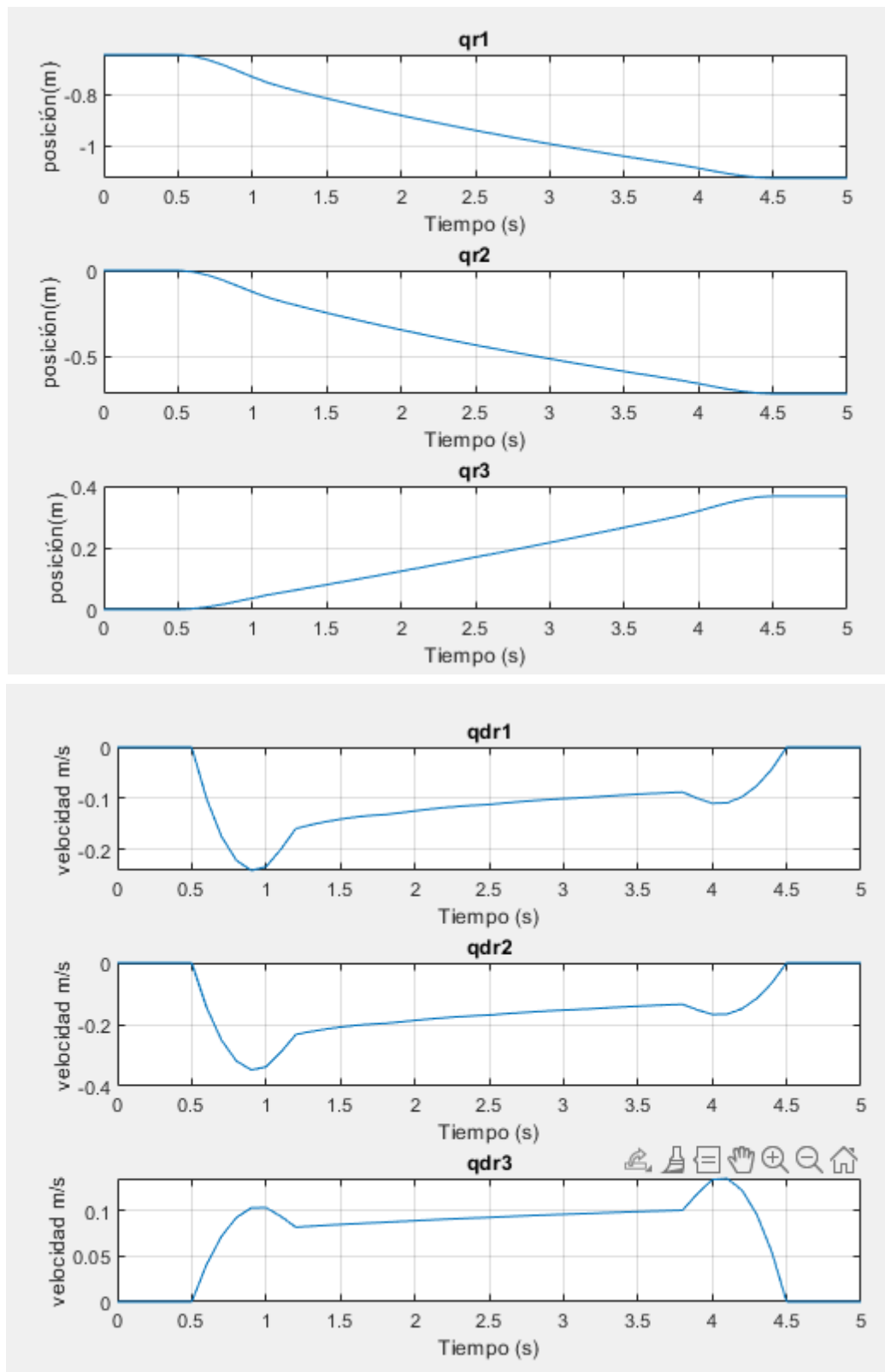


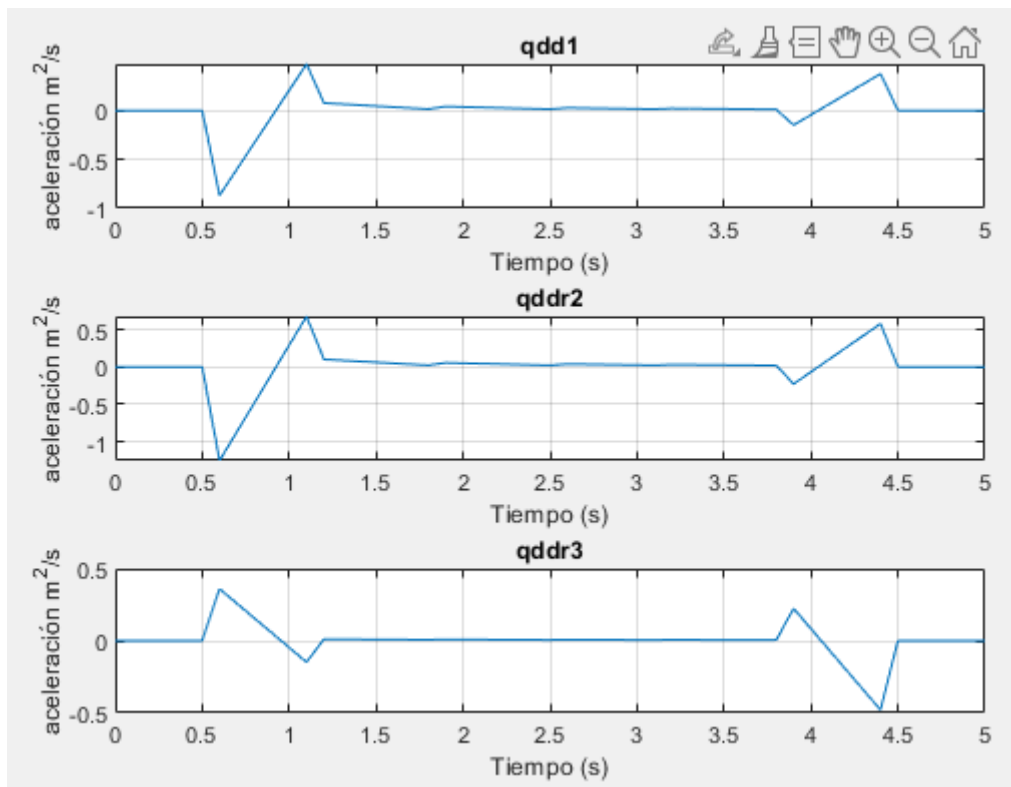
Este bloque devuelve el valor de cada variable articular (posición velocidad y aceleración) de cada punto simulado. Una vez obtenido estas variables articulares, se llama a la cinemática directa del robot ("CinematicaDirecta.m") para obtener las coordenadas y dibujar por tanto la referencia en el plano de 3 dimensiones.

Inicializando el programa "PROYECTO_C.m" se obtiene esta trayectoria del efector final:



Y también la evolución de las variables articulares:





Se puede apreciar que la simulación respeta los rangos de las articulaciones, manteniendo q_1 y q_2 entre $-\pi$ y π , y q_3 entre 0 y 1.5. Además, se observa correctamente cuando empieza y termina la simulación (de 0.5s a 4.5s).

5 – Control Dinámico

El control de dinámica de un brazo robótico se refiere a cómo se calculan y aplican las fuerzas y torques necesarios para que el brazo siga un movimiento deseado, teniendo en cuenta su dinámica física.

En el control cinemático solo se considera la geometría del movimiento (posiciones, velocidades), pero no se consideran fuerzas o masas.

Una vez modelado la dinámica del brazo robótico, se procede a linealizar su modelo según diferentes consideraciones, ya que se simplifica el diseño de controladores.

En cuanto a la linealización del modelo, las consideraciones a tener en cuenta son:

- Velocidades y aceleraciones en equilibrio

$$\dot{q}_{eq} = 0 \text{ unidades velocidad articular}$$

$$\ddot{q}_{eq} = 0 \text{ unidades aceleración articular}$$

-Linealización de términos de la matriz de Inercia:

Se desprecian los términos no diagonales con el fin de desacoplar el sistema

$$M_A(q_{eq}) \approx \text{diag}(M_A(q_{eq}))$$

Elección valores de inercia máximos de los términos de la diagonal de $M_A(q)$

-Linealización sobre los términos centrípetos y de Coriolis:

$$C(q, \dot{q}_{eq}) = C(q, 0) = 0$$

No hay términos lineales aportados por $V(q, \dot{q}) = C(q, \dot{q}) \dot{q}$

- Los términos gravitatorios se consideran como perturbaciones mantenidas a la entrada
- Se desprecian los términos de fricción

$$F(q, \dot{q}) \approx 0$$

Una vez linealizado, se procede a la creación de las funciones de transferencia de cada articulación, que básicamente es representar las funciones de transferencia como:

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

Siendo a_i cada término de la diagonal de la matriz de inercia linealizada en su máximo rango, y siendo b_i los términos lineales de la matriz de aceleraciones centrípetas y de Coriolis:

Mamax =		Vamax =	
269.4110	0	0	0.0406
0	315.6100	0	0.1412
0	0	69.8630	0.1840

Obteniendo estas funciones de transferencia:

G11 =	G22 =	G33 =
$\frac{1}{269.4 s^2 + 0.04064 s}$	$\frac{1}{315.6 s^2 + 0.1412 s}$	$\frac{1}{69.86 s^2 + 0.184 s}$

5.1 – Diseño de Controladores

El objetivo del diseño de controladores es regular el comportamiento de un sistema dinámico para que siga una trayectoria deseada. Un controlador actúa sobre el sistema ajustando sus entradas para lograr que las salidas sigan una referencia o rechacen perturbaciones.

En un controlador se busca estabilidad del sistema, precisión en el seguimiento, comportamiento robusto ante variaciones del modelo...

Por tanto, se proponen 4 controladores diferentes cada uno con sus características particulares con los cuales se pretende controlar el robot de 3gdl propuesto.

En resumen, diseñar un controlador es encontrar una función de realimentación que garantice que el sistema cumpla ciertos criterios dinámicos y de precisión, teniendo en cuenta tanto el modelo como sus limitaciones físicas.

Todos los controladores utilizados en el proyecto son desarrollados en un archivo MATLAB llamado "PROYECTO_D".

5.1.1 – PD descentralizado

En primer lugar, se propone un controlador PD descentralizado sin compensación de gravedad.

Su objetivo es controlar cada articulación por separado, sin necesidad de conocer ni compensar la dinámica completa del sistema ya que no requiere el modelo completo del sistema (ni la matriz de Inercia M_a , ni los términos centrípetos y de Coriolis V_a ni el par gravitatorio G_a) además de su bajo coste computacional.

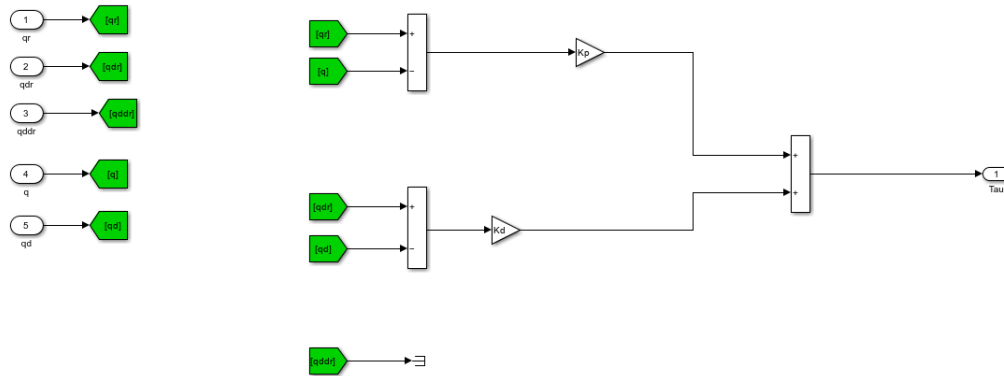
Modelo del sistema:

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

Control:

$$\begin{aligned}\tau(t) &= K_p \tilde{q}(t) + K_D \dot{\tilde{q}}(t) = K_p (\tilde{q}(t) + T_D \dot{\tilde{q}}(t)) \\ \tilde{q}(t) &= q_r(t) - q(t) \quad K_p = \text{diag}(K_{p1}, \dots, K_{pn}), \quad K_D = \text{diag}(K_{D1}, \dots, K_{Dn})\end{aligned}$$

Esquema de implementación:



En cuanto a la expresión del controlador, se ha decidido hacer un PD sin cancelación de dinámica.

Se saca por lugar de las raíces con el objetivo de ajustar el comportamiento del sistema en lazo cerrado según los requerimientos establecidos. En este caso se decide un tiempo de subida de $t_{sbc} = 0,1$ segundos, obteniendo una frecuencia natural de $\omega_n = 30$ lo que decide ubicar los polos del sistema de lazo cerrado en $p_{bc} = -30$.

Por tanto, el controlador se formará de un único cero que aporta el efecto derivativo, y una ganancia proporcional ajustada para obligar a los polos de bucle cerrado a caer en el lugar deseado, obteniendo:

$$K_d = \frac{1.0e+04}{1.6230 + 1.9031s + 0.4181s^2} \quad K_p = \frac{244800}{284040 + 62400s}$$

5.1.2 – PD descentralizado con compensación de gravedad

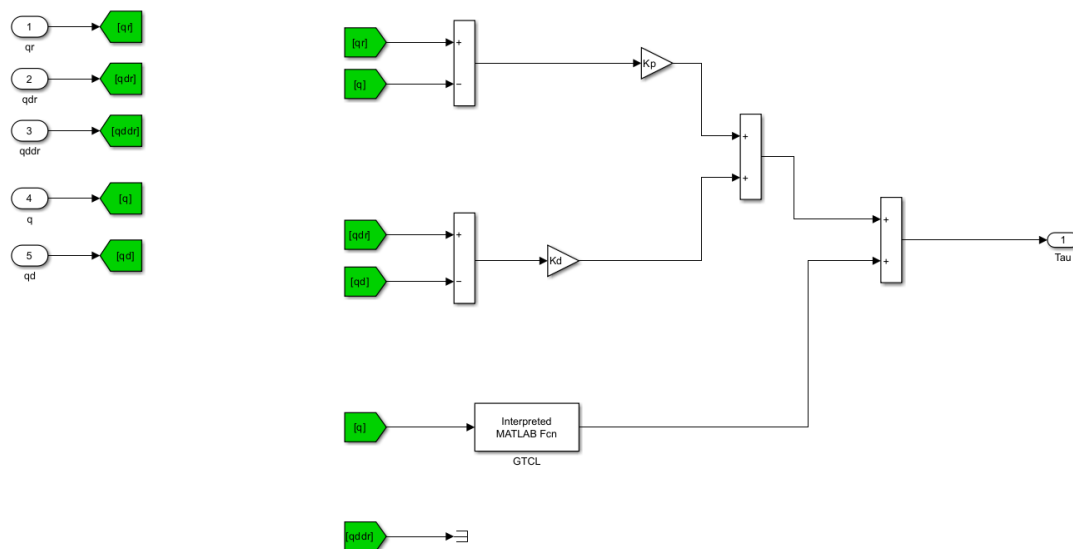
En segundo lugar, se propone el mismo controlador PD pero esta vez se pretende compensar la gravedad para atenuar las perturbaciones.

En el PD sin compensación, el controlador debe generar un esfuerzo adicional para “sujetar” el brazo, además de moverlo. Modificando la ley de control, añadiendo el par gravitatorio se compensa el peso del robot, reduciendo el esfuerzo del control y la precisión.

El control queda como:

$$\tau = K_p \tilde{q} + K_D \dot{\tilde{q}} + G_A(q)$$

Esquema de implementación:



En este caso se ha creado una función “Compensacion.m” la cual para cualquier valor de la posición de las variables articulares devuelve el valor del par gravitatorio en ese instante:

```
function [Tau_add] = Compensacion(in)
% Variables de entrada en la funcion: [q(1) q(2) q(3)]

q1      = in(1);
q2      = in(2);
q3      = in(3);

g=9.8;

% Par gravitatorio
Ga=[-1.0*g*(172.19*sin(q1) - 90.195*cos(q1)*cos(q2) + 32.798*cos(q1)*sin(q2) + 65.596*q3*cos(q1)*sin(q2));
    -8.1996*g*sin(q1)*(4.0*cos(q2) + 11.0*sin(q2) + 8.0*q3*cos(q2));
    -65.596*g*sin(q1)*sin(q2)];

Tau_add=Ga;
```

5.1.3 – PID descentralizado

En tercer lugar, se propone un controlador PID descentralizado sin compensación de gravedad.

Su objetivo es corregir el error, comparando la salida del sistema con la referencia, y ajustando la entrada según cuánto error hay, cuánto tiempo ha durado ese error y qué tan rápido cambia el error; sin necesidad de conocer ni compensar la dinámica completa del sistema ya que no requiere el modelo completo del sistema (ni la matriz de Inercia M_a , ni los términos centrípetos y de Coriolis V_a ni el par gravitatorio G_a).

Modelo del sistema:

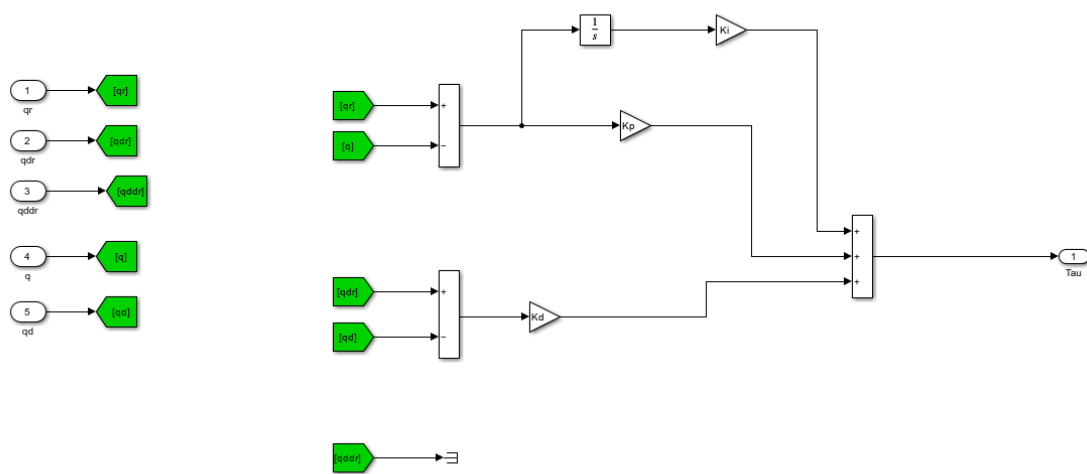
$$\tau = M_A(q)\ddot{q} + C_A(q, \dot{q})\dot{q} + G_A(q)$$

Control:

$$\tau(t) = K_P \tilde{q}(t) + K_D \dot{\tilde{q}}(t) + K_I \int_0^t \tilde{q}(\tau) d\tau = K_P \left(\tilde{q} + T_D \dot{\tilde{q}} + T_I^{-1} \int_0^t \tilde{q}(\tau) d\tau \right)$$

$$\tilde{q}(t) = q_r(t) - q(t) \quad K_P = \text{diag}(K_{P1}, \dots, K_{Pn}), \quad K_D = \text{diag}(K_{D1}, \dots, K_{Dn}), \quad K_I = \text{diag}(K_{I1}, \dots, K_{In})$$

Esquema de implementación:



En cuanto a la expresión del controlador, se ha decidido hacer un PID sin cancelación de dinámica.

Se saca por lugar de las raíces con el objetivo de ajustar el comportamiento del sistema en lazo cerrado según los requerimientos establecidos. En este caso se decide un tiempo de subida de $t_{sbc} = 0,1$ segundos, obteniendo una frecuencia natural de $\omega_n = 60$ lo que decide ubicar los polos del sistema de lazo cerrado en $p_{bc} = -60$.

Por tanto, el controlador se formará de dos ceros, uno que introduce el efecto derivativo y otro que incorpora el efecto integral, y una ganancia proporcional ajustada para obligar a los polos de bucle cerrado a caer en el lugar deseado, obteniendo:

Kd =	Kp =	Ki =
1.0e+04 *	1492000	14920000
3.6703	1704000	17040000
4.2600	377200	3772000
0.9430		

5.1.4 - Control de Par Calculado

Por último, se propone un controlador de par calculado.

A diferencia de los demás controladores, este compensa toda la dinámica del robot (compensa gravedad, Coriolis e Inercia) y transforma el sistema no lineal en un conjunto de integradores desacoplados (desacopla articulaciones).

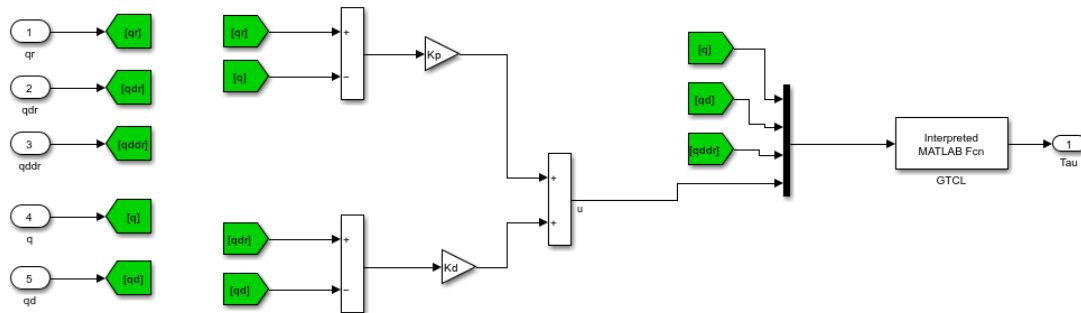
Además, se diseña un PD como control externo para el doble integrador del bucle interno resultante.

Control de bucle interno:

$$u = K_P \tilde{q} + K_D \dot{\tilde{q}}$$

$$\tau = M(q)(\ddot{q}_r + K_D(\dot{q}_r - \dot{q}) + K_P(q_r - q)) + C(q, \dot{q})\dot{q} + G(q)$$

Esquema de implementación:



En este caso se ha creado una función “ControlBucleInterno.m” la cual le entra las posiciones reales (q_1, q_2, q_3), las velocidades reales (q_{d1}, q_{d2}, q_{d3}), las aceleraciones de referencia ($q_{ddr1}, q_{ddr2}, q_{ddr3}$) y la señal auxiliar del control PD (u_1, u_2, u_3), devolviendo directamente el par a aplicar al robot de 3 gdl.

Sistema en lazo cerrado:

$$\ddot{\tilde{q}} = -u = -K_P \tilde{q} - K_D \dot{\tilde{q}}$$

Y su función de transferencia es:

$$G(s) = \frac{1}{s^2 + K_D s + K_P}$$

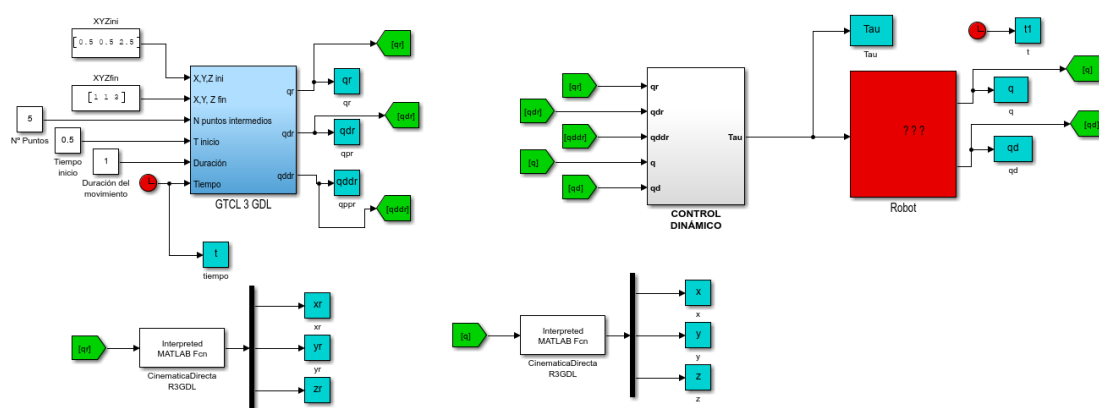
Comparando con un sistema de segundo orden: $K_D = 2\zeta\omega_n$, $K_P = \omega_n^2$

Obteniendo:

$K_d =$	$K_p =$
60	900
60	900
60	900

5.2 – Simulación de Controladores

En cuanto al simulador de los controladores se propone este formato:



En primer lugar, se genera la trayectoria deseada en espacio cartesiano a partir de la posición inicial HOME y una final correspondiente a un incremento en coordenadas articulares de valor 0.5, calculando para ello los vectores de posición, velocidad y aceleración de referencia en coordenadas articulares (q_r , q_{dr} , q_{ddr}). Estos valores alimentan el bloque de control dinámico, el cual también recibe el estado actual del robot (posición q y velocidad q_d)

En cuanto al control dinámico se calcula el par de control τ necesario para que el robot siga la trayectoria deseada implementando los controladores desarrollados anteriormente. Este par es aplicado al modelo dinámico del robot, que simula la respuesta del sistema mecánico.

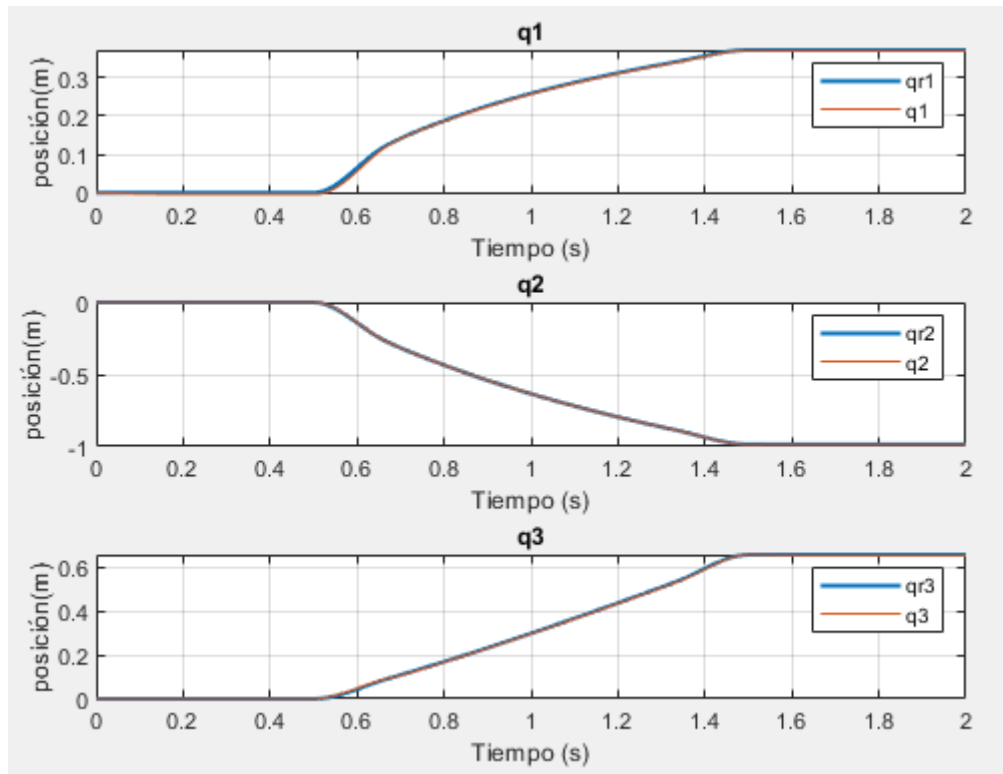
Por último, la cinemática directa también se implementa para calcular la posición cartesiana del extremo del robot a partir de las coordenadas articulares actuales (q_1 , q_2 , q_3) y así poder comparar visualmente la trayectoria seguida (x , y , z) frente a la de referencia (x_r , y_r , z_r).

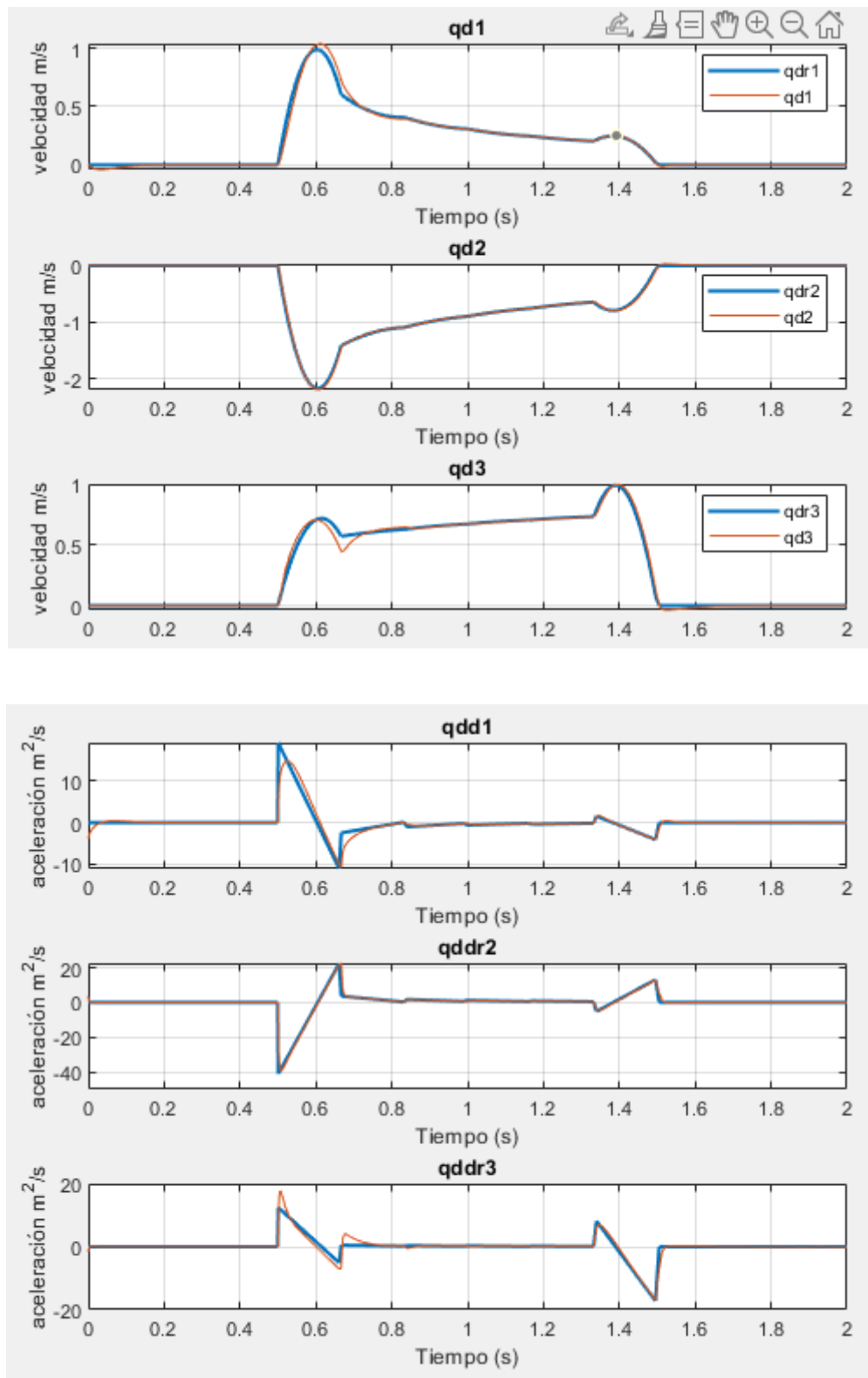
5.2.1 -PD descentralizado

Para este controlador se facilita la simulación

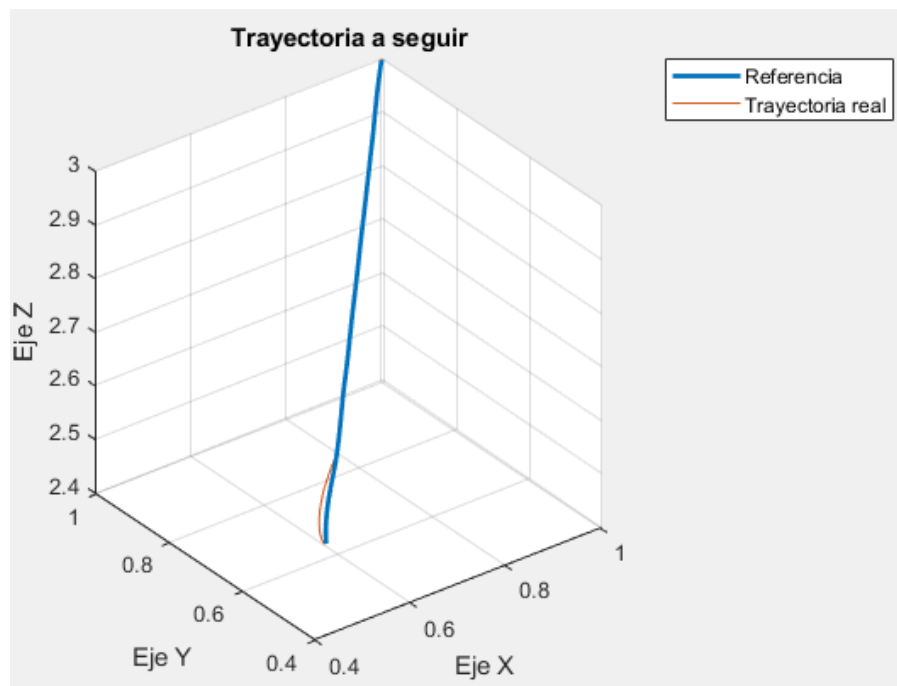
“Simulador_PD_sincompensacion.slx”, obteniendo las siguientes gráficas:

-Evolución de las coordenadas articulares:

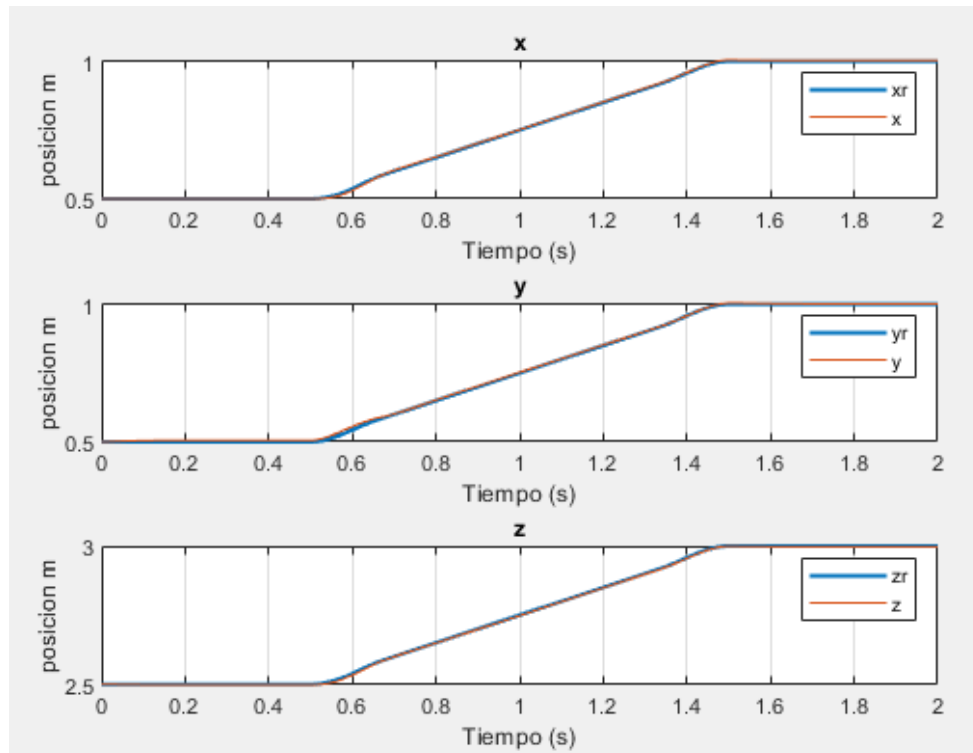




-Trayectoria a seguir:



-Trayectoria en función del tiempo:

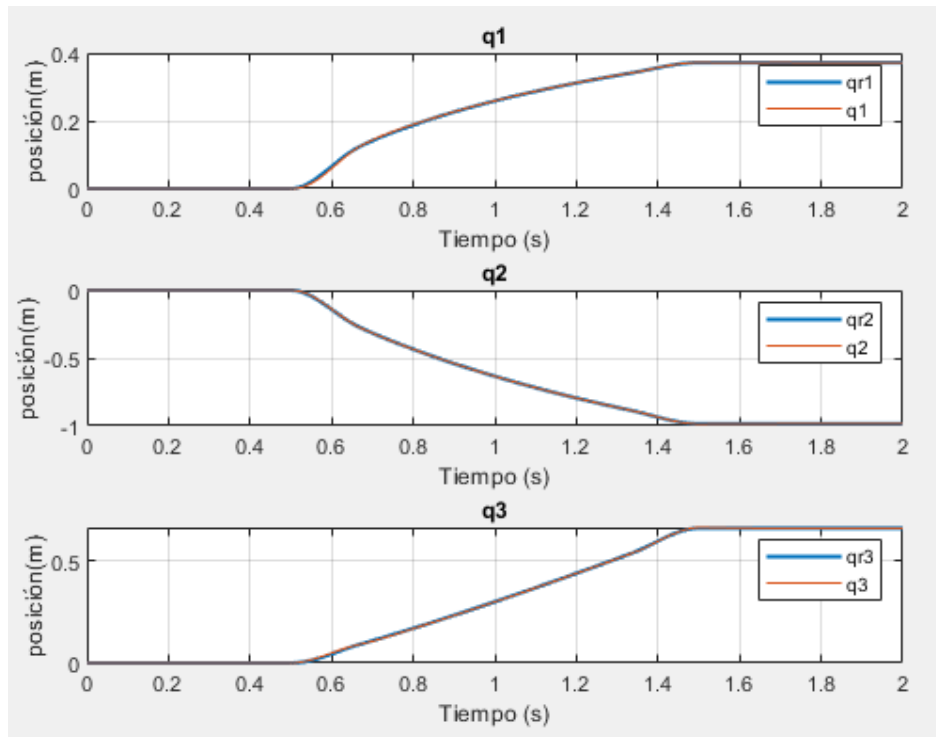


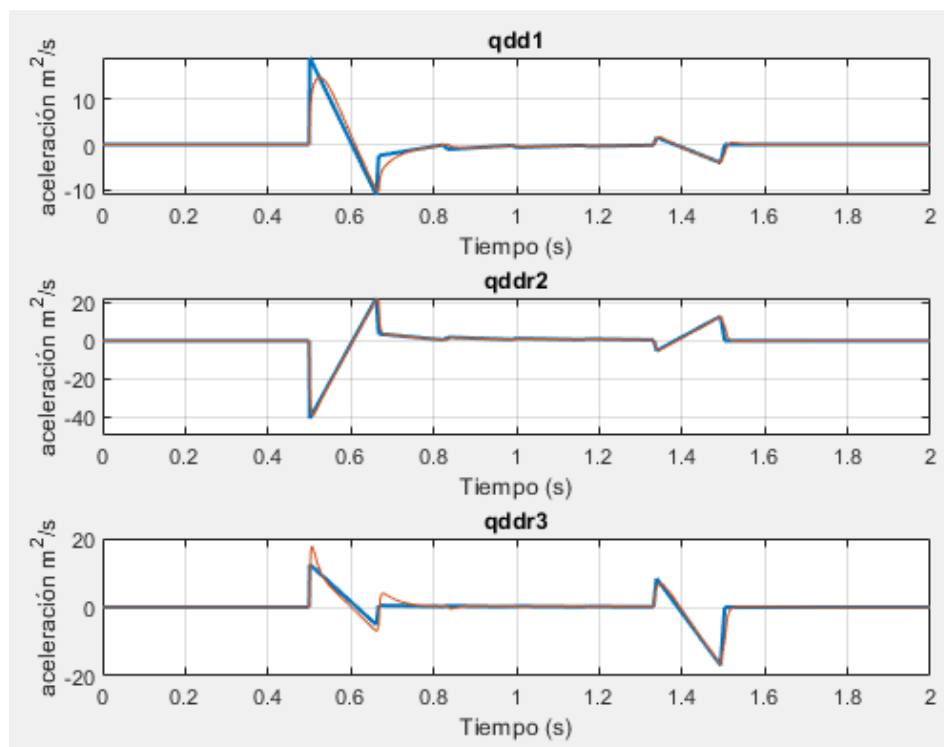
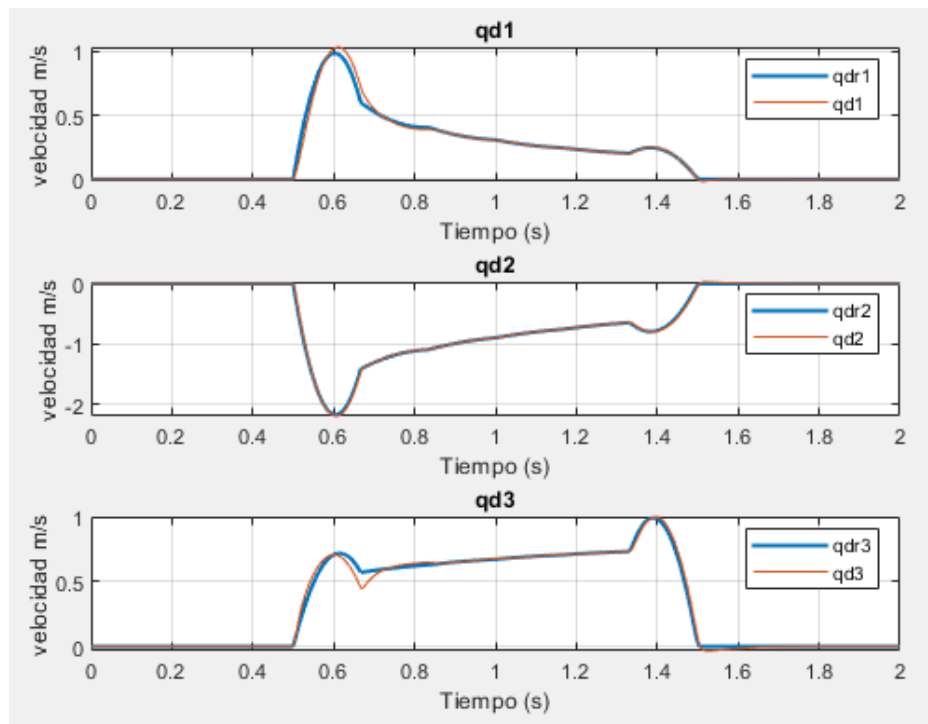
5.2.2 -PD descentralizado con compensación de gravedad

Para este controlador se facilita la simulación

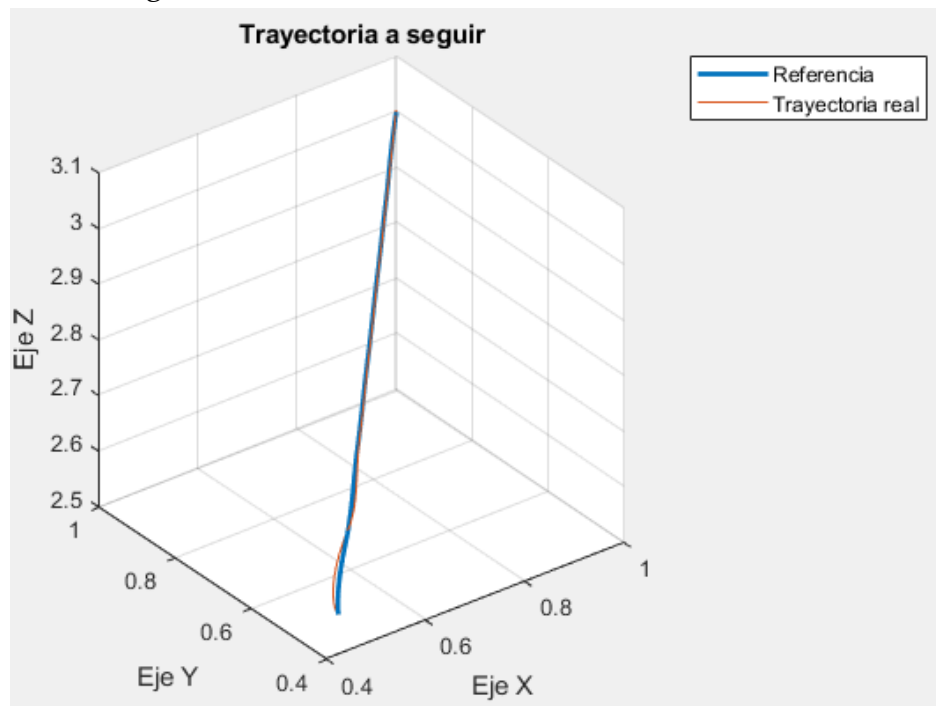
“Simulador_PD_concompensacion.slx”, obteniendo las siguientes gráficas:

-Evolución de las coordenadas articulares:

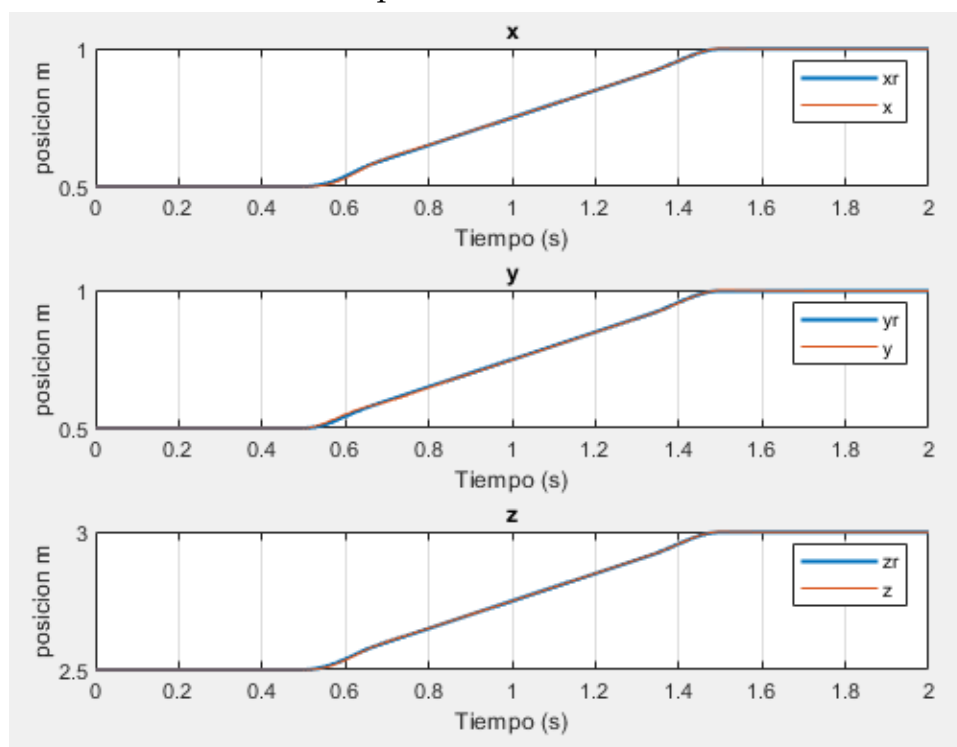




-Trayectoria a seguir:



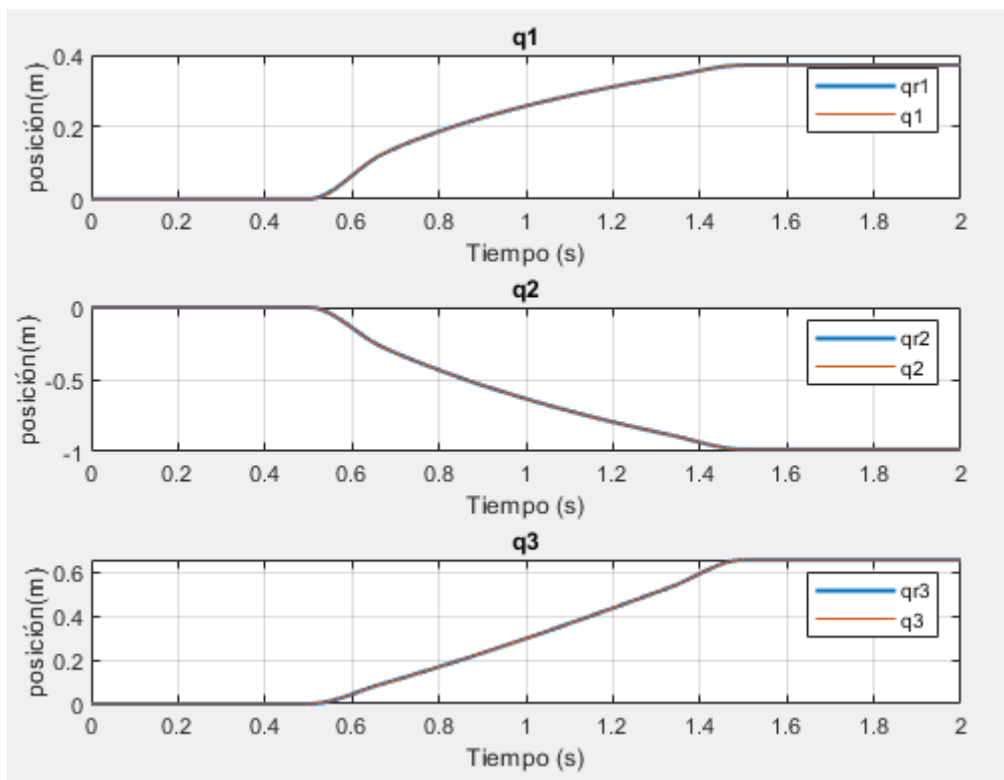
-Trayectoria en función del tiempo

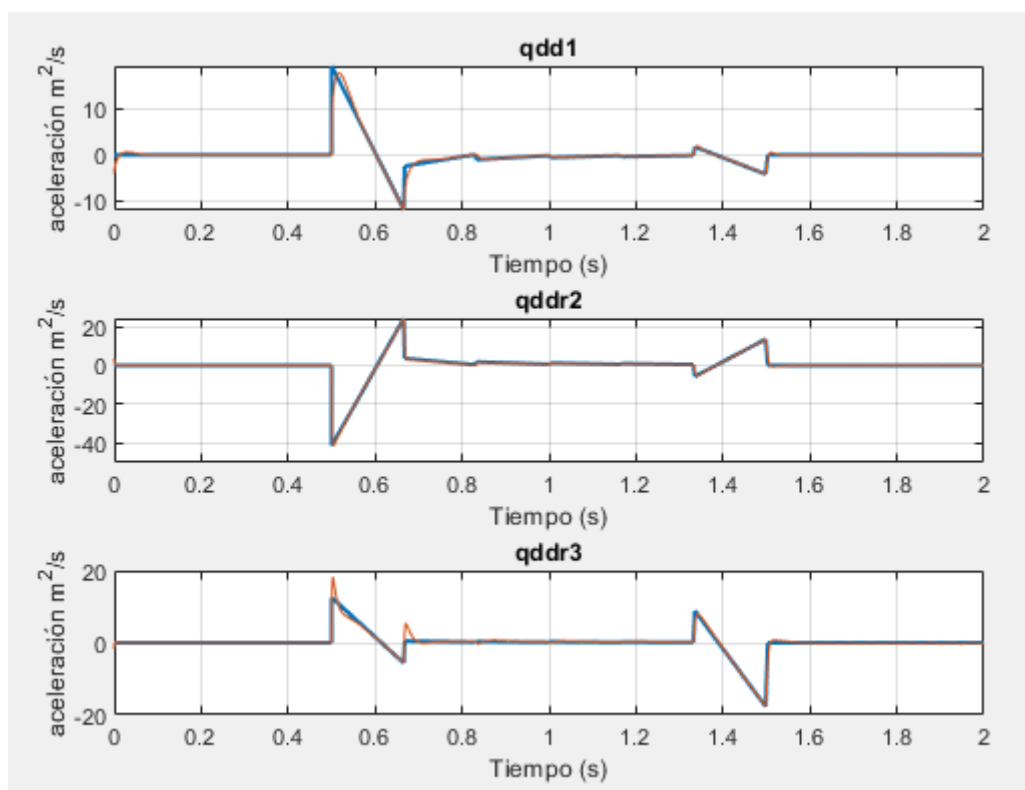
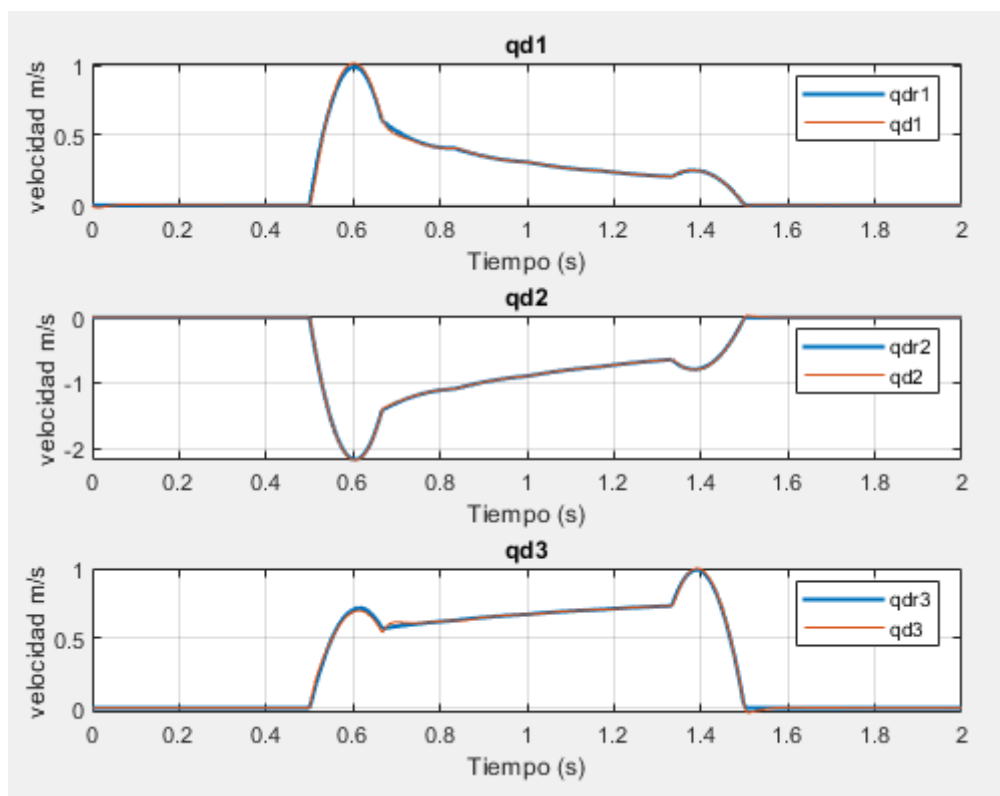


5.2.3 -PID descentralizado

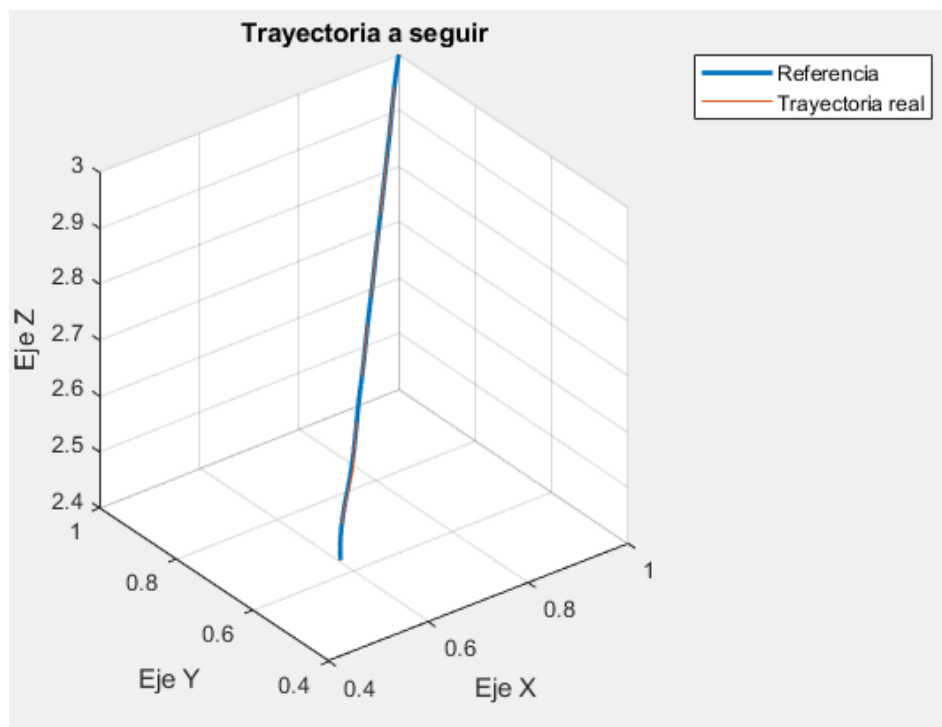
Para este controlador se facilita la simulación “Simulador_PID.slx”, obteniendo las siguientes gráficas:

-Evolución de las coordenadas articulares:

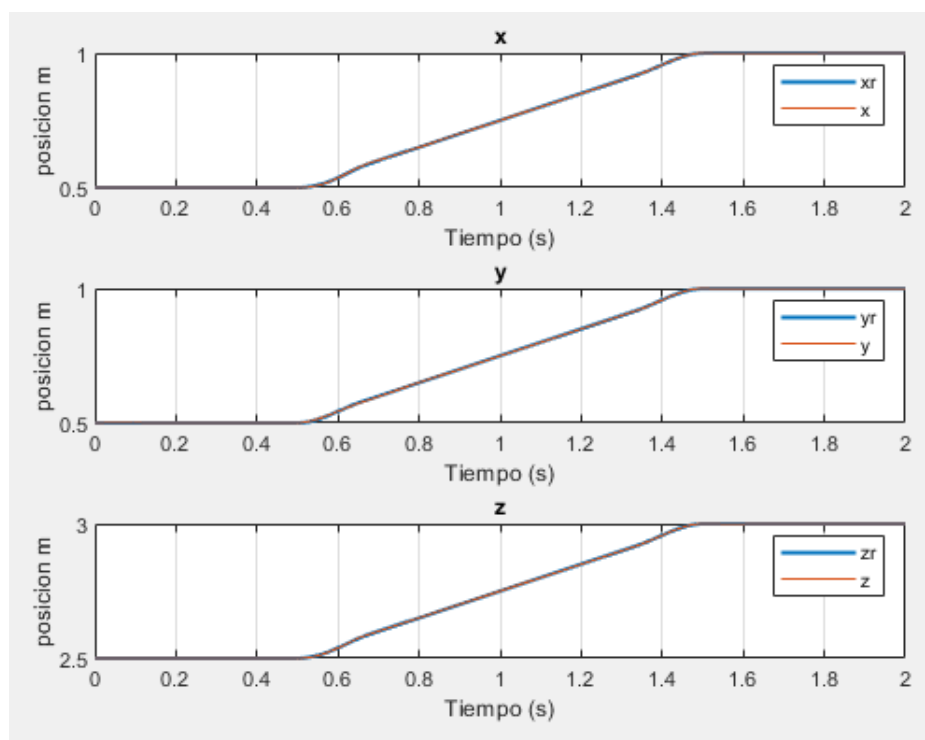




-Trayectoria a seguir:



-Trayectoria en función del tiempo:

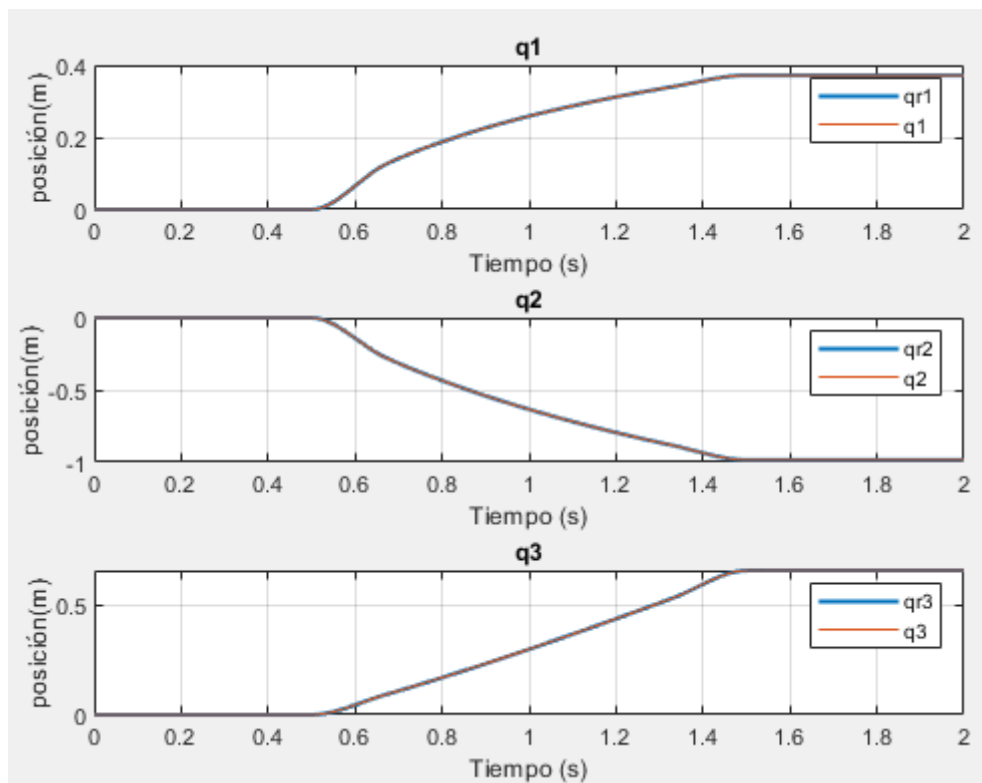


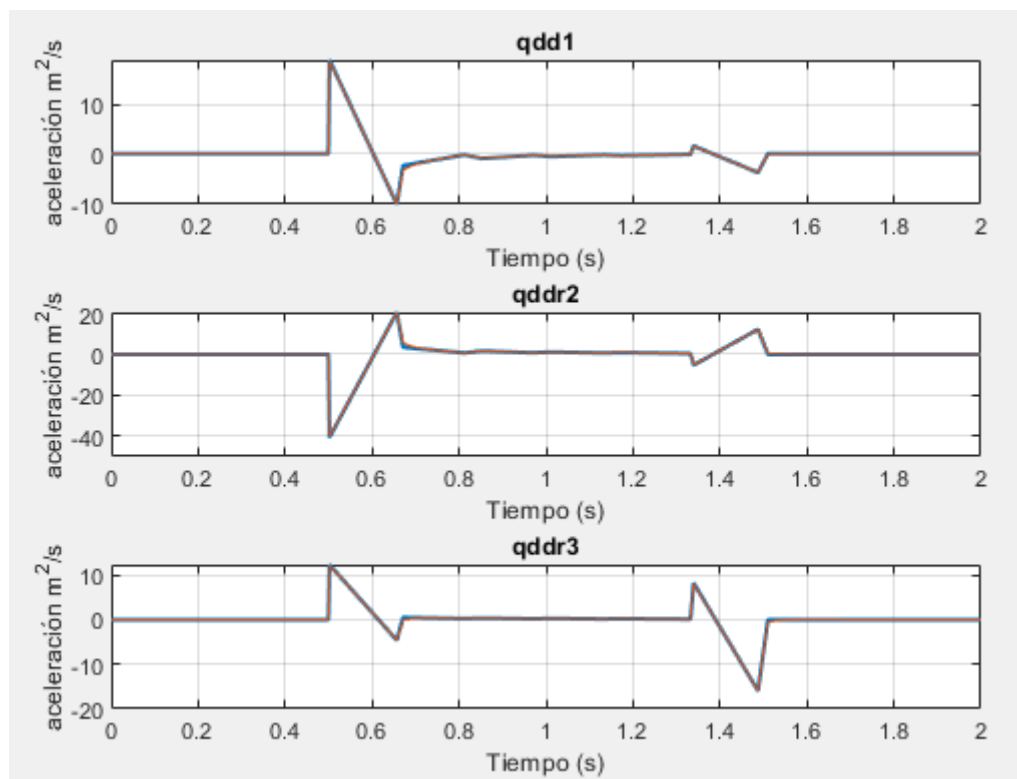
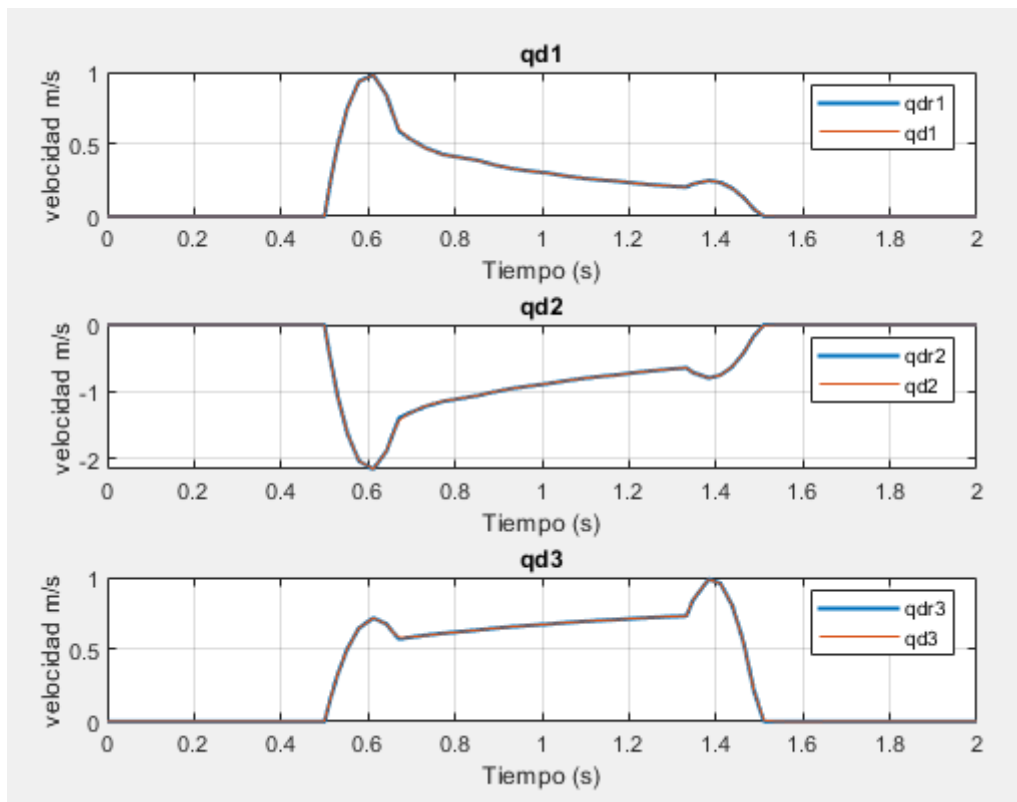
5.2.4 –Controlador de Par Calculado

Para este controlador se facilita la simulación

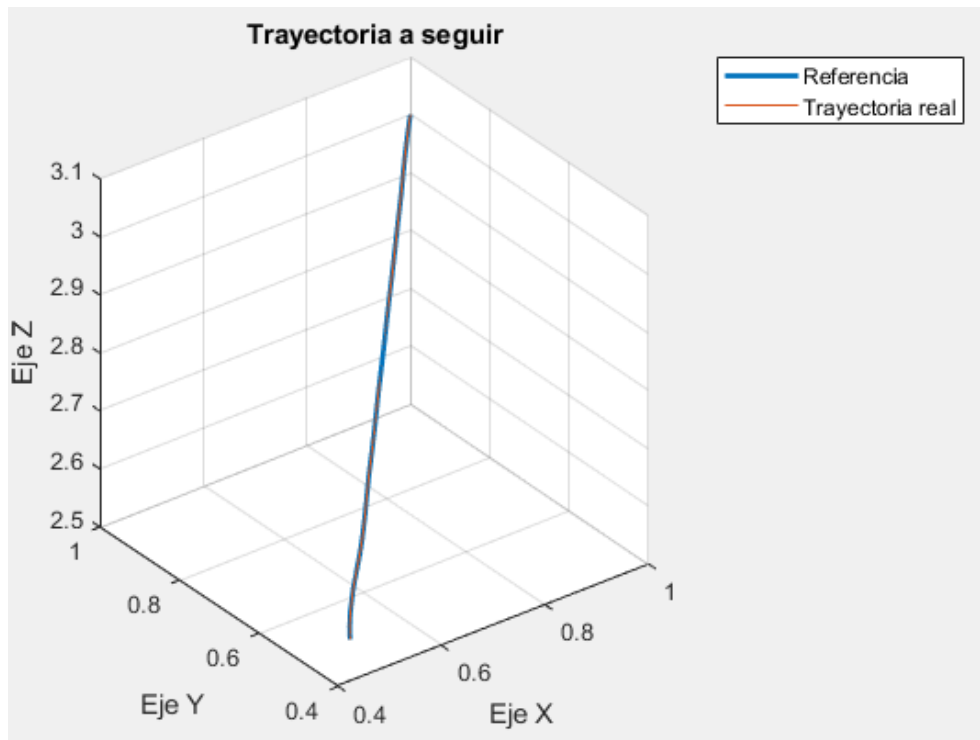
“Simulador_Control_Par_calculado.slx”, obteniendo las siguientes gráficas:

-Evolución de las coordenadas articulares:

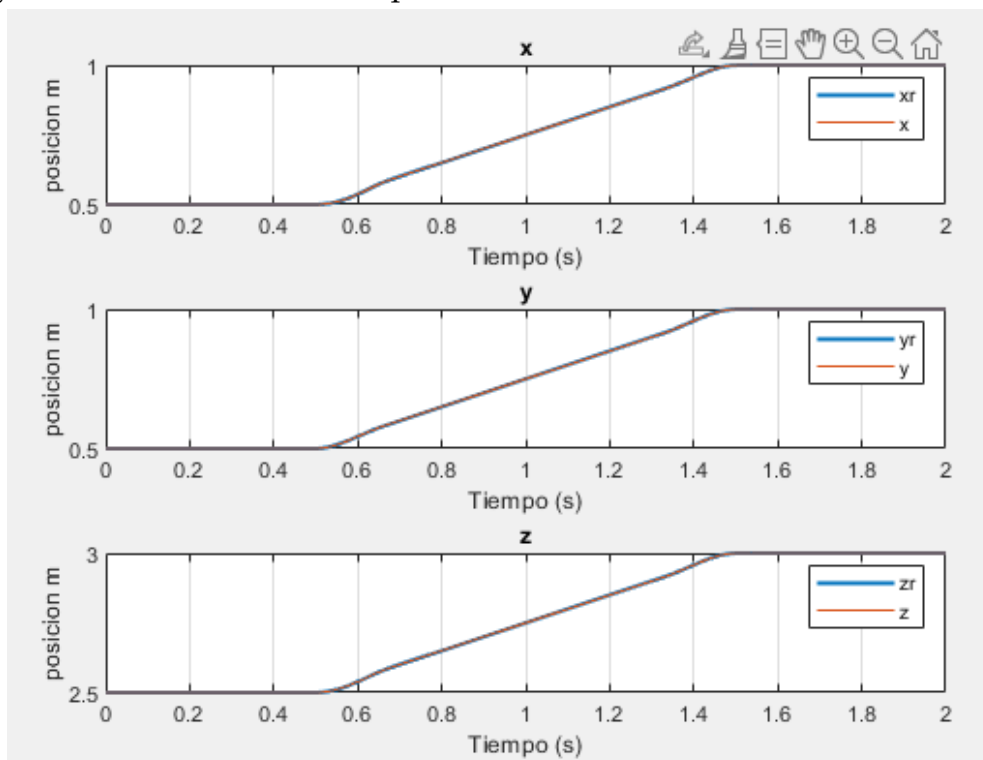




-Trayectoria a seguir:



-Trayectoria en función del tiempo:



5.3 – Análisis de Robustez

Para el análisis de robustez se debe tener en cuenta la masa del bloque que se va a añadir en el extremo del robot a la hora de sacar las matrices M_a , V_a y G_a , por tanto, se modifica el archivo NE_R3GDL.m, considerando que la masa no generara un par de fuerzas:

```
f44= [0 -30*g 0]';%Análisis de Robustez  
n44= [0 0 0]';
```

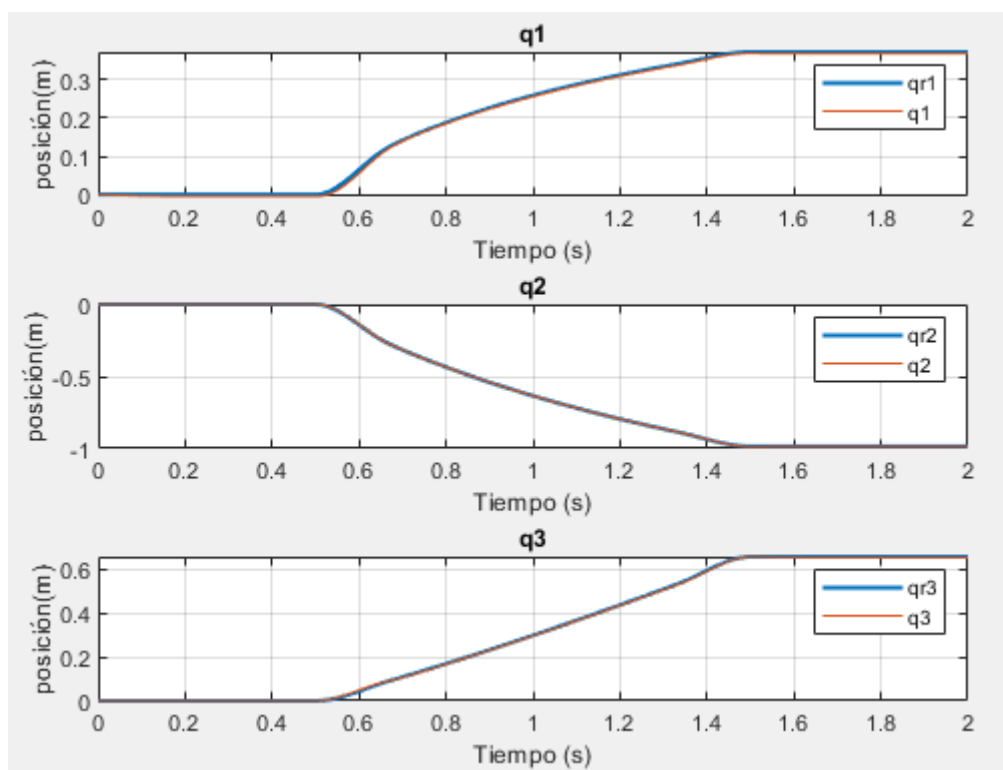
A continuación, se presentan de nuevo todas las gráficas necesarias de cada controlador.

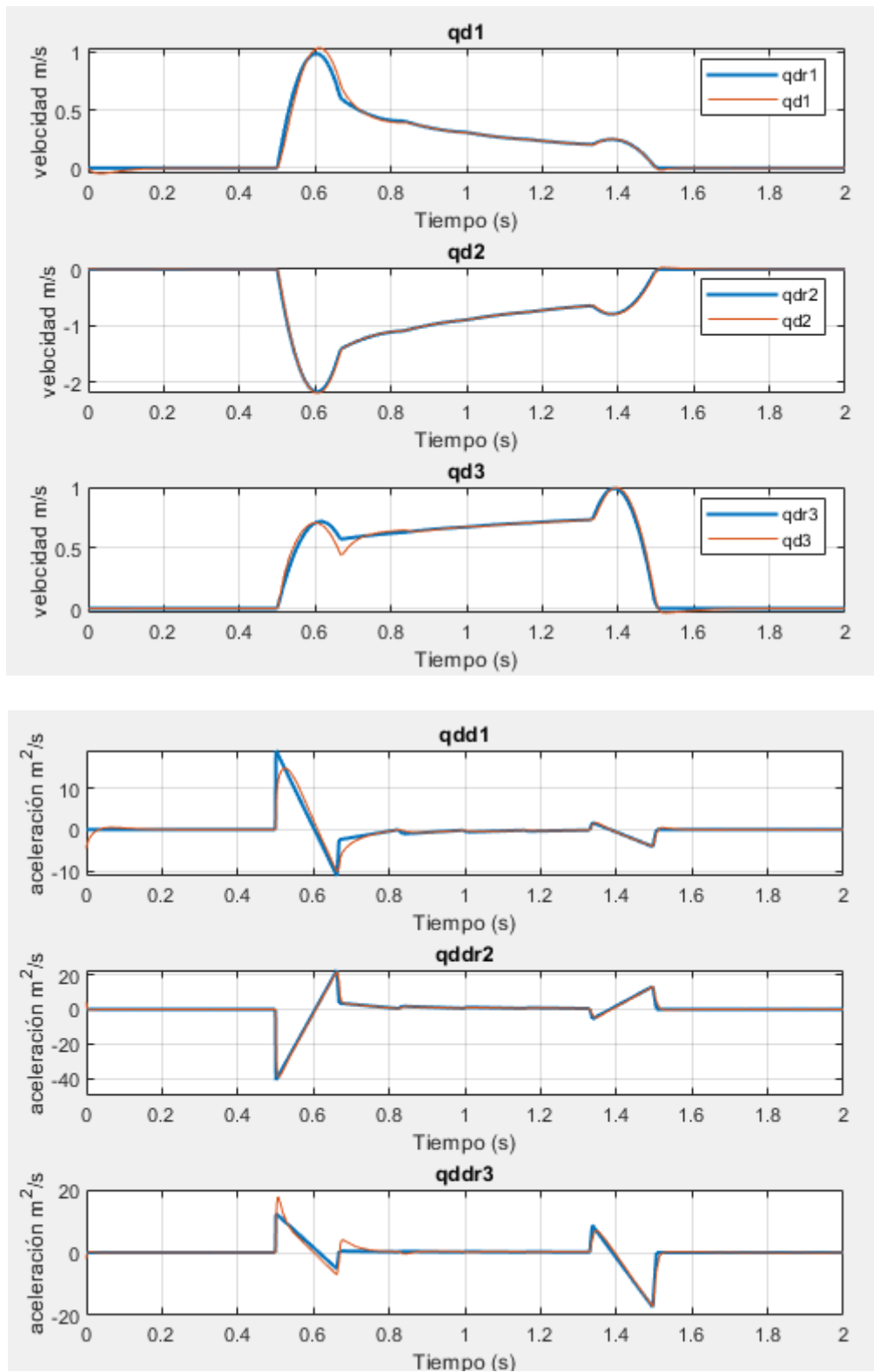
Para generar las gráficas se usa el mismo programa MATLAB “PROYECTO_D.m” cambiando la simulación en cada caso.

5.3.1 - PD descentralizado

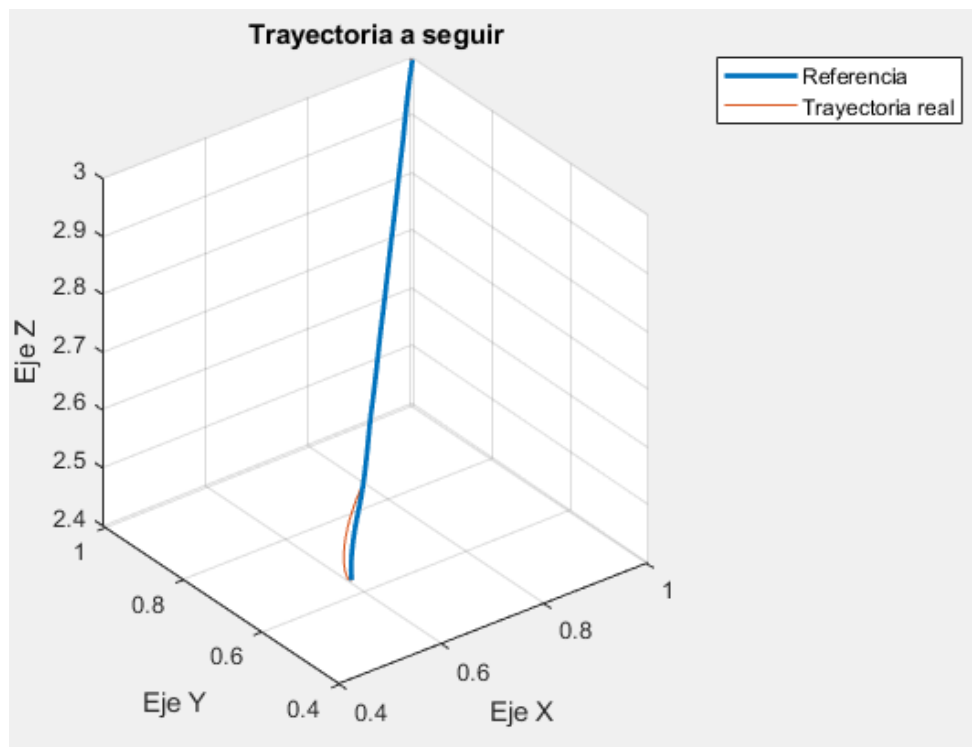
Para este controlador se facilita la simulación “Simulador_PD_sincompensacion_Robustez.slx”, obteniendo las siguientes gráficas:

-Evolución de las coordenadas articulares:

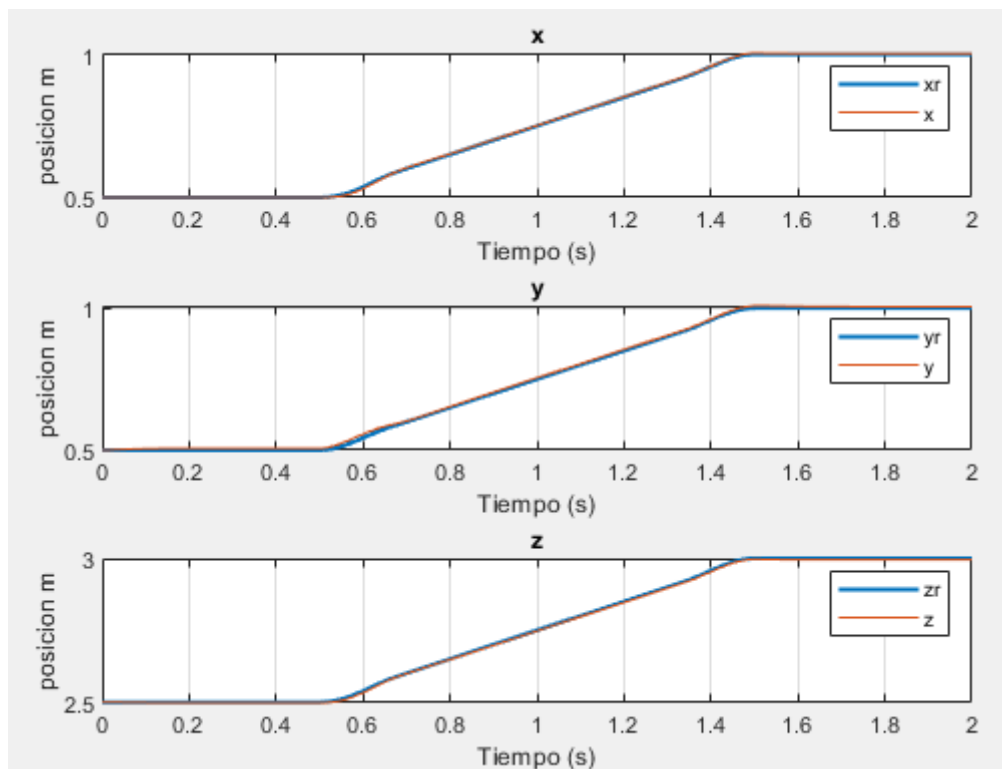




-Trayectoria a seguir:



-Trayectoria en función del tiempo:

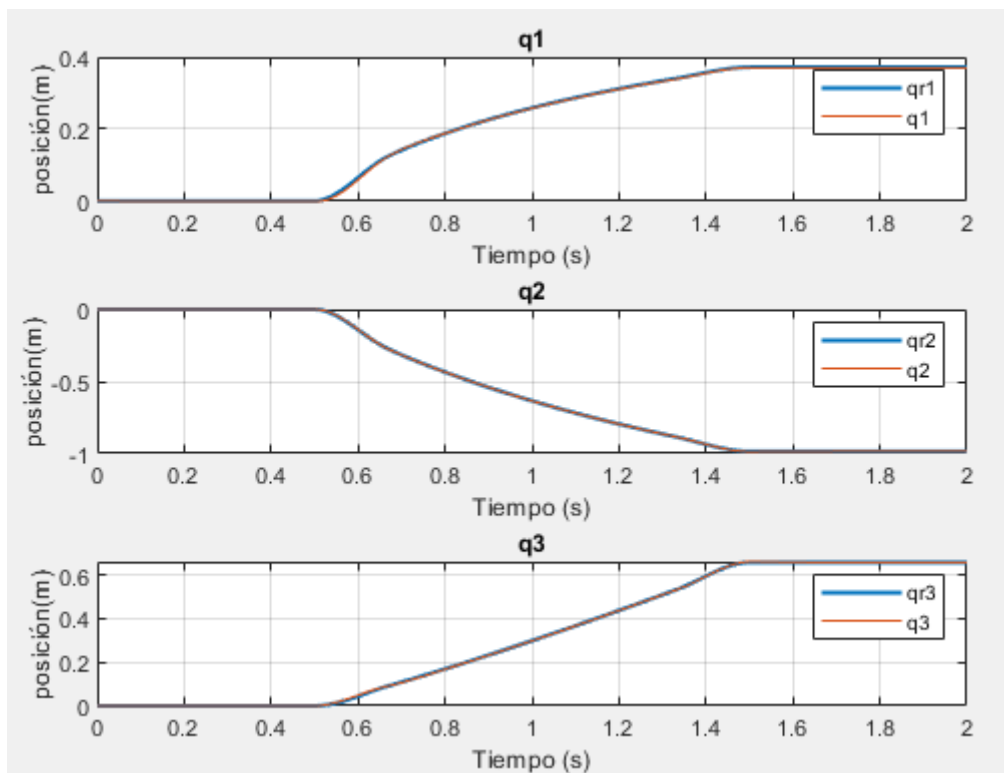


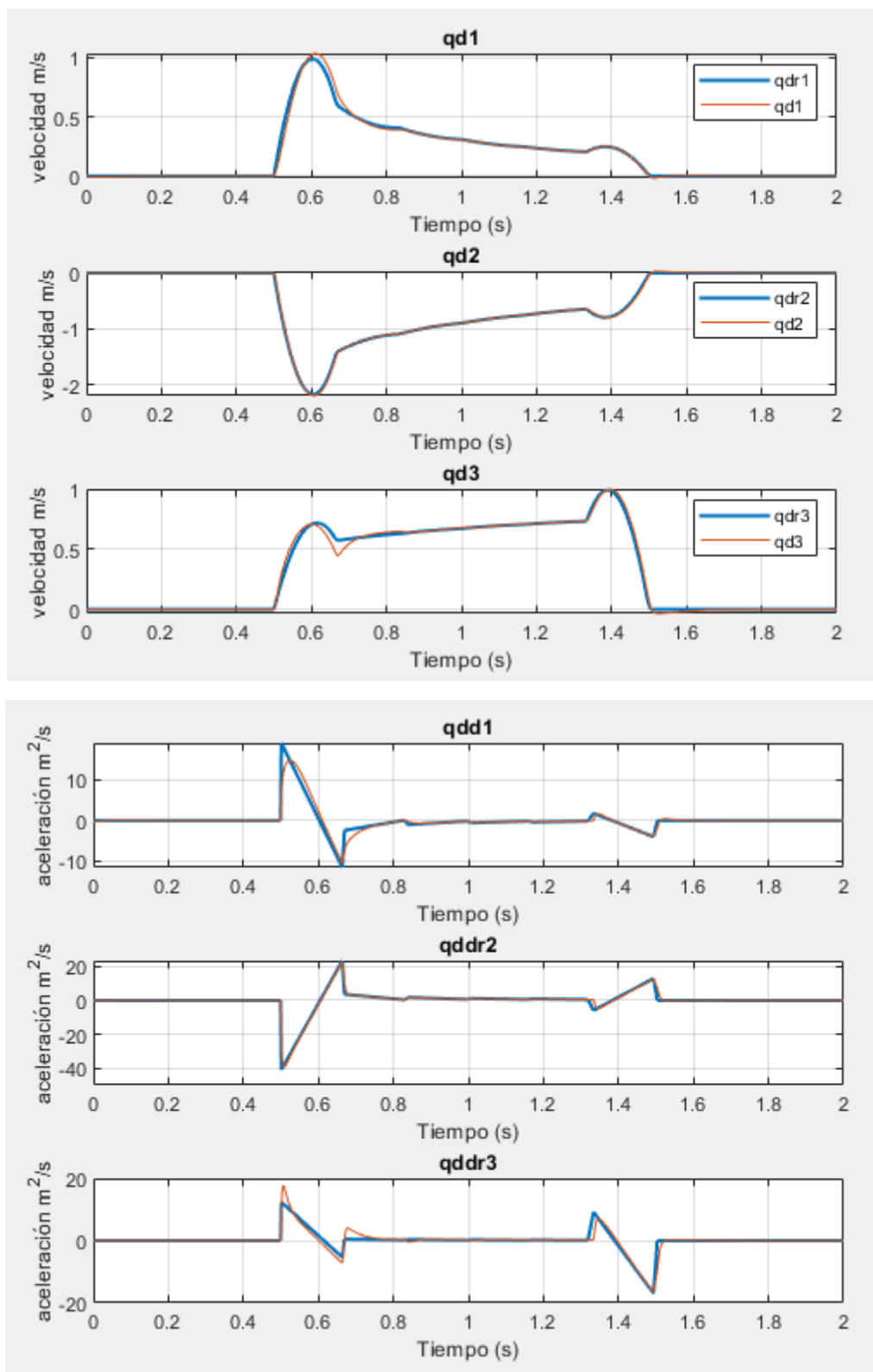
5.3.2 -PD descentralizado con compensación de gravedad

Para este controlador se facilita la simulación

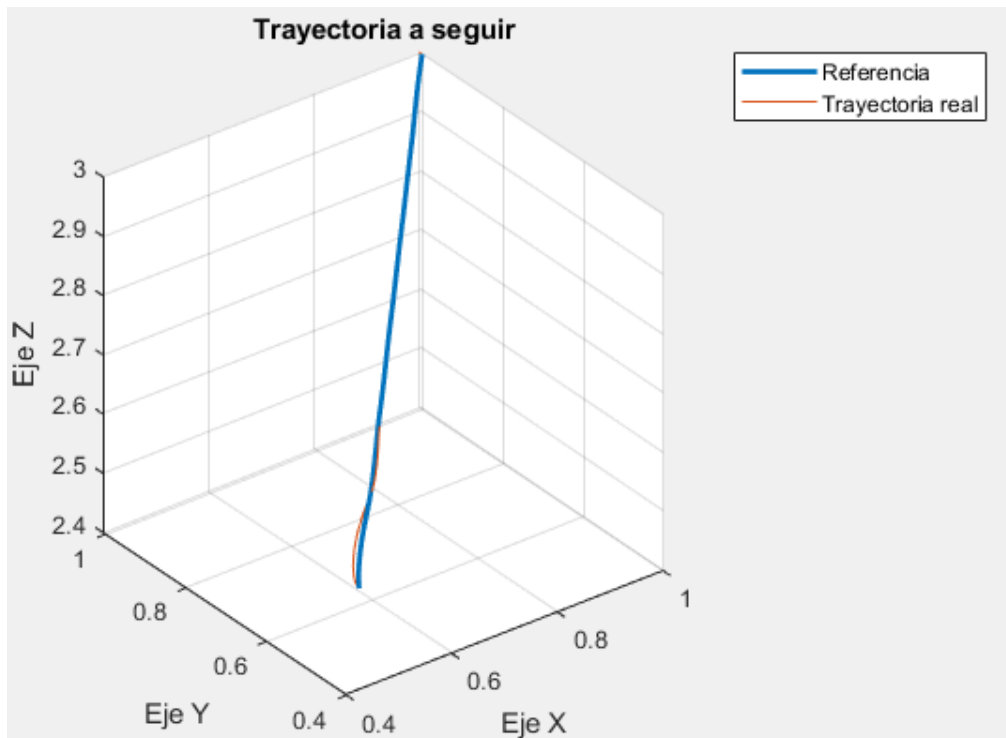
“Simulador_PD_concompensacion_Robustez.slx”, obteniendo las siguientes gráficas:

-Evolución de las coordenadas articulares:

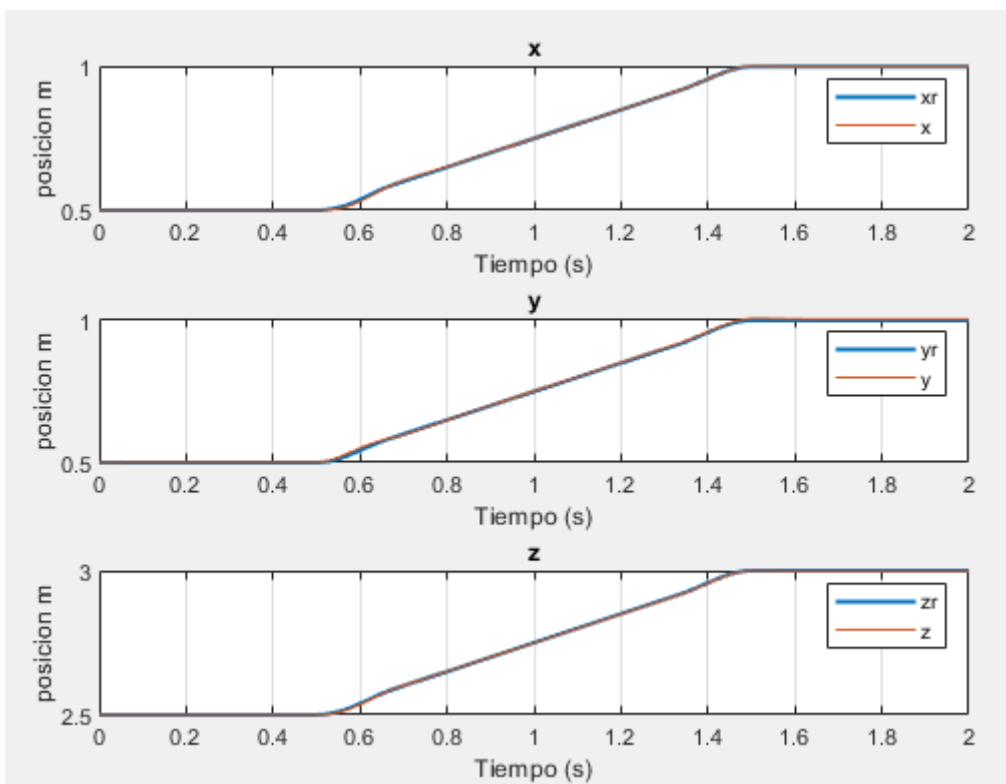




-Trayectoria a seguir:



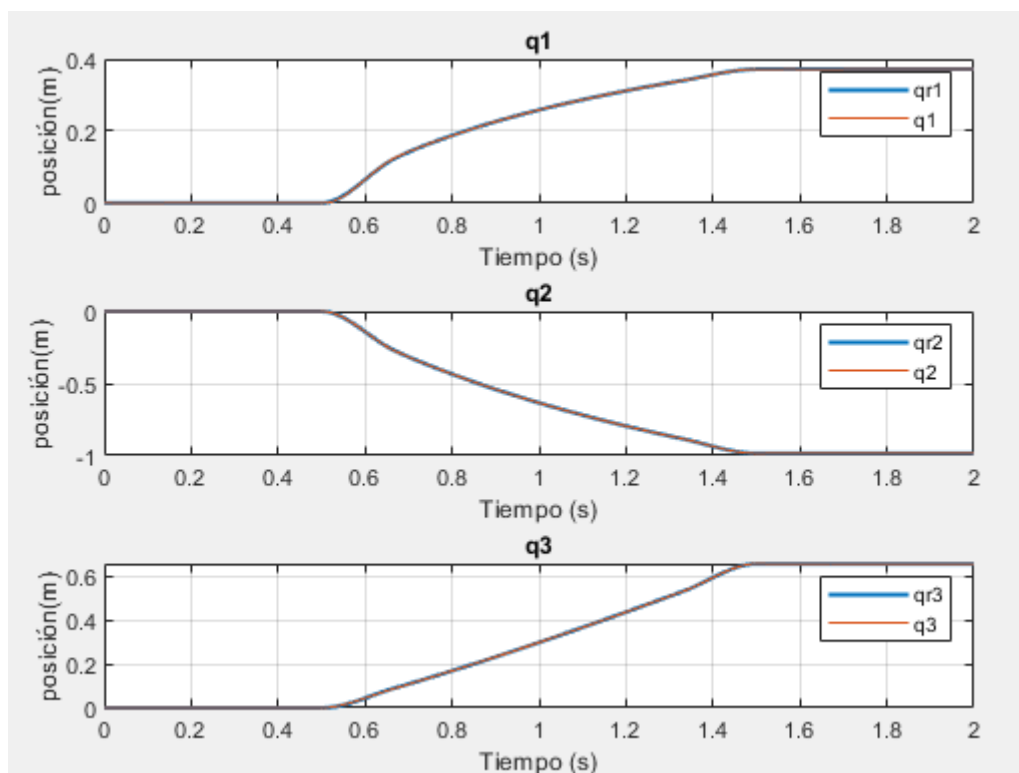
-Trayectoria en función del tiempo:

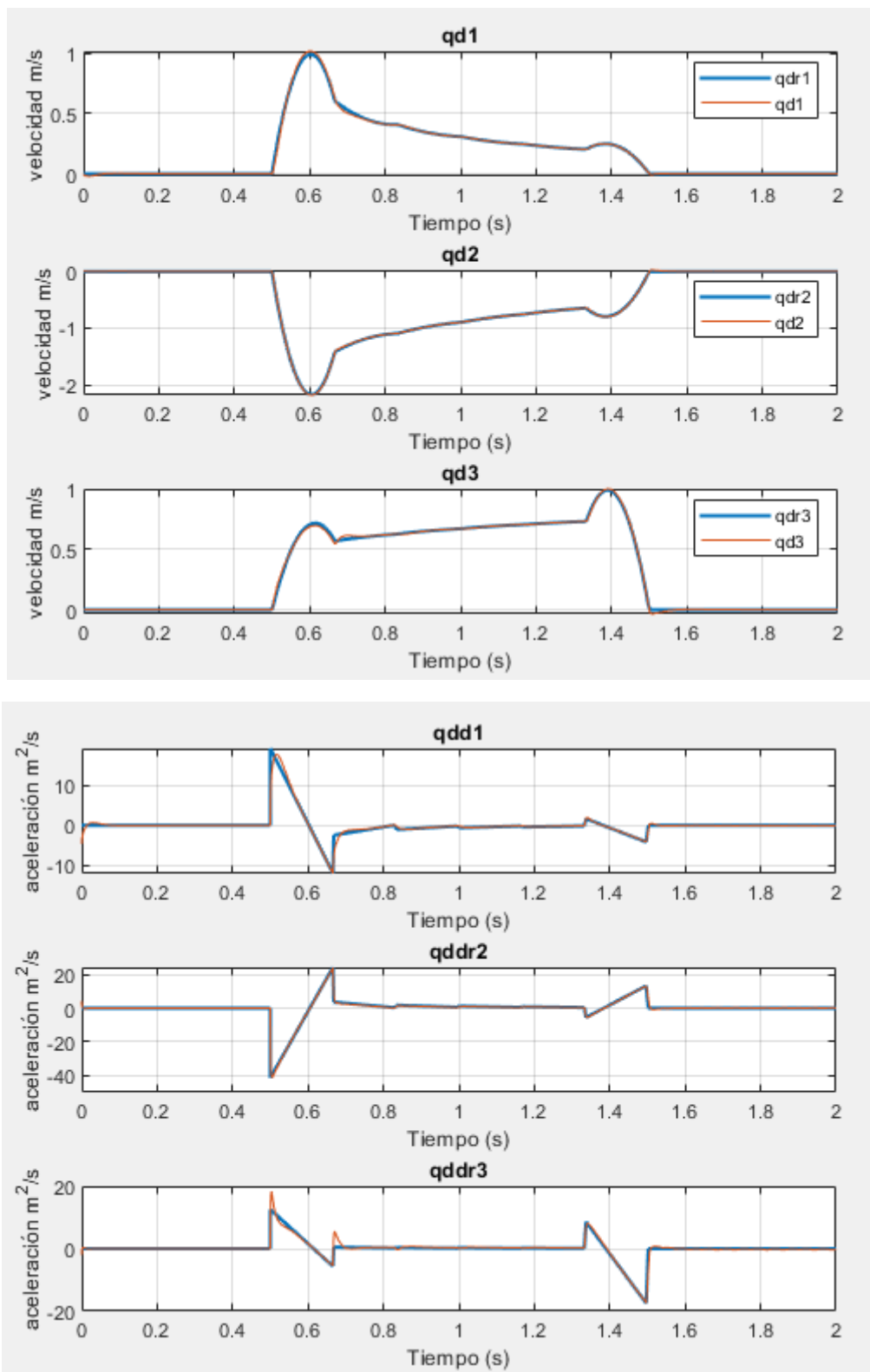


5.3.3 -PID descentralizado

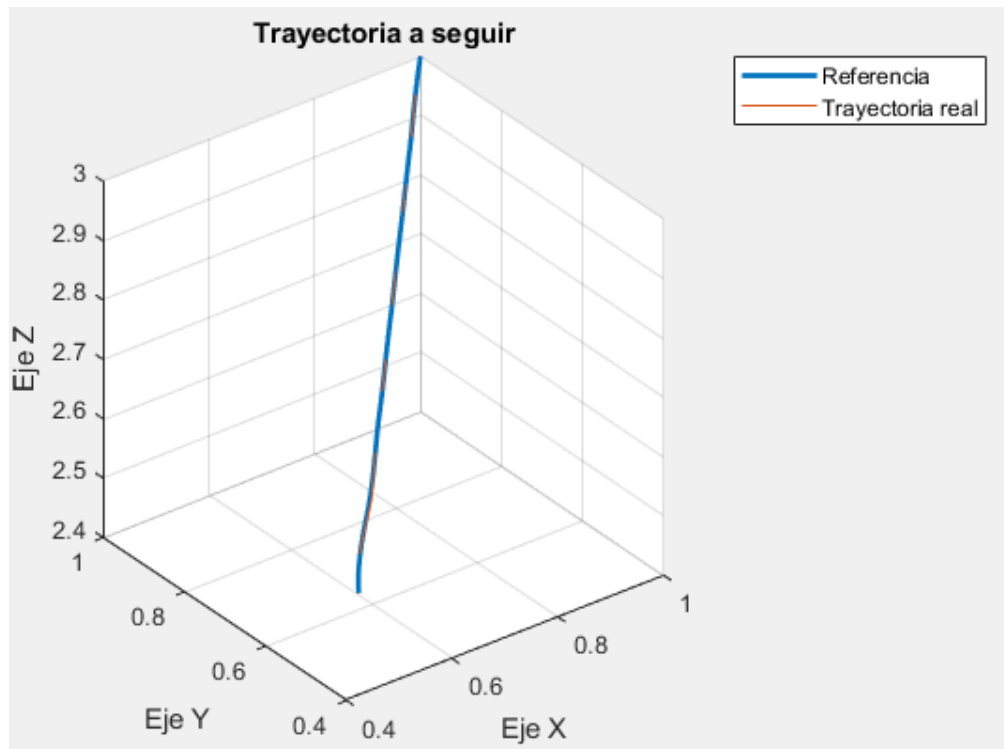
Para este controlador se facilita la simulación “Simulador_PID_Robustez.slx”, obteniendo las siguientes gráficas:

-Evolución de las coordenadas articulares:

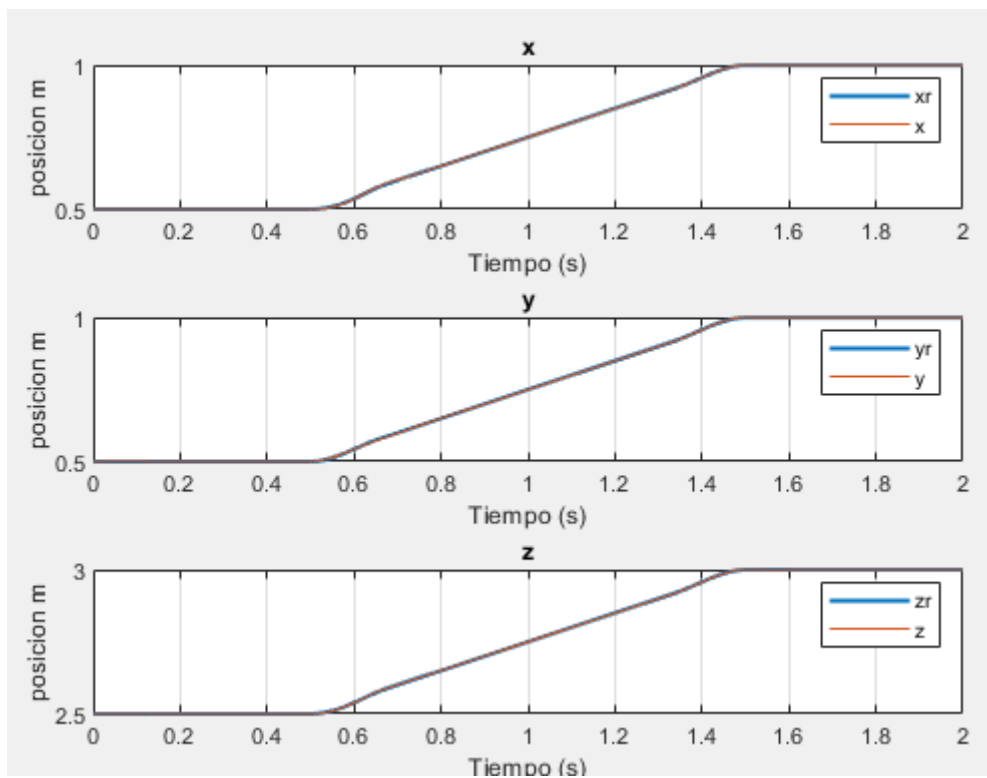




-Trayectoria a seguir:



-Trayectoria en función del tiempo:

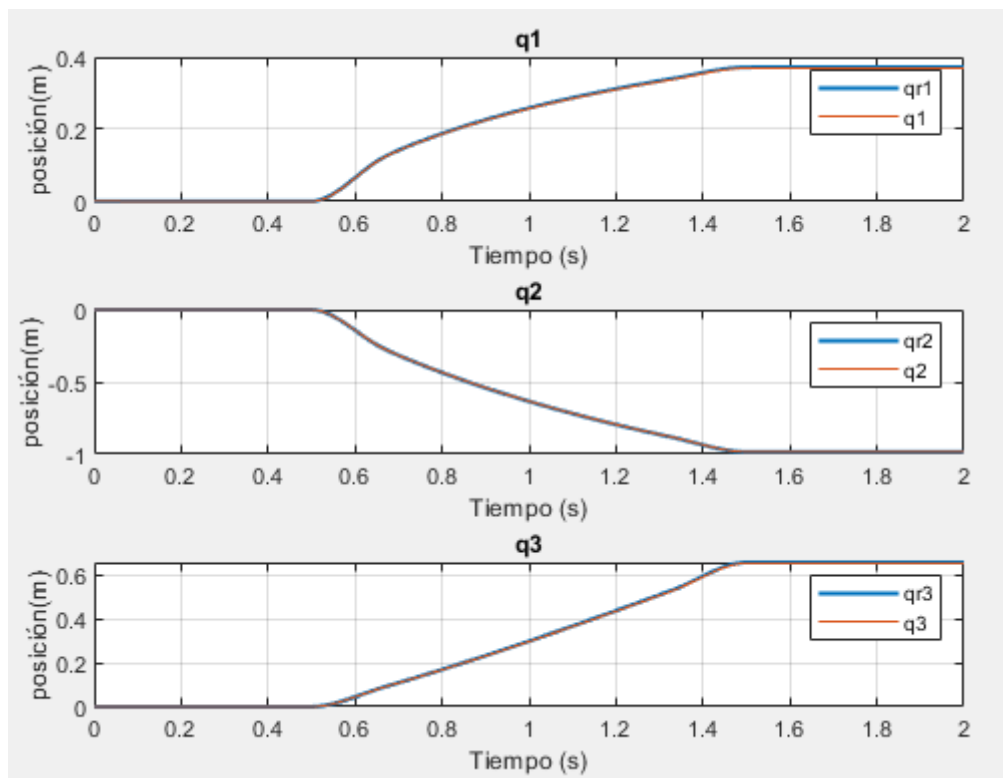


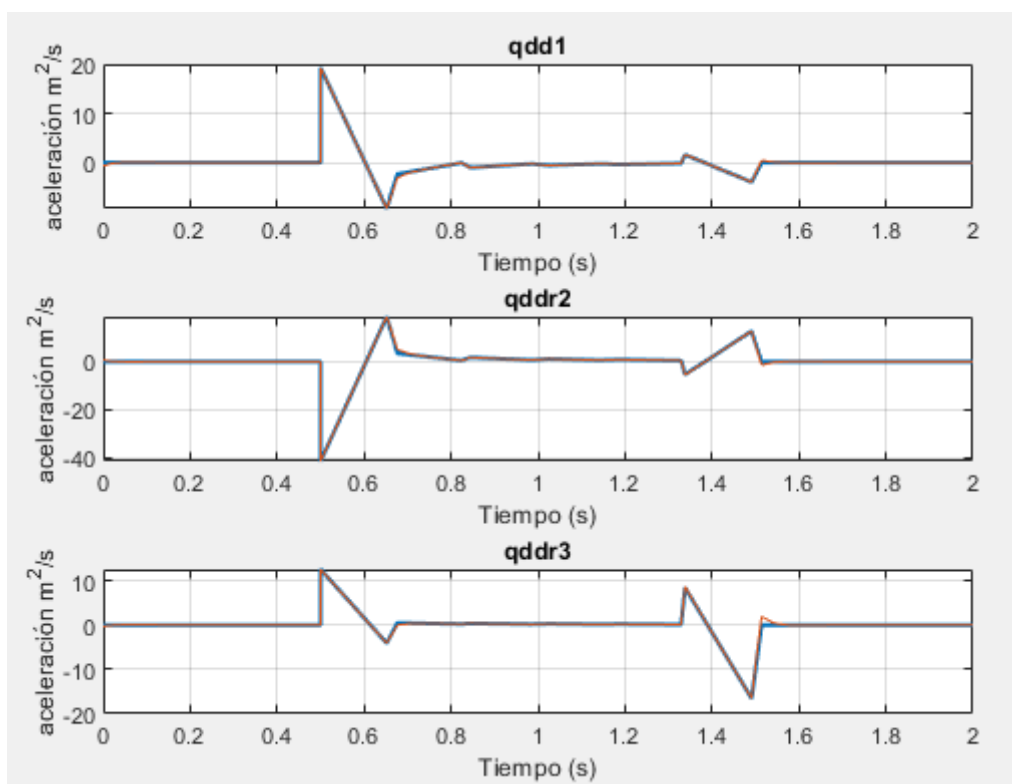
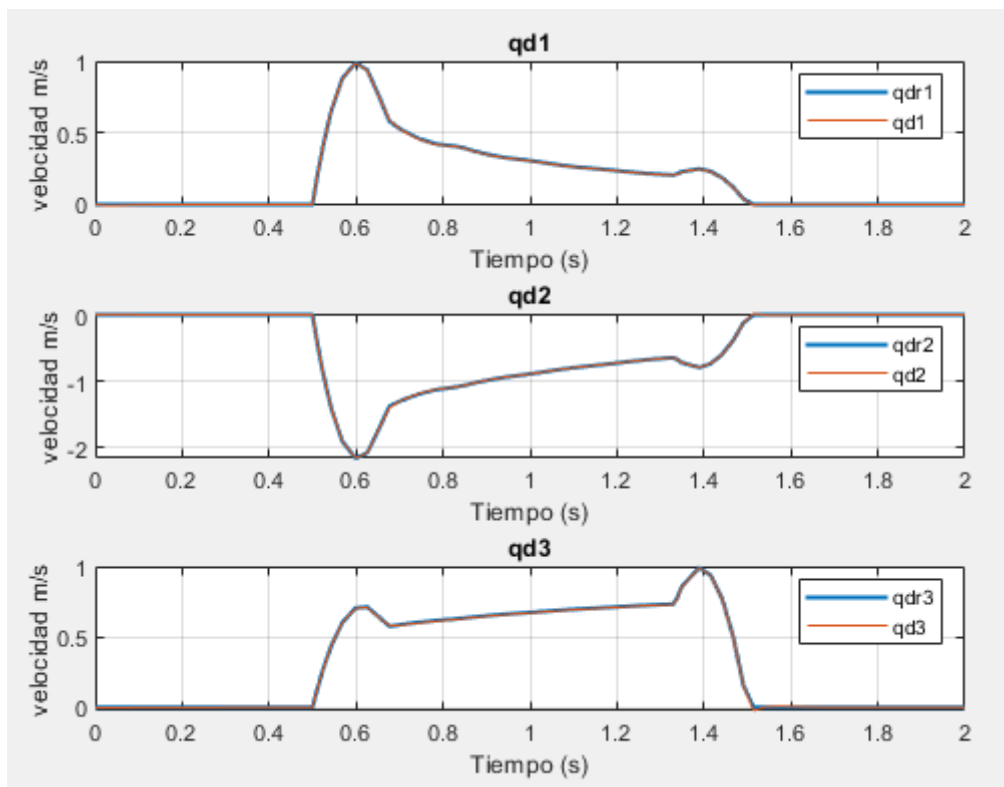
5.3.4 –Controlador de Par Calculado

Para este controlador se facilita la simulación

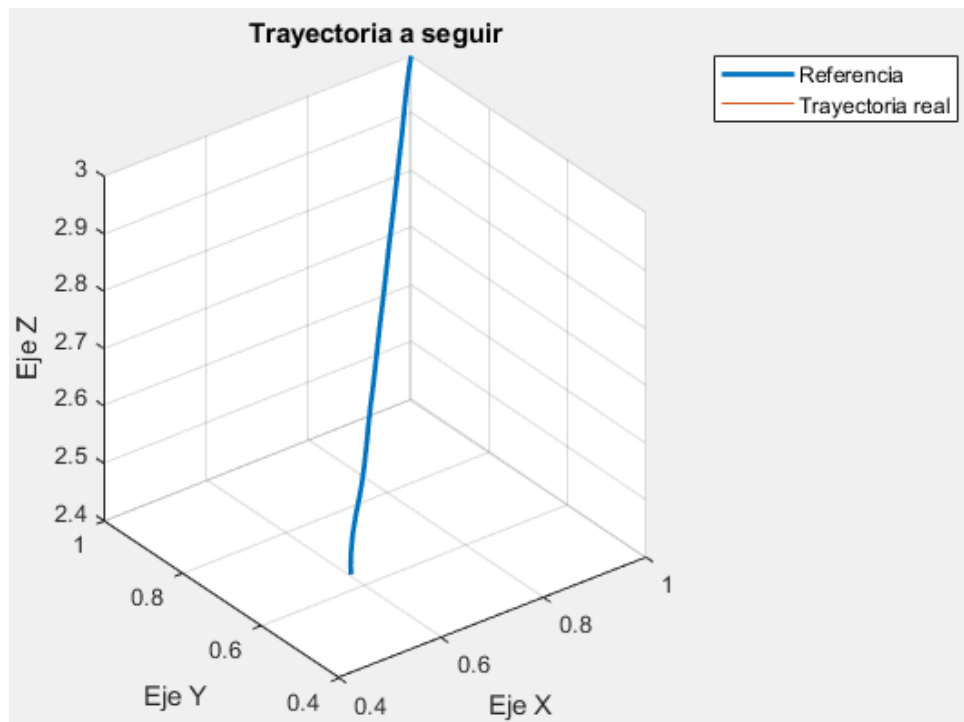
“Simulador_Control_Par_calculado_Robustez.slx”, obteniendo las siguientes gráficas:

-Evolución de las coordenadas articulares:

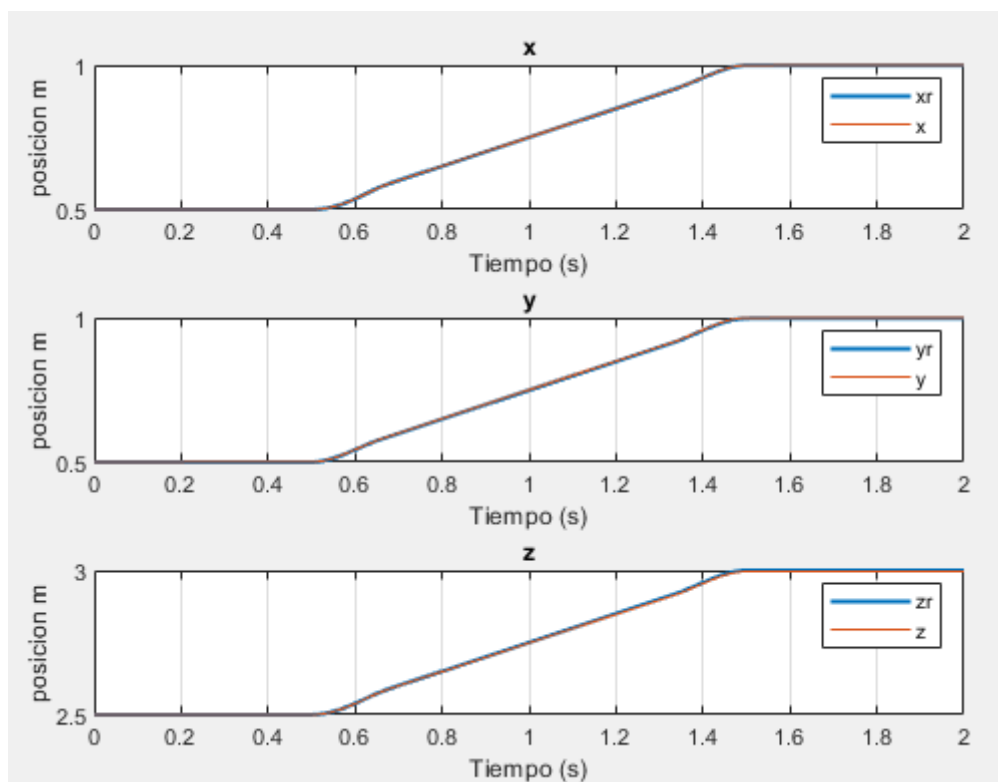




-Trayectoria a seguir:



-Trayectoria en función del tiempo:



6 – Verificación

Una vez verificado todos los resultados, al llamar a la función “verificador.p” se obtiene:

```
Comprobando ficheros de verificación
-----

- Fichero tablaDH.m -----> OK
- Fichero matricesTH.m -----> OK
- Fichero CinematicaDirectaSimbolica.m --> OK
- Fichero CinematicaDirecta.m -----> OK
- Fichero CinematicaInversaSimbolica.m --> OK
- Fichero CinematicaInversa.m -----> OK
- Fichero trayectoriaCircular.m -----> OK
- Fichero jacobiano.m -----> OK
- Fichero parametrosDinamicos.m -----> OK
- Fichero ModeloDinamico_R3GDL.m -----> OK

-----

Todos los ficheros de verificación son sintácticamente correctos

Puede subir los ficheros junto con la memoria en pdf en un fichero
comprimido para su evaluación
```

Por tanto, todos los archivos de verificación funcionan correctamente.