

Control y Programación de Robots

TRABAJO DE CONTROL DE ROBOTS MANUPULADORES
ÁLVARO CALVO MATOS, RAÚL ZAHÍNOS MARÍN Y FEDERICO VAZ FERNÁNDEZ
4º GIERM

1. Análisis Cinemático y Dinámico.

El objetivo de esta parte de la asignatura es el de profundizar un poco más en lo que ya se empezó a ver el año pasado en la asignatura de Fundamentos de Robótica donde se simuló el comportamiento de un brazo robótico para su posterior control.

Esta vez se intentarán simular problemas que tendríamos en un robot real tales como medidas imperfectas o ruidosas o desconocimiento inicial de algunos de los parámetros dinámicos del robot, que tendrán que ser estimados para desarrollar un modelo dinámico a partir del cual poder diseñar controles.

1.1. Cinemática directa e inversa.

La obtención de las cinemáticas directa e inversa del robot es materia del curso pasado, así que no se profundizará en la explicación de su obtención.

A partir del esquema del robot, mediante el algoritmo de Denavit-Hartenberg, se obtienen los parámetros homónimos con los que se pueden generar las matrices de transformación homogéneas que trasladan las coordenadas del extremo del robot al sistema de referencia de la base.

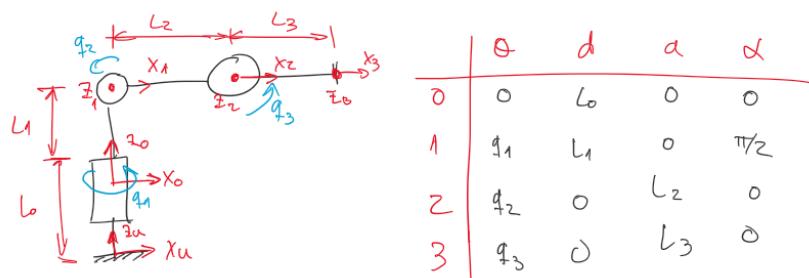


Figura 1. Parámetros de Denavit-Hartenberg.

Con la ayuda del Robotics-Toolbox, se generan las MTH, obteniendo la cinemática directa del robot, implícita en el producto de las 4 matrices de transformación homogéneas.

```
syms L0 L1 L2 L3 q1 q2 q3 real;
PI = sym('pi'); %para que el numero pi sea exacto
AU0 = MDH(0, L0, 0, 0);
A01 = MDH(q1, L1, 0, PI/2);
A12 = MDH(q2, 0, L2, 0);
A23 = MDH(q3, 0, L3, 0);
T = simplify(AU0*A01*A12*A23)
```

El modelo cinemático directo resultante es:

```
x = cos(q1)*(L3*cos(q2 + q3) + L2*cos(q2));
y = sin(q1)*(L3*cos(q2 + q3) + L2*cos(q2));
z = L0 + L1 + L3*sin(q2 + q3) + L2*sin(q2);
```

Para resolver el problema de la cinemática inversa podemos usar también alguno de los procedimientos vistos el curso pasado. Aquí se ha optado por su resolución mediante métodos geométricos.

Se obtiene:

```
q1 = atan2(y,x);  
C3 = (x^2+y^2+(z-L0-L1)^2-L2^2-L3^2) / (2*L2*L3);  
q3 = atan2(sqrt(1-C3^2),C3);  
if q3 < 0  
    q3 = atan2(-1*sqrt(1-C3^2),C3);  
end  
q2 = atan2((z-L0-L1),sqrt(x^2+y^2))-atan2((L3*sin(q3)),(L2+L3*cos(q3)));  
q = [q1 q2 q3]';
```

La función que cumple la sentencia condicional que se observa en el código es la de decidir que con qué configuración alcanzará el robot el punto en cartesianas que le indiquemos.

Sin estas tres líneas de código podemos tener complicaciones en bloques posteriores que requieran de la cinemática inversa tales como el GTCL. El uso de la función “atan2”, en lugar de la más simple “atan” de Matlab también ayuda a la hora de controlar que soluciones se obtienen de la cinemática inversa.

1.2. Ecuaciones dinámicas del robot.

La obtención del modelo dinámico del robot se lleva a cabo de la misma forma que en la asignatura de Fundamentos de Robótica del curso pasado.

Se ha partido del algoritmo de Newton-Euler recursivo que se usó el pasado curso, adaptándolo a la morfología del robot actual y modificándolo para su funcionamiento con variables simbólicas.

Los resultados de aplicar el mencionado algoritmo son las fuerzas y/o pares en cada articulación.

Este año los transformaremos en las intensidades eléctricas que circulan en cada motor multiplicando por la constante de par de los motores y por las reductoras, pasando a trabajar con intensidades eléctricas como señal de control.

El código del que se habla se encuentra en el fichero “GAMMA_REDUCIDA.m”.

Llegados a este punto, dado que los resultados obtenidos dependen de parámetros dinámicos desconocidos del robot, se reescriben las ecuaciones para facilitar la posterior estimación de estos parámetros.

3. Control cinemático.

El objetivo del control cinemático es el de establecer la trayectoria que debe seguir cada articulación para que el extremo del robot describa el movimiento deseado en coordenadas cartesianas, concretando para ello tanto posiciones, como velocidades y aceleraciones articulares en cada instante de tiempo.

De entre las distintas especificaciones de movimiento podríamos mencionar el movimiento punto a punto, en el que el trayecto intermedio nos sería indiferente, movimiento lineal, movimiento circular..., especificaciones de tiempo, velocidad, etc.

Es importante tener en cuenta las limitaciones físicas del robot a la hora de establecer referencias de velocidad o bien saber si la trayectoria pedida es realizable con la configuración del brazo de la que se dispone.

En el bloque de control cinemático se ha introducido un sencillo código que comprueba la accesibilidad de la trayectoria. La configuración del brazo genera un espacio de trabajo comprendido entre dos esferas de radios $R_1 = L_2 - L_3$ y $R_2 = L_2 + L_3$ dentro de los cuales cualquier trayectoria es válida. Dado que el generador de trayectorias que se programará solo genera trayectorias lineales para el extremo del robot, este problema se reduce a comprobar tres condiciones, que, si se cumplen, aseguran que el recorrido es alcanzable físicamente por el brazo robótico con el que se está trabajando.

- El punto inicial debe estar fuera de la esfera de menor tamaño, pero dentro de la mayor.
- El punto final debe estar fuera de la esfera de menor tamaño, pero dentro de la mayor.
- El punto más cercano de la trayectoria al centro de las esferas debe estar dentro del espacio de trabajo. Esto se calcula mediante álgebra vectorial, a través de proyecciones.

```

if (L2-L3)^2 <= (XYZinicio(1)^2+XYZinicio(2)^2+(XYZinicio(3)-L0-L1)^2)
&& (XYZinicio(1)^2+XYZinicio(2)^2+(XYZinicio(3)-L0-L1)^2) <= (L2+L3)^2
    if (L2 - L3)^2 <= (XYZfin(1)^2 + XYZfin(2)^2 + (XYZfin(3) - L0 -
L1)^2) && (XYZfin(1)^2 + XYZfin(2)^2 + (XYZfin(3) - L0 - L1)^2) <= (L2 +
L3)^2
        % si los dos extremos de la recta están en el espacio de trabajo
        % comprobamos si también lo está la recta que los une
        C = [0; 0; L0 + L1]; % Centro de las esferas
        u = (XYZfin - XYZinicio)'/norm(XYZfin - XYZinicio); % Vector
unitario en direccion de la trayectoria
        PiC = C - XYZinicio; % Vector de XYZinicio a C
        Proy = ((u*PiC)/(u*u'))*u; % Proyeccion del vector PiC en direccion de u
        distancia = norm(PiC - Proy'); % Menor distancia de C a la trayectoria

        if distancia >= (L2 - L3)
            % La recta está dentro del espacio de tarea.
            disp('La trayectoria fijada es VALIDA')
        else
            disp('La trayectoria fijada NO es valida. Atraviesa zonas no alcanzables')
        end
    else
        % Error: recta no valida
        disp('La trayectoria fijada NO es valida. El punto final no es alcanzable')
    end
else
    % Error: recta no valida
    disp('La trayectoria fijada NO es valida. El punto inicial no es
alcanzable')

```

Generador de trayectorias cartesianas punto a punto

Lo primero que se pide es desarrollar un generador de trayectorias cuyas únicas especificaciones sean los puntos inicial y final, siendo válida cualquier tipo de trayectoria que pueda realizar el extremo del brazo para alcanzar el punto final desde la posición inicial.

Este apartado se presenta como paso previo o simplificación del objetivo real de esta parte del proyecto: desarrollar un generador de trayectoria cartesiano lineal para el robot de tres grados de libertad (GTCLR3GDL).

A pesar de ello, hemos desarrollado desde un inicio el generador de trayectorias cartesiano lineal, ya que no considerábamos que supusiera mucha dificultad. Esto no significa sin embargo que no se haya obtenido un generador de trayectorias punto a punto, ya que basta con especificar un número de puntos intermedios igual a cero, bien desde fuera, o bien fijando la condición desde el interior del propio código, para obtener este generador de trayectorias a partir de el de mayor complejidad.

En la siguiente figura se demuestra que se posee la capacidad de generar trayectorias punto a punto.

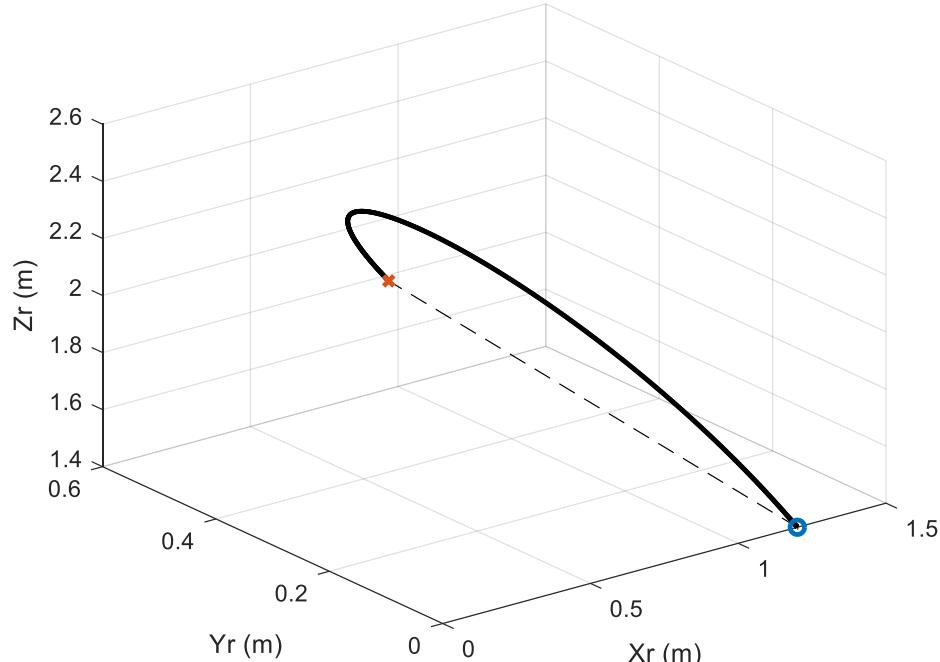
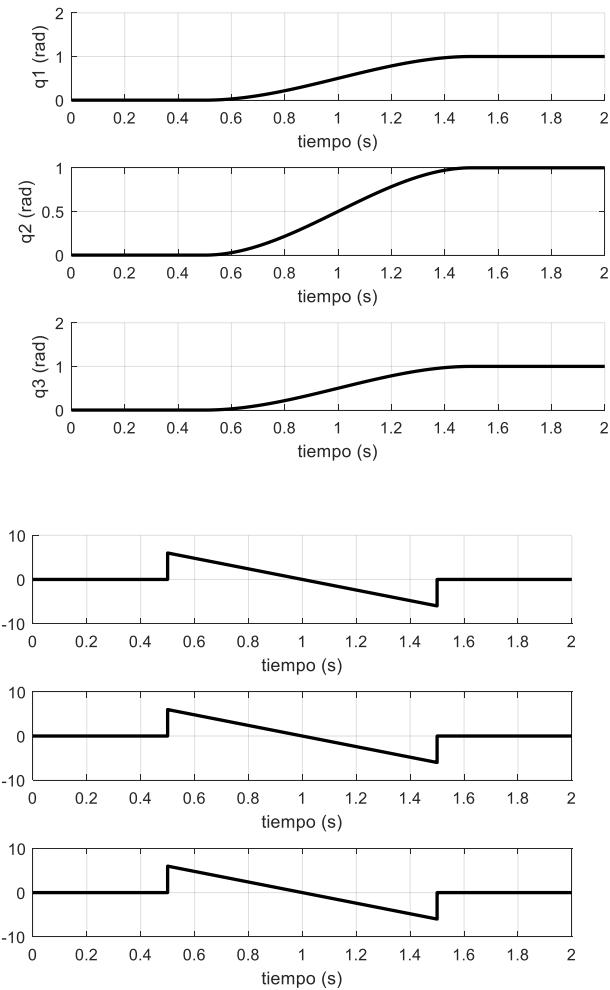
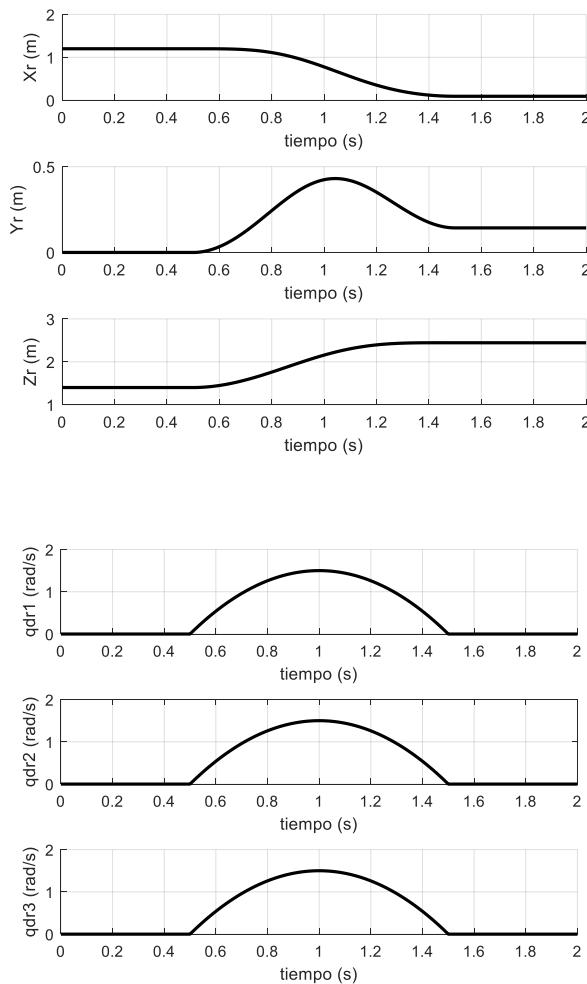


Figura 2. Trayectoria punto a punto generada.

Las referencias de posiciones, velocidades y aceleraciones articulares en el tiempo para recorrer esta trayectoria:



Generador de trayectorias cartesianas lineal.

El funcionamiento del generador de trayectorias cartesiano lineal al que se ha llegado tras el desarrollo puede dividirse en X partes bien diferenciadas:

- Estudio de la trazabilidad de la trayectoria especificada: Se realiza tal y como se ha descrito en la introducción de este apartado, comprobando que la trayectoria cumpla las tres condiciones.
- Generación y muestreo de la ecuación que describe la trayectoria en coordenadas cartesianas respecto al tiempo a partir de los puntos inicial y final. Primero se obtiene la ecuación de la recta en cartesianas, teniendo en cuenta el tiempo de inicio y la duración del movimiento; y luego se muestrea la ecuación obtenida, extrayendo el número de puntos intermedios en cartesianas especificados N.
- Conversión de los N puntos intermedios y los dos extremos a coordenadas articulares por medio de la Cinemática Inversa. Se toman también los tiempos de inicio de cada tramo para uso futuro.

Estos dos últimos puntos han pasado por un proceso de optimización, fusionándose y resultando en un menor número de operaciones cuyo fragmento de código es el siguiente.

```

q = zeros(3, n + 2);
t_tramos = zeros(1, n + 2);
pendiente = (XYZfin - XYZinicio)/duracion;
for punto = 0:(n+1)
    aux = (pendiente*T*punto + XYZinicio);
    q(:, punto+1) = CinematicaInversa(aux);
    % Calculamos los tiempos a los que comienza cada tramo
    t_tramos(punto+1) = inicio + T * punto;
end

```

- Aplicación del método heurístico para obtener las velocidades de paso por cada punto. Se deciden las referencias en velocidad para cada uno de los N puntos intermedios observando el signo de la velocidad en los tramos anterior y posterior a cada uno. De esta forma, si los signos de las velocidades en los tramos coinciden, se establece la velocidad de paso por dicho punto como la media de las dos. Si por el contrario las velocidades tienen signos contrarios, se fija la velocidad de paso a cero. Las velocidades de los puntos inicial y final también se fijan a cero.

```

qd = zeros(3, n + 2);
qd(:, 1) = [0 0 0]';
qd(:, n + 2) = [0 0 0]';
for i = 2:(n + 1)
    if (sign(q(:,i)) - sign(q(:,i-1))) ~= sign(q(:,i+1)) - sign(q(:,i)))
        qd(:,i) = [0 0 0]';
    else
        qd(:,i) = [((q(:,i+1)-q(:,i))/T)+(q(:,i)-q(:,i-1))/T)/2];
    end
end

```

- Interpolación de cada par de puntos con un polinomio de orden tres. Se calculan y almacenan los coeficientes que definen los polinomios de cada tramo para que no se necesite realizar más de una sola vez estos cálculos.

```

a = zeros(3, n + 1);
b = zeros(3, n + 1);
c = zeros(3, n + 1);
d = zeros(3, n + 1);
for i = 1:(n+1)
    a(:, i) = q(:,i);
    b(:, i) = qd(:,i);
    c(:, i) = 3/T^2*(q(:,i+1)-q(:,i))-1/T*(qd(:,i+1)+2*qd(:,i));
    d(:, i) = -2/T^3*(q(:,i+1)-q(:,i))+1/T^2*(qd(:,i+1)+qd(:,i));
end

```

- Identificación del tramo desde el que se ha llamado a la función. Es necesario conocer el tramo actual para poder evaluar el polinomio correspondiente y devolver las referencias adecuadas. Este, junto con los posteriores cálculos para evaluar el tiempo actual en el tramo correspondiente, serán los

únicos cálculos a realizar en sucesiones llamadas al programa para obtener la siguiente referencia en posición, velocidad y aceleración articulares.

```
% Calculos a realizar una vez por llamada a la función:
% Comprobamos en que tramo ha sido llamada la función
if t <= inicio
    % t < tiempo de inicio (reposo inicial) --> ref = q_inicio
    trayectoria = [q(:, 1); [0 0 0]'; [0 0 0]'];
elseif t >= inicio + duracion
    % t > tiempo de inicio + duracion (reposo final) --> ref = q_fin
    trayectoria = [q(:, n + 2); [0 0 0]'; [0 0 0]'];
else
    % Hay que detectar en que tramo nos encontramos
    for i = 1:(n+1)
        if t_tramos(i) <= t && t < t_tramos(i+1)
            tramo = i;
            break
        end
    end
    % Evaluamos la ecuacion para el tramo y tiempo actual
    % q = a + b*(t - ti) + c*(t - ti)^2 + d*(t - ti)^3
    % qd = b + c*(2*t - 2*ti) + 3*d*(t - ti)^2
    % qdd = 2*c + 3*d*(2*t - 2*ti)
    q_t = a(:,tramo)+b(:,tramo)*(t-t_tramos(tramo))+c(:,tramo)*
          (t-t_tramos(tramo))^2+d(:,tramo)*(t-t_tramos(tramo))^3;
    qd_t = b(:,tramo)+c(:,tramo)*(2*t-2*t_tramos(tramo))+3*d(:,tramo)*
          (t-t_tramos(tramo))^2;
    qdd_t = 2*c(:,tramo)+3*d(:,tramo)*(2*t-2*t_tramos(tramo));
    trayectoria = [q_t; qd_t; qdd_t];
end
```

Hay diversas formas de mejorar este algoritmo para hacerlo más eficiente, algunos de estos métodos son el uso de memoria persistente, mencionado implícitamente en la descripción del algoritmo, o la inclusión de la cinemática inversa en el código para así prescindir de llamadas a funciones externas que ralenticen la ejecución.

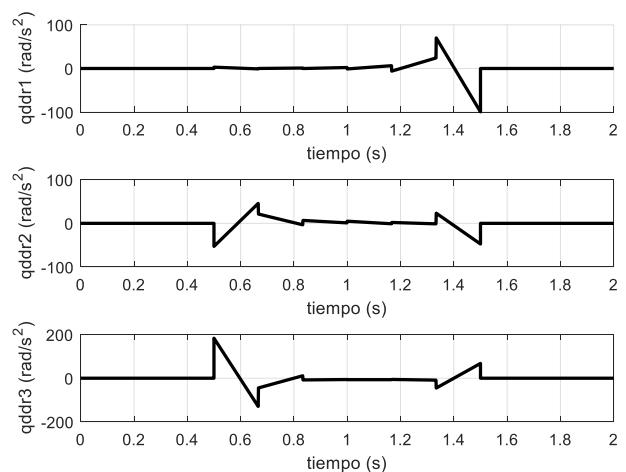
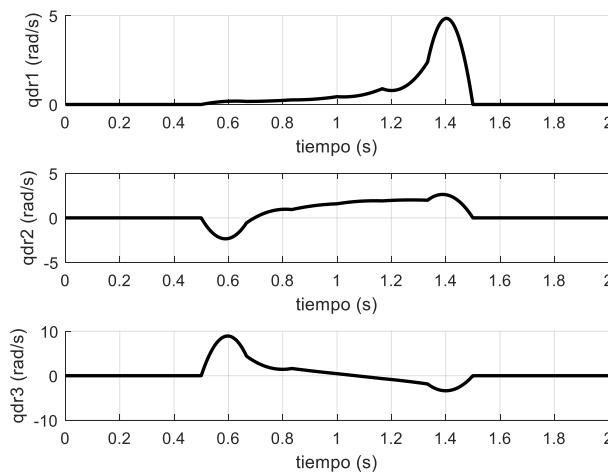
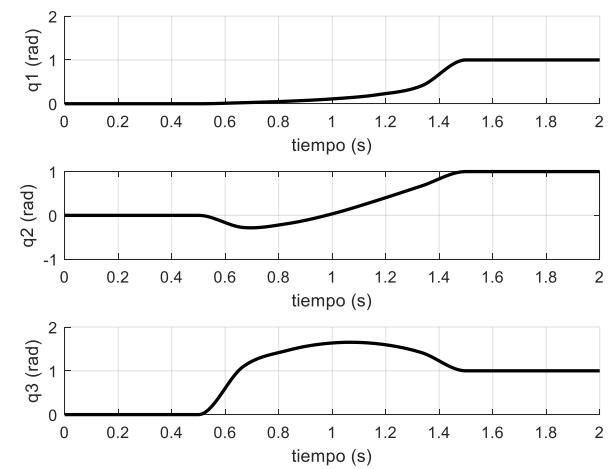
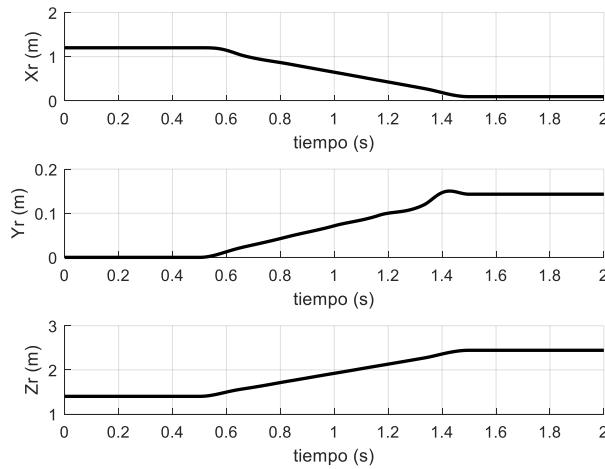
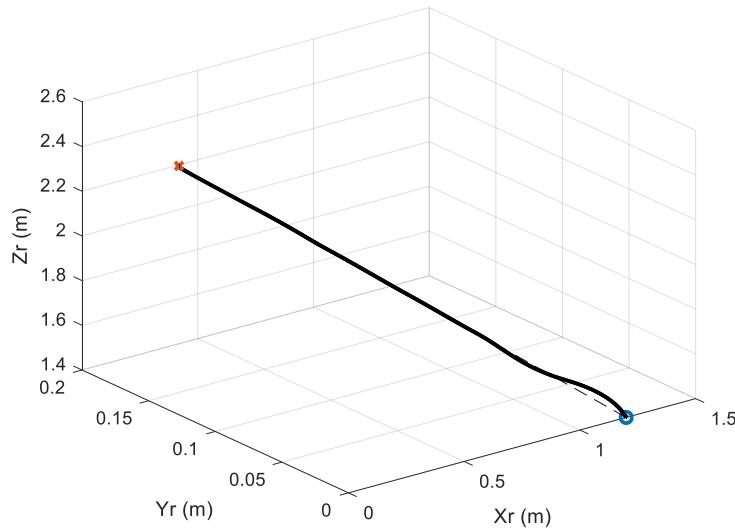
Cabe recalcar que la implementación de este mismo algoritmo en algún lenguaje de programación compilado, y por tanto ejecutable a nivel de procesador, será en principio mucho más rápido que otro implementado en un script de Matlab.

El código completo del generador de trayectorias se encuentra en el archivo adjunto de nombre y extensión "GTCL_R3GDL.m".

Las gráficas que se obtienen como resultado de ejecutar el presente código con unas especificaciones de:

Tiempo de simulación	2 s
Punto inicial em cartesianas (XYZinicial)	CinematicaDirecta([0 0 0])
Punto final en cartesianas (XYZfinal)	CinematicaDirecta([1 1 1])
Número de puntos intermedios (N)	5
Tiempo de inicio del movimiento (t_ini)	0.5 s
Tiempo de finalización del movimiento (t_fin)	1 s

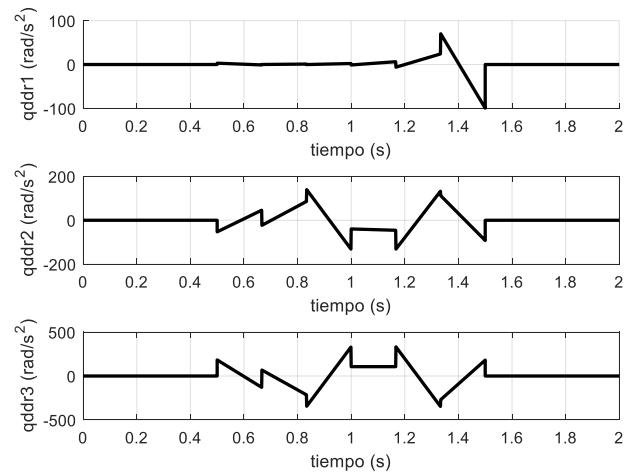
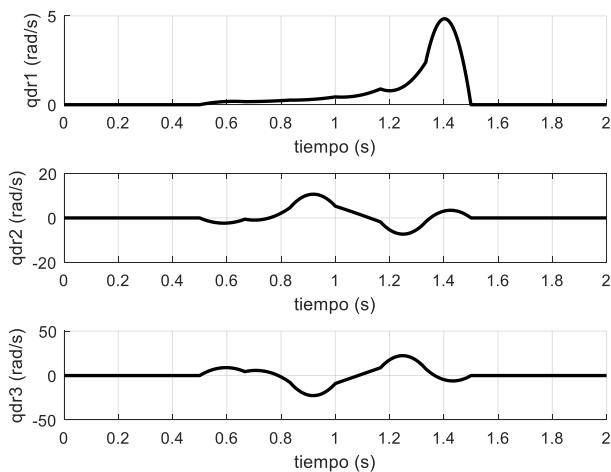
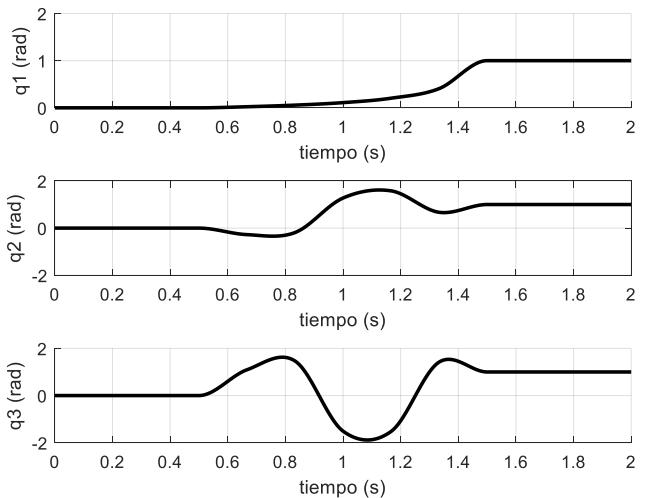
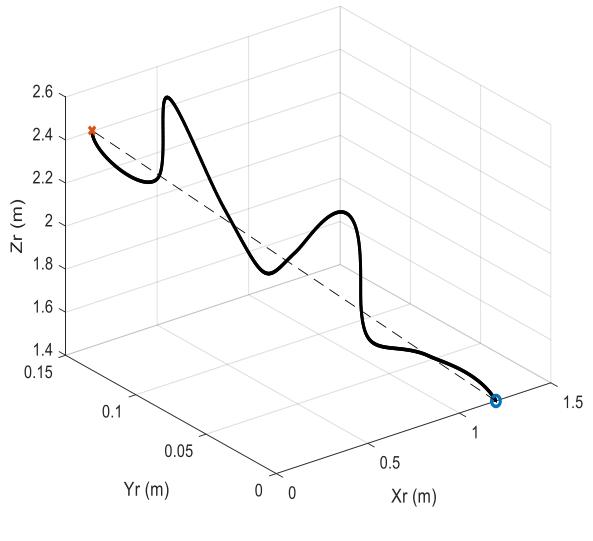
Son:



Aumentando el número de puntos de la trayectoria se pueden lograr mejores resultados en cuanto a posición, pero por el contrario se podrían necesitar unas referencias de aceleración o velocidad muy exigentes para las que necesitaríamos un modelo y control muy buenos.

Durante la prueba de clase se presenció un comportamiento anómalo del generador de trayectorias. Este comportamiento parecía no deberse al propio generador de trayectorias, ya que solo parecía ocurrir con determinadas trayectorias. El problema fue atribuido pues a esa trayectoria en particular.

Tras examinarlo más detenidamente se llegó a la conclusión de que era un error presente en la cinemática inversa, y es que, a pesar de ser correcta la solución obtenida para el problema, no se tomó ninguna decisión respecto a qué configuración tomar. Esto provocaba un cambio de configuración durante la trayectoria que, al ser muestreada y reconstruida, no se veía claramente como una discontinuidad en las trayectorias articulares.



Esta es una de las complicaciones a las que se hace referencia en el primer punto de la memoria. La solución en cuanto a código podemos observarla ahí.

La idea tras la solución es la de fijar por defecto una de las dos configuraciones posibles, ya que se especifica muy concretamente cuáles deben ser las entradas y salidas del bloque del GTCL. En caso de disponer de flexibilidad, se habría dado la opción a decidir la configuración del brazo. Para ello simplemente se habría cambiado la condición del bucle “if” presente en la cinemática inversa por el valor (0 ó 1) de un parámetro externo.

Conclusiones

- No se debe olvidar que la cinemática inversa presentará más de una solución articular para unas mismas coordenadas cartesianas. Si no se toma ninguna decisión en cuanto a qué configuración usar para el codo, la trayectoria generada por el GTCL podría presentar cambios de configuración, resultando en una trayectoria indeseada o que no cumpla especificaciones. Lo recomendable sería, a falta de una especificación externa sobre la configuración del codo, fijar internamente una de las dos.
- Es importante tener en cuenta las limitaciones físicas del robot, de otra forma, las referencias de velocidad o aceleración fijadas podrían estar fuera del alcance del mismo y por tanto la trayectoria, en ningún caso podrá ser recorrida de la forma deseada. También es importante comprobar que la totalidad de la trayectoria esté comprendida dentro del entorno de trabajo del robot.
- Tras muestrear la trayectoria en coordenadas cartesianas y expresar estos puntos en coordenadas articulares, debemos reconstruirla. Cómo llevemos a cabo esta tarea afecta a las velocidades y las aceleraciones de referencia que debamos seguir durante la trayectoria. Una interpolación lineal de la posición se traduce en saltos en velocidad, solo posibles con aceleraciones infinitas. Con un interpolador polinómico de grado 3 tendremos 4 parámetros para fijar condiciones de contorno, decidiendo los puntos de cada tramo y las velocidades a su paso. Con este no disponemos de parámetros para fijar especificaciones de aceleración, para ello necesitaremos un polinomio de orden superior, aunque aumentaría la complejidad y por tanto el coste computacional del problema. Un interpolador de tipo trapezoidal intenta mantener las velocidades constantes todo lo posible, necesitando acelerar únicamente en cambios de dirección, en el inicio y el fin de la trayectoria. El problema se vuelve algo más difícil de resolver, pero como compensación se obtiene una trayectoria en la que el papel de las aceleraciones pierde mucha importancia, reduciendo consecuentemente los problemas futuros de modelado y control dinámico derivados de ellas.
- Existen varios criterios para estipular las velocidades de paso por cada punto. Con el método de la jacobiana podemos obtener las referencias de velocidad directamente de la expresión cartesiana. Podemos obtener un perfil de velocidades continuo, aunque esto no quita que su derivada sea demasiado alta. Otra posibilidad es buscar unas velocidades para las que tengamos aceleraciones continuas, pero necesitaríamos resolver sistemas de ecuaciones. El criterio finalmente usado para este fin, por su simplicidad respecto a los otros dos propuestos, es el heurístico.
- Los parámetros externos del GTCL como el número de puntos intermedios para muestrear la trayectoria o la duración de movimiento afectan directamente sobre las referencias de velocidad y aceleración. Un número elevado de puntos intermedios, cuya finalidad sea la de conseguir una mejor trayectoria en posición, puede hacer que los cambios en los tramos requieran aceleraciones muy altas imposibles o difíciles de seguir con el mejor de los controles a los que podamos llegar. El efecto de variar la duración del movimiento es evidente, a menor duración, mayor rapidez y por tanto, mayores exigencias para el control cinemático.
- Para mejorar el coste computacional de GTCL, y por tanto, el funcionamiento del simulador aquí implementado, se han tomado diversas medidas ya comentadas como pueden ser el uso de variables

persistentes o la localización de todo el código necesario en un único script, prescindiendo de las llamadas a otros archivos, lo cual consume tiempo y recursos. Todo esto y otras buenas prácticas de programación que no son objeto de esta asignatura influye en la velocidad de simulación del robot, pero sin duda, la mejor solución posible pasa por su compilación en un lenguaje máquina. Esto es un paso más que se ha llevado a cabo para el generador de trayectorias, implementando el algoritmo en lenguaje de programación C y posteriormente dejando que un compilador lo optimice para su ejecución en la máquina encargada de realizar las simulaciones. Tras conseguir que el código compilado se comunicara con Simulink y ejecutara las primeras versiones no terminadas del algoritmo del GTCL, se abandonó el código, centrándonos en el script de Matlab, ya que no se nos exigía velocidad y en este punto se había aprendido ya lo realmente interesante, que es saber implementar este tipo de códigos en Simulink, y no queda duda alguna de que de terminarlo (para lo cual solo bastaría traducir el muy parecido lenguaje de Matlab a C), sería mucho más rápido.

2. Estimación de parámetros dinámicos

Tras aplicar el algoritmo recursivo de Newton-Euler, se dispone de un modelo dinámico del robot que puede escribirse matricialmente de la siguiente forma:

$$\tau = K_t R I_m = M(q)\ddot{q} + V(q, \dot{q}) + G(q)$$

A pesar de que dichas ecuaciones son no lineales con q , \dot{q} y \ddot{q} , pueden escribirse de manera lineal respecto a sus parámetros dinámicos, que son el objeto de la identificación. Por lo tanto, es posible utilizar técnicas de regresión lineal para su estimación; en concreto se hará una estimación por mínimos cuadrados.

En primer lugar, será necesario reescribir el modelo para tener la siguiente estructura:

$$y(I_m) = \varphi(q, \dot{q}, \ddot{q}) \Theta$$

Siendo Θ el vector de 24 parámetros a estimar:

$$\Theta = [I_{xx1} \quad I_{xx2} \quad I_{xx3} \quad I_{yy1} \quad I_{yy2} \quad I_{yy3} \quad I_{zz1} \quad I_{zz2} \quad I_{zz3} \quad J_{m1} \quad J_{m2} \quad J_{m3} \quad B_{m1} \quad B_{m2} \quad B_{m3} \\ m_1 L_1^2 \quad m_2 L_2^2 \quad m_3 L_3^2 \quad m_1 L_1 \quad m_2 L_2 \quad m_3 L_3 \quad m_1 \quad m_2 \quad m_3]^T$$

Nótese que en las ecuaciones de Newton-Euler aparecían las distancias a los centros de masa como parámetros desconocidos (sy1, sx2 y sx3 en NE_3GDL.m). Dado que el sólido rígido rota respecto a uno de sus extremos, se aprecia más claramente el significado físico de los momentos de orden 1 y 2 si se realiza el siguiente cambio de variables:

$$s_i = l_i - L_i$$

Donde l_i es la longitud del eslabón, s_i la distancia al centro de masas desde el origen de los ejes del eslabón y L_i la distancia del centro de masas al eje de giro.

De esta forma, los parámetros $m_i L_i^2$ representan la inercia adicional que aparece al girar en torno a un eje paralelo a una cierta distancia del centro de masas.

Utilizando el Symbolic Toolbox se obtuvo la matriz $\varphi(q, \dot{q}, \ddot{q})$, de dimensiones 3 x 24. Sin embargo, no es apropiada para la estimación por mínimos cuadrados, ya que existen tanto columnas nulas como linealmente dependientes, lo cual imposibilita el cálculo de la pseudoinversa. Esto implica que existen parámetros no identificables (no aparecen en las ecuaciones de movimiento) y parcialmente identificables (solo pueden obtenerse en combinación lineal con otros).

En su lugar, se obtuvo una base de parámetros de dimensión mínima. Para ello, se hizo uso de la función *rref*, la cual proporciona la forma escalonada reducida de la matriz y un vector que indica las columnas linealmente independientes. Dado que este procedimiento se implementa en numérico, se construyó una matriz cuadrada encadenando filas con valores aleatorios de q , \dot{q} y \ddot{q} . El archivo Reducción.m ejecuta dichas instrucciones.

Una vez halladas las columnas linealmente independientes, se obtuvieron por inspección los coeficientes de proporcionalidad de las combinaciones lineales y con ellos fue posible obtener la siguiente base reducida:

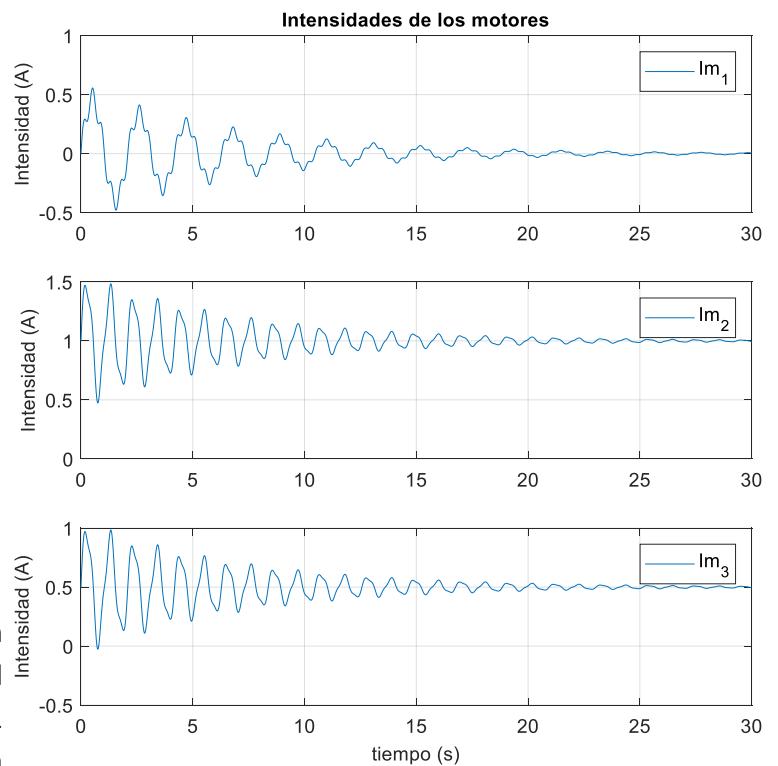
$$\Theta_{\text{red}} = \begin{bmatrix} -m_2 L_2^2 - m_3 l_3^2 + I_{xx2} - I_{zz2} \\ -m_3 L_3^2 + I_{xx3} - I_{yy3} \\ m_2 L_2^2 + m_3 L_3^2 + J_{m1} R_1^2 + m_3 l_2^2 + I_{yy1} + I_{yy3} + I_{yy2} \\ m_2 L_2^2 + J_{m2} R_2^2 + m_3 l_2^2 + I_{zz2} \\ m_3 L_3^2 + I_{zz3} \\ J_{m3} \\ B_{m1} \\ B_{m2} \\ B_{m3} \\ L_2 m_2 + l_2 m_3 \\ L_3 m_3 \end{bmatrix}$$

Por simplicidad no se ha incluido en la memoria la matriz φ_{red} , pero puede ser consultada ejecutando la última sección del fichero NE_R3GDL.m

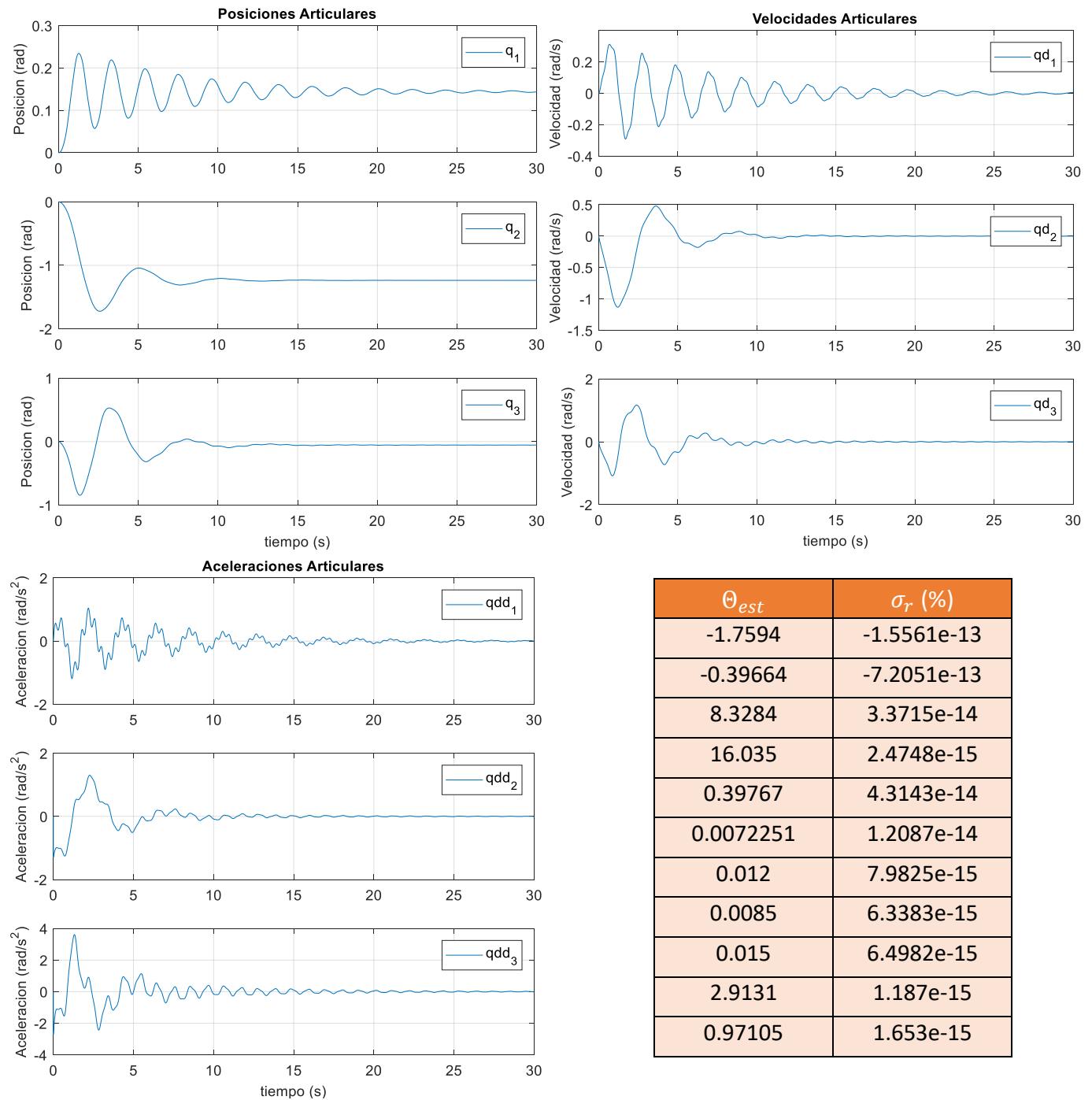
Para realizar los experimentos, se diseñó un escenario en Simulink en el que se excita al robot con señales compuestas por una suma de dos senoides, de diferentes amplitudes y frecuencias, amortiguadas por exponenciales negativas.

De esta manera se pretende provocar grandes aceleraciones al inicio para que aparezcan datos significativos en la estimación de parámetros inerciales, así como un tramo en el que exista poco movimiento para identificar mejor los que intervienen en los términos gravitatorios.

Para que el robot se quedara en una posición estable, la componente continua de la intensidad se eligió nula para la primera articulación, de 1 Amperio para la segunda y de 0.5 Amperios para la tercera. Las oscilaciones se amortiguan en aproximadamente 20 segundos ($\tau = 7$ s). Las amplitudes son moderadas para no volver al sistema inestable y las frecuencias relativamente bajas (~ 15 rad/s) para evitar desfases a la hora de filtrar.



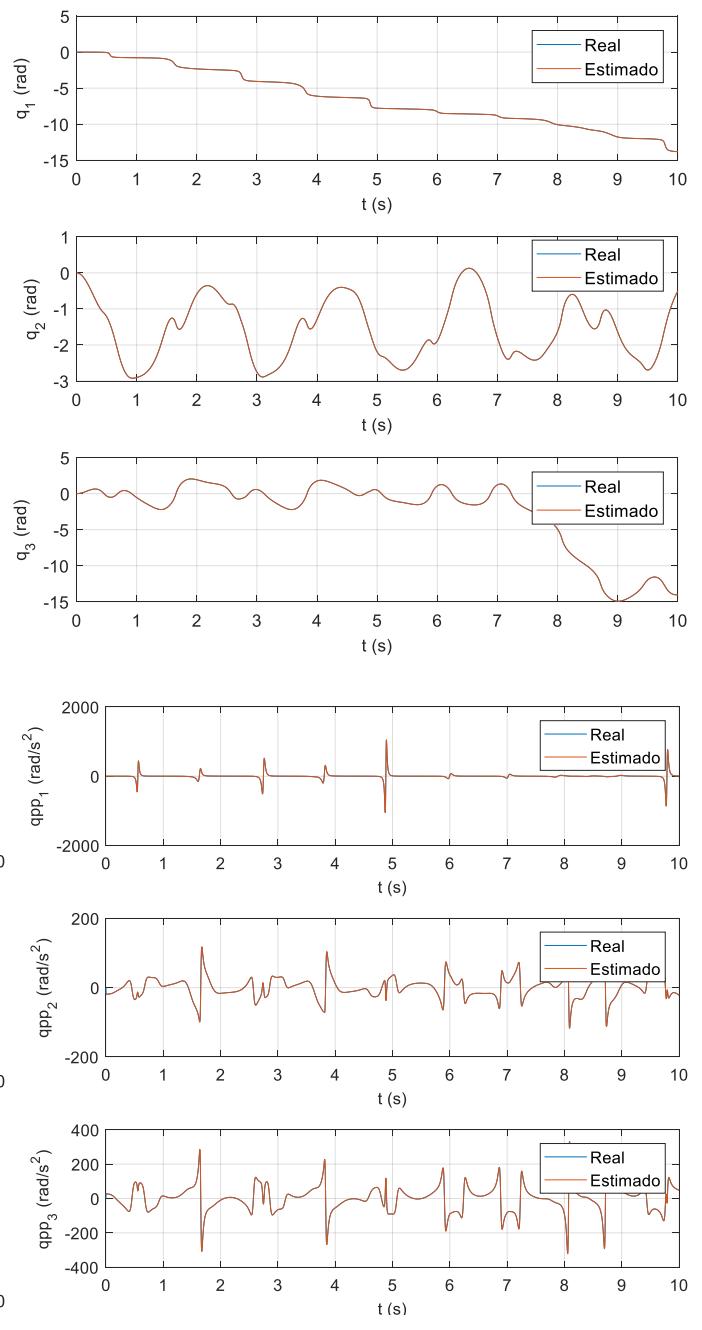
En primer lugar, se realizó la estimación utilizando medidas ideales.



Se recopilaron datos muestreando cada 1 ms, pero para aplicar mínimos cuadrados se utilizaron medidas separadas 10 muestras entre sí, pretendiendo evitar filas prácticamente iguales. Dichos datos generaron una matriz de más de 7000 filas que permitió estimar los parámetros de manera precisa.

De la misma forma, se estimaron para el caso de accionamiento directo y se reconstruyó el modelo dinámico con los valores numéricos obtenidos (`ModeloDinamicoEstimado_GXX.m`). Se observó que para que los resultados no divergiesen era necesario tomar un número elevado de cifras decimales.

Θ_{est} (A.D.)
-1.75940038424998
-0.396642654749998
2.16551925299997
1.77669585024997
0.397667654749993
0.00722506600000024
0.0119999999999998
0.0084999999999985
0.0149999999999998
2.91314677499997
0.971048924999985



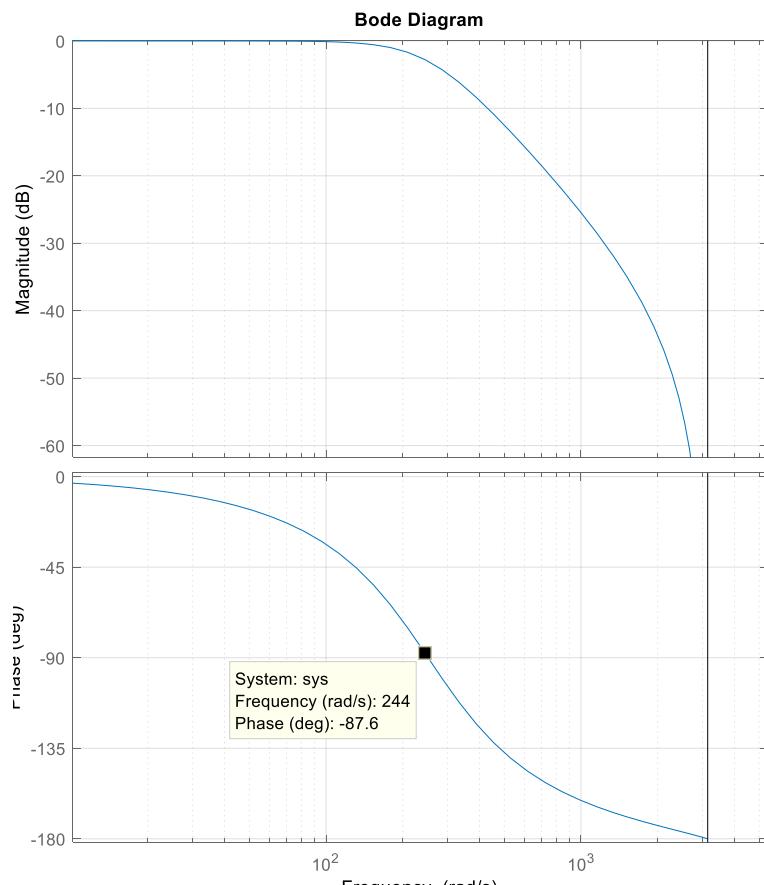
Para el caso de las medidas reales se siguió la misma estrategia, aunque con ciertas particularidades que se comentarán a continuación. En concreto, para simular las no idealidades, se introdujo un cuantizador que imitara el funcionamiento de un encoder de 9 bits. Se supuso que se disponía de una medida de la velocidad obtenida a través de un tacómetro, la cual presenta ruido blanco de magnitud máxima un 2% del rango de medida. En el robot real, ambos sensores irían montados en el lado del eje del motor. Como las medidas de interés son la posición y velocidad de la carga, de esta forma la precisión se incrementa en proporción al factor de reducción. En las simulaciones, las señales disponibles (q y \dot{q}) se corresponden con el lado de la carga, pero para emular lo comentado anteriormente se han dividido por R el paso de cuantización y la magnitud del ruido aditivo.

Las medidas de velocidad fueron filtradas en digital utilizando un filtro Butterworth de segundo orden con frecuencia de corte de 40 Hz.

Aunque el ruido se reduce significativamente, habrá que ser cuidadosos con las señales de excitación, ya que a medida que se aumente la frecuencia aumenta también el desfase. Este hecho puede dar lugar a que se estimen parámetros erróneamente, tales como parámetros inerciales negativos. Para garantizar que el conjunto estimado es coherente, el fichero `M_def_pos.m` reconstruye la matriz M del modelo y comprueba que es definida positiva para diferentes valores de q .

Dado que no es posible medir las aceleraciones, fue necesario estimarlas a partir de la derivada numérica de la velocidad (una vez filtrada). Se utilizó un esquema de diferenciación centrada:

$$\ddot{q} = \frac{\dot{q}(k+1) - \dot{q}(k-1)}{2T}$$



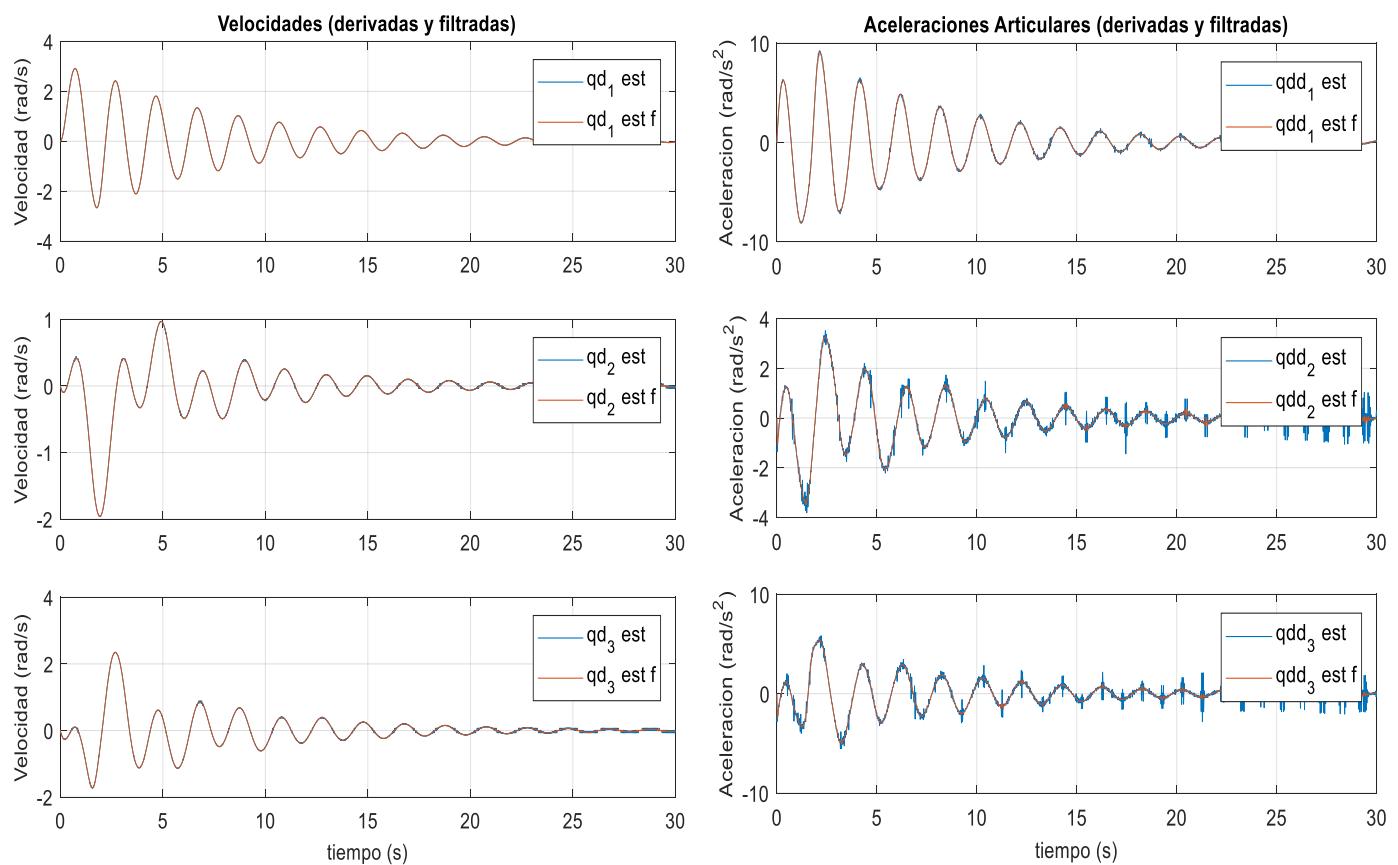
Llegados a este punto, se realizaron experimentos para recopilar datos con los que hacer la estimación, pero las desviaciones típicas resultaban desorbitadas y los parámetros obtenidos no guardaban similitud con los del caso ideal.

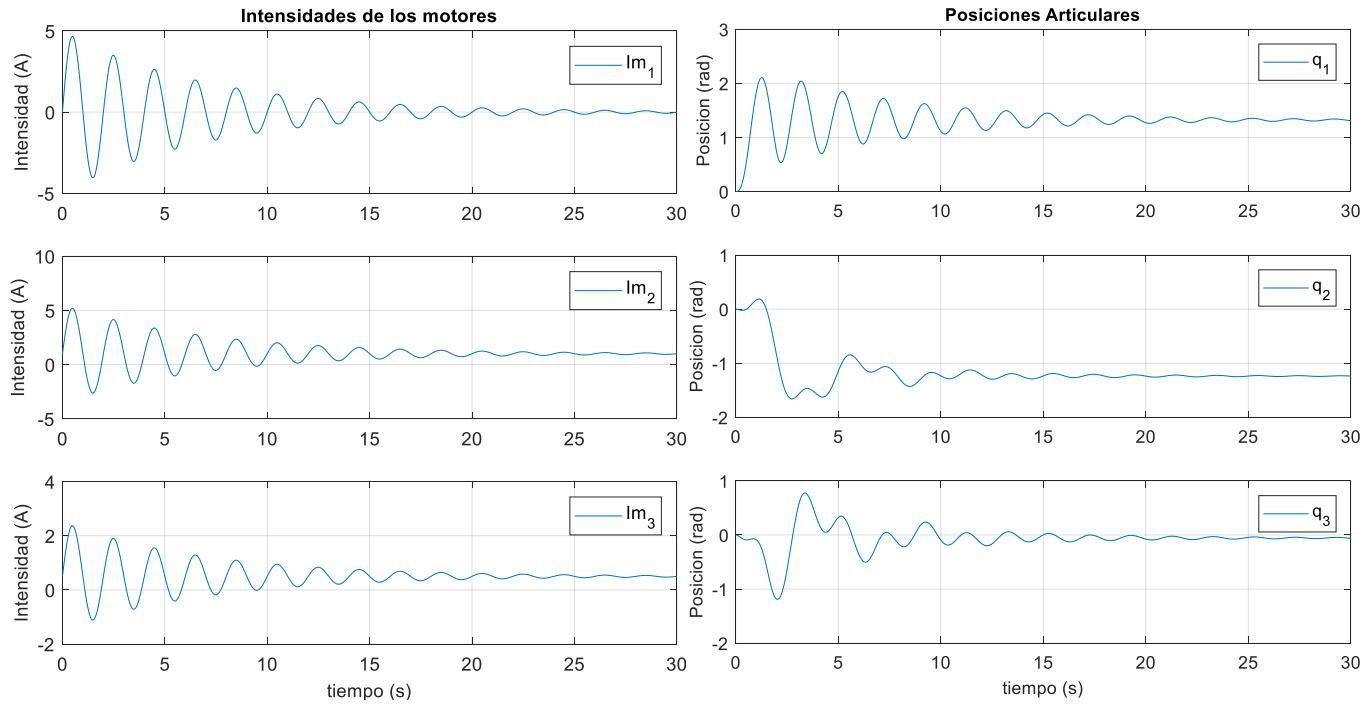
La primera estrategia consistió en repetir los experimentos un elevado número de veces, variando las señales de excitación en cada iteración y guardando los valores de aquellos parámetros que presentaran una desviación típica inferior al 15%. Sin embargo, nos dimos cuenta de que había casos en los que aun teniendo una desviación típica aceptable, el parámetro estimado distaba mucho del parámetro ideal, por lo que decidimos descartar este procedimiento.

El problema que estábamos experimentando se debía a la forma en la cual estábamos obteniendo las aceleraciones. Dado que la señal de la velocidad era muy ruidosa, al aplicar una derivada numérica, con un paso extremadamente pequeño además, las aceleraciones obtenidas eran aún más ruidosas y de valores desmesurados.

Intentamos corregir dicho problema mediante filtrado, tanto de velocidades como de aceleraciones, pero el resultado seguía sin ser demasiado bueno. El motivo por el que sucedía esto tiene que ver con el muestreo de señales. El tiempo de muestreo se fijó en 1 ms, por lo que la frecuencia de Nyquist estaría en 500 Hz; mientras que el bloque de Simulink para emular el ruido se había configurado a $T_s/10$ ms, es decir, a 1kHz. Por este motivo, al muestrear se producía un solapamiento de los espectros y aparecían componentes asociadas al ruido en la zona de baja frecuencia. Debido a esto, el filtro que se estaba utilizando no eliminaba por completo el ruido y las señales de velocidad filtradas no eran adecuadas para ser derivadas numéricamente.

La nueva estrategia consistió en partir de las medidas de la posición para obtener tanto velocidades como aceleraciones. Se aplicó filtrado tanto a la señal a derivar como a la señal obtenida, utilizando la orden `filter` de MATLAB. El resultado en este caso mejoró en lo que a ruido se refiere, pero las señales obtenidas presentaban un cierto retraso respecto a la magnitud real. Sin embargo, como la identificación se realiza fuera de línea es posible realizar una corrección de la distorsión de fase. En concreto, se utilizó la orden `filtfilt` para filtrar con distorsión de fase cero.



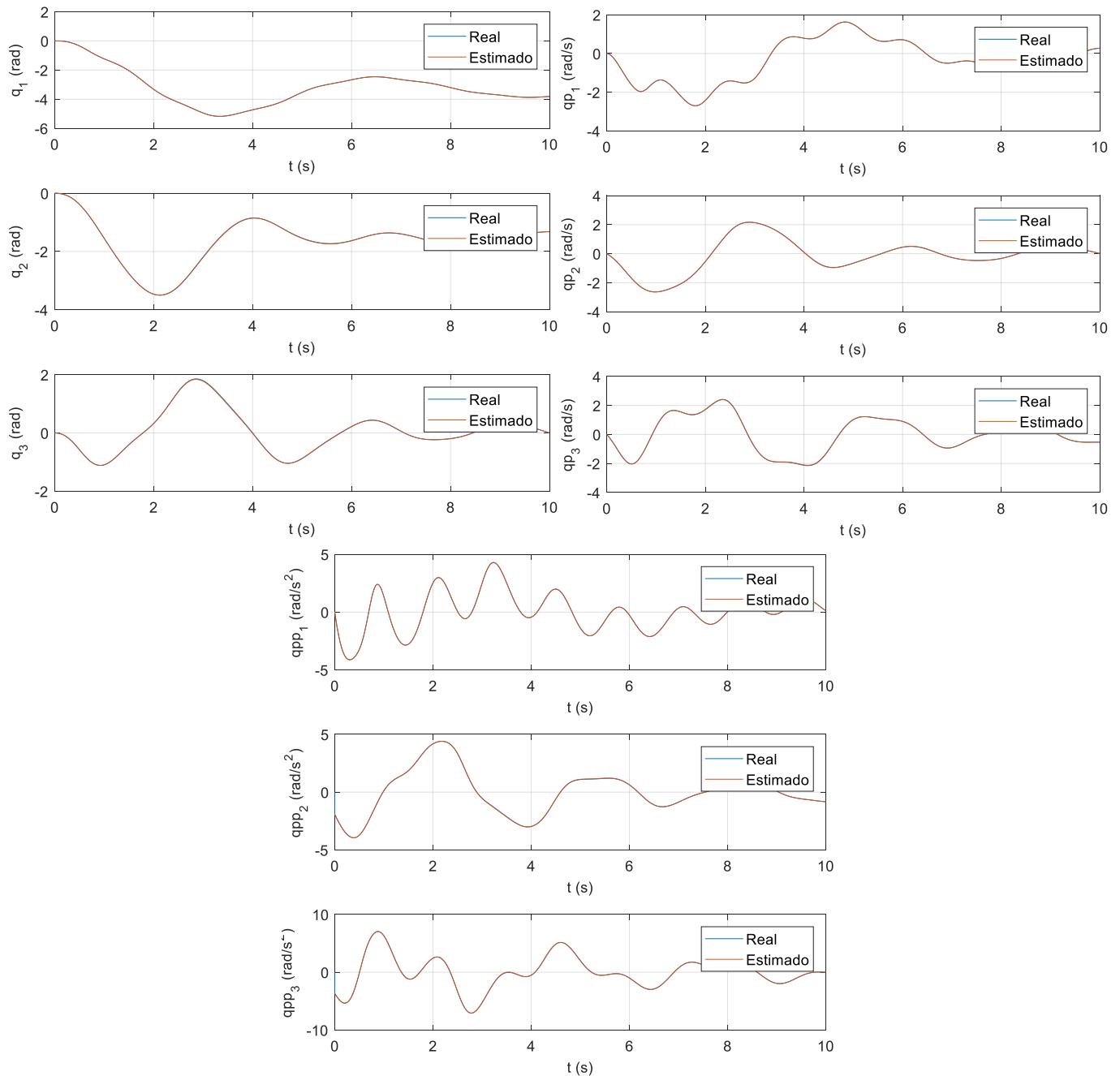


Una vez que se ha tratado adecuadamente las señales, la estimación por mínimos cuadrados proporciona el siguiente resultado:

Θ_{est}	$\sigma_r (\%)$
-1.7648	0.7972
-0.4017	1.9194
8.3358	0.1425
15.9600	0.1154
0.4198	1.8265
0.0071	0.7040
0.0120	0.0681
0.0086	0.3678
0.0149	0.3506
2.9121	0.0863
0.9694	0.1257

$\Theta_{est} (\text{ideal})$
-1.7594
-0.39664
8.3284
16.035
0.39767
0.0072251
0.012
0.0085
0.015
2.9131
0.97105

Tras reconstruir el modelo y simular, se verifica que la estimación es aceptable:



La estimación de parámetros para el caso del robot de accionamiento directo con medidas reales fue especialmente difícil. Inicialmente se siguió la estrategia de variar las señales de excitación hasta dar con los parámetros adecuados, que fueron los siguientes:

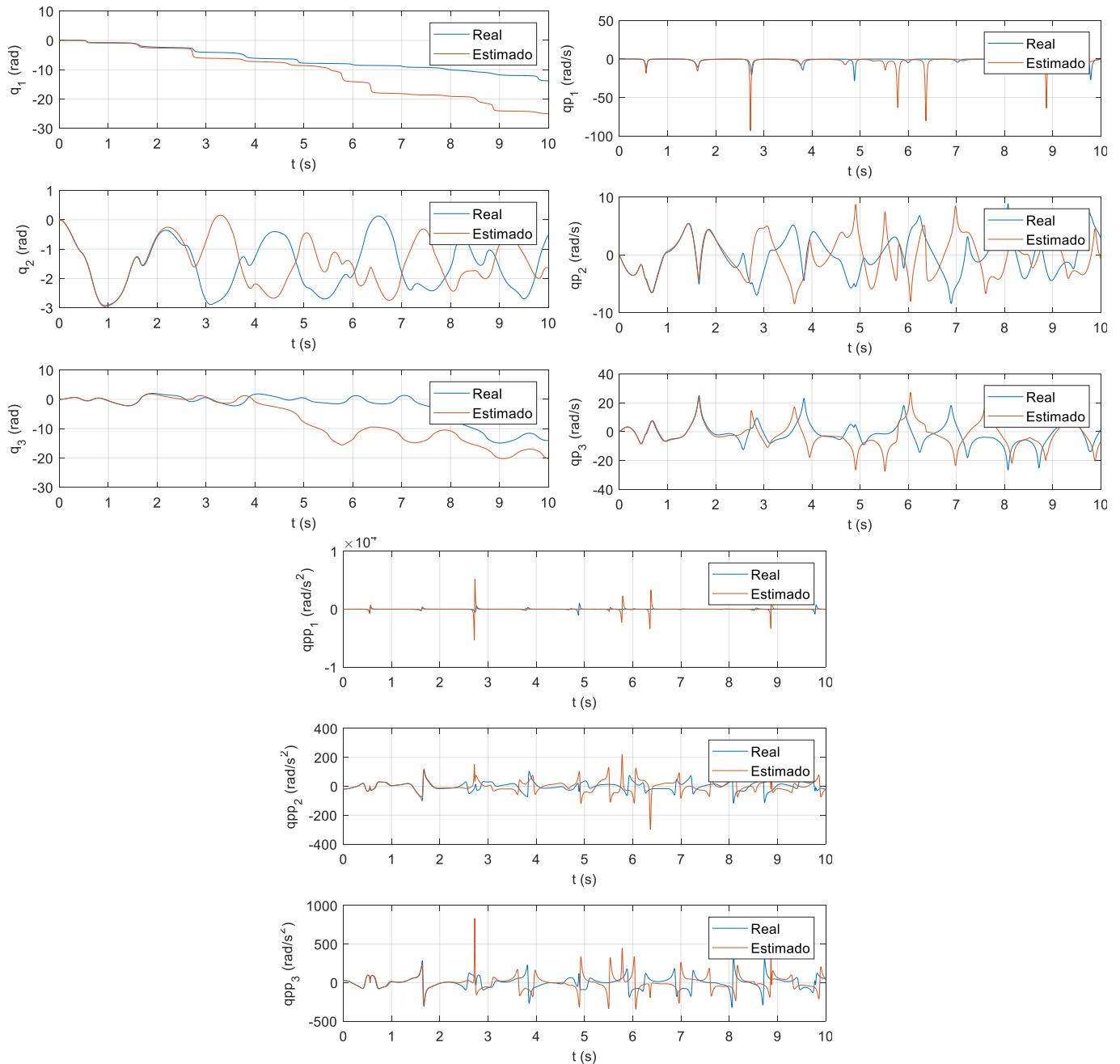
Θ_{est}
-1.7594
-0.3966
2.1575
1.7723
0.3977
0.0072
0.0120
0.0085
0.0150
2.9131
0.9710

Sin embargo, el criterio de la desviación típica no garantizaba que la estimación fuese correcta; de hecho la única forma que teníamos para corregirnos era por comparación con los parámetros ideales previamente estimados. Por lo tanto, en un robot real no hubiese sido posible obtenerlos de esta forma.

Se intentó mejorar el resultado de diversas formas (filtrando adecuadamente las medidas, tomando un mayor número de muestras, variando las excitaciones, etc.), sin embargo, el hecho de que la dinámica del sistema sin reductoras fuera tan rápida, imposibilitaba la obtención de unos parámetros que no se apoyara en los del caso real.

Se nos ocurre que quizá hubiera sido una posibilidad tratar de obtener los parámetros con el robot controlado en bucle cerrado. Para ello, se podría haber diseñado controladores que estabilizasen al sistema y permitieran obtener movimientos más lentos. Aunque no se ha llegado a implementar esta estrategia, para el diseño de controladores se podrían haber utilizado modelos aproximados (suponiendo el orden de magnitud de las masas e inercias, por ejemplo) o a partir de la respuesta en frecuencia del sistema. A pesar de que el modelo sería aproximado, a frecuencias bajas debería ser capaz de controlar el sistema.

Una vez obtenidas unas estimaciones que se consideran aceptables, podríamos pensar en un primer momento que al reconstruir el modelo y simular, los movimientos deberían ser similares a los que hiciese el robot real con las mismas señales de excitación. Sin embargo, es importante recordar que estamos trabajando en bucle abierto y en este caso, el sistema es mucho más rápido que en el caso de las reductoras. Más concretamente, la constante de tiempo del sistema es inversamente proporcional al cuadrado de R. Puesto que en el dominio del tiempo la respuesta del sistema se modela por funciones exponenciales, en poco tiempo las incertidumbres paramétricas harán que el resultado diverja enormemente del esperado. A pesar de este hecho, los parámetros estimados sí serán útiles para la síntesis de controladores.



Conclusiones

- El método de estimación utilizado parte de suponer una cierta estructura para el modelo dinámico. El grado de sofisticación del modelo influirá en la calidad de la estimación. Para el caso que se ha tratado no se ha apreciado esta influencia, pues el robot simulado (considerado como robot real) se correspondía fielmente con el modelo elegido. Sin embargo, en un sistema físico aparecerían otros fenómenos que aquí no se han tenido en cuenta, como por ejemplo la fricción estática.
- Para realizar la estimación por mínimos cuadrados es imprescindible obtener una base de parámetros de dimensión mínima, de tal forma que gamma tenga rango completo. En la obtención de esta base es muy probable que aparezcan combinaciones lineales de columnas de la matriz gamma, lo que se traduce a que habrá parámetros que no puedan ser estimados de manera independiente pero sí en combinación lineal junto a otros. A pesar de ello, no debería ser un impedimento para reconstruir el modelo dinámico.
- A la hora de tratar con las ecuaciones, tener cierta intuición del significado físico de los parámetros puede ayudar a detectar posibles errores.
- Al recopilar datos para construir el sistema sobredimensionado, conviene que no haya muchas filas iguales entre sí, pues podría obtenerse una matriz de rango deficiente.
- Trabajar con medidas reales implica que existirá una cierta incertidumbre paramétrica. A pesar de ello, los modelos obtenidos siguen siendo útiles para simulación y control.
- La calidad de las medidas puede ser decisiva a la hora de lograr una buena estimación. Dado que la aceleración ha de estimarse mediante derivadas numéricas, es casi imprescindible que la señal de la velocidad no presente demasiado ruido.
- Existen dos posibilidades para obtener las velocidades. En el caso de que se disponga de tacómetros, la velocidad puede medirse; si presenta ruido conviene filtrarlo. Si el filtrado va a realizarse en digital, hay que prestar cierta atención a detalles referidos al muestreo de señales, pues puede darse el caso de que se produzca aliasing y en consecuencia sea más difícil filtrar. Respecto a este punto, también es posible que en simulación se haya estimado mal cuál podría ser la frecuencia del ruido; en un caso real dichas componentes frecuenciales podrían encontrarse por ejemplo en la frecuencia de la red eléctrica, y en ese caso no aparecería el problema antes descrito.
Si no se puede medir la velocidad habría que obtenerla derivando. Igualmente conviene filtrar tanto antes como después de derivar.
- Los filtros utilizados, de tipo IIR, introducen un cierto desfase que es dependiente de la frecuencia. Si este desfase es grande pueden aparecer errores a la hora de estimar parámetros, como por ejemplo términos iniciales negativos. Sin embargo, como la identificación se realiza fuera de línea, pueden aplicarse técnicas que corrijan la distorsión de fase. En este trabajo se utilizó la orden `filtfilt` de MATLAB y dio buenos resultados. Es posible que hubiese sido más recomendable diseñar un filtro FIR con fase lineal, pues en este caso podría corregirse simplemente mediante un retraso de muestras.

- En el caso del robot de accionamiento directo, las magnitudes de las intensidades son mucho mayores, pues para aplicar un mismo par sin reductoras es necesaria una intensidad mayor. Otra posibilidad sería considerar que se trata de un motor distinto con una constante de par mayor.
- La ausencia de reductoras hace al sistema mucho más rápido, por lo que trabajar con él en bucle abierto se hace más difícil que en el primer caso. Por el mismo motivo, al realizar una comparativa entre el robot y el modelo el resultado diverge mucho antes para el caso de accionamiento directo. Considerando cada articulación como un sistema de segundo orden con un polo en el origen, la constante de tiempo del otro polo es inversamente proporcional al cuadrado de R. Por ello, las funciones exponenciales en el dominio del tiempo toman valores muy distintos.

3. Control Dinámico

Partiendo de las medidas del sistema con accionamiento directo, se han hecho experimentos tanto con ruido de cuantización y del tacómetro simulado como sin ellos.

Varios controles a la hora de implementar el caso de medidas reales se han omitido, pues o bien eran redundantes con casos anteriormente mostrados, o el sistema se volvía inestable debido a la técnica de control empleada.

Medidas Perfectas

Para hacer la función de transferencia de cada articulación, se procede a linealizar en torno a velocidades y aceleraciones nulas. Se utiliza de la expresión de los pares aplicados a los motores, en forma matricial, y se despejan elementos suponiendo senos y cosenos en el peor caso (=1), despreciando términos gravitatorios y despreciando términos de varios órdenes menores o cuadráticos. Pese a que se haya usado accionamiento directo, se aprecia una dominancia de los términos de la diagonal principal sobre el resto, por lo que se pueden despreciar (lo cual conlleva un pequeño error en el modelo).

Una vez se tiene dicha expresión del par de la articulación i , la cual solo depende de los valores en velocidad o aceleración de dicha articulación i , se hace una transformada al dominio frecuencial y se obtiene su función de transferencia incremental.

$$G_1(s) = \frac{\Delta q_1}{\Delta \tau_1} = \frac{41.67}{(293.7s + 1)s} \quad \left(\frac{\text{rad}}{\text{Nm}}\right)$$

$$G_2(s) = \frac{47.06}{(415.7s + 1)s} \quad \left(\frac{\text{rad}}{\text{Nm}}\right)$$

$$G_3(s) = \frac{23.33}{(26.99s + 1)s} \quad \left(\frac{\text{rad}}{\text{Nm}}\right)$$

Dicho esto, se parte de que el tiempo de muestreo usado normalmente por las computadoras que controlan es de 0.001 segundos y que la frecuencia de resonancia del robot es de unos 300 Hz aproximadamente. Con estos datos, se obtiene que la frecuencia máxima a la que podría trabajar el robot, según el tiempo de muestreo facilitado, es del orden de 3000 rad/s. Sin embargo, para lograr un correcto funcionamiento de este, se establece su ancho de banda a unos 150 rad/s, lo que equivale a unos 20 milisegundos para el tiempo de subida en bucle cerrado.

$$t_s^{bc} = \frac{\pi}{\omega_c} \approx 20 \text{ ms}$$

3.1. PD sin cancelación

Para poder realizar un PD sin cancelación, se puede optar por hacer uso de técnicas frecuenciales o diseñarlo analíticamente. Tras probar ambos métodos, dando resultados prácticamente idénticos, se muestran los resultados obtenidos por técnicas frecuenciales.

La ley de control obedece a la expresión:

$$\tau = K_P(q_r - q) + K_D(\dot{q}_r - \dot{q})$$

Donde τ ahora se refiere a la señal de control (fuerzas o pares).

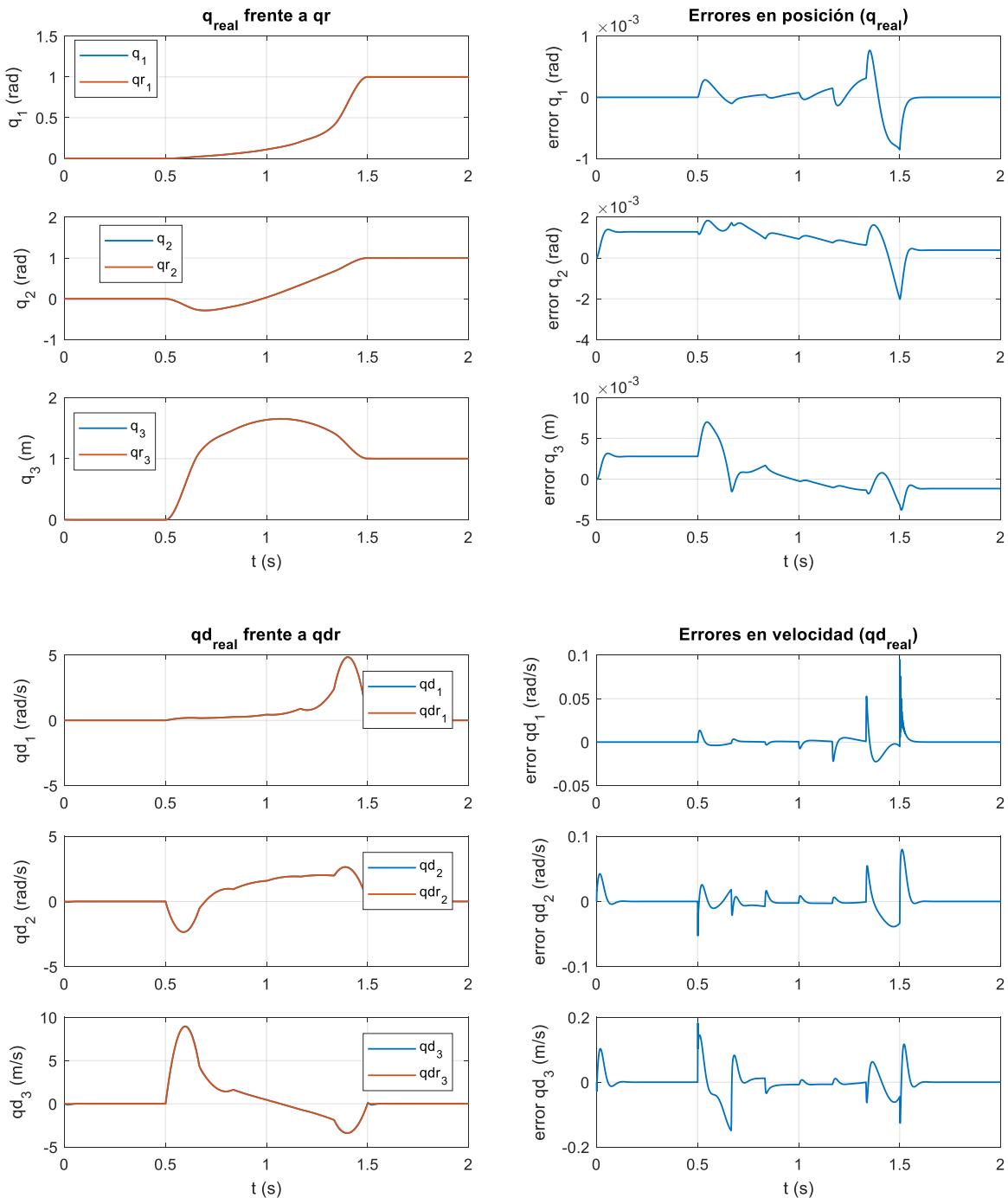
Las ganancias del controlador pueden calcularse como:

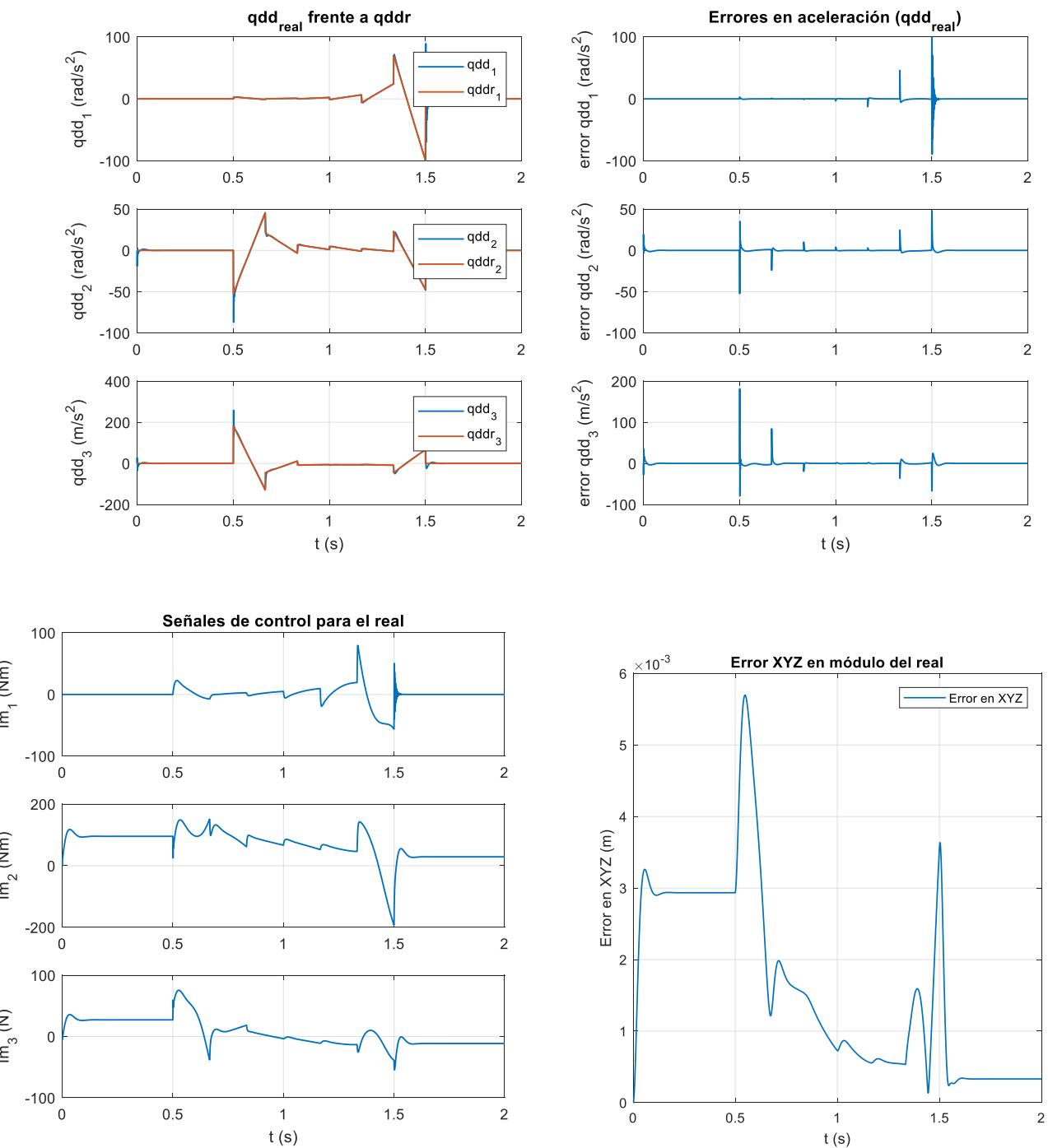
$$K_P = 10^{-K/20} \quad K_D = K_P \cdot \tau_s$$

Siendo K el margen de ganancia a $w_c=150$ rad/s

Articulación j	K_P	K_I	K_D
1	5.9566e+04	0	1.0419e+03
2	7.4131e+04	0	1.2966e+03
3	9.7724e+03	0	170.9284

La simulación proporciona los siguientes resultados:

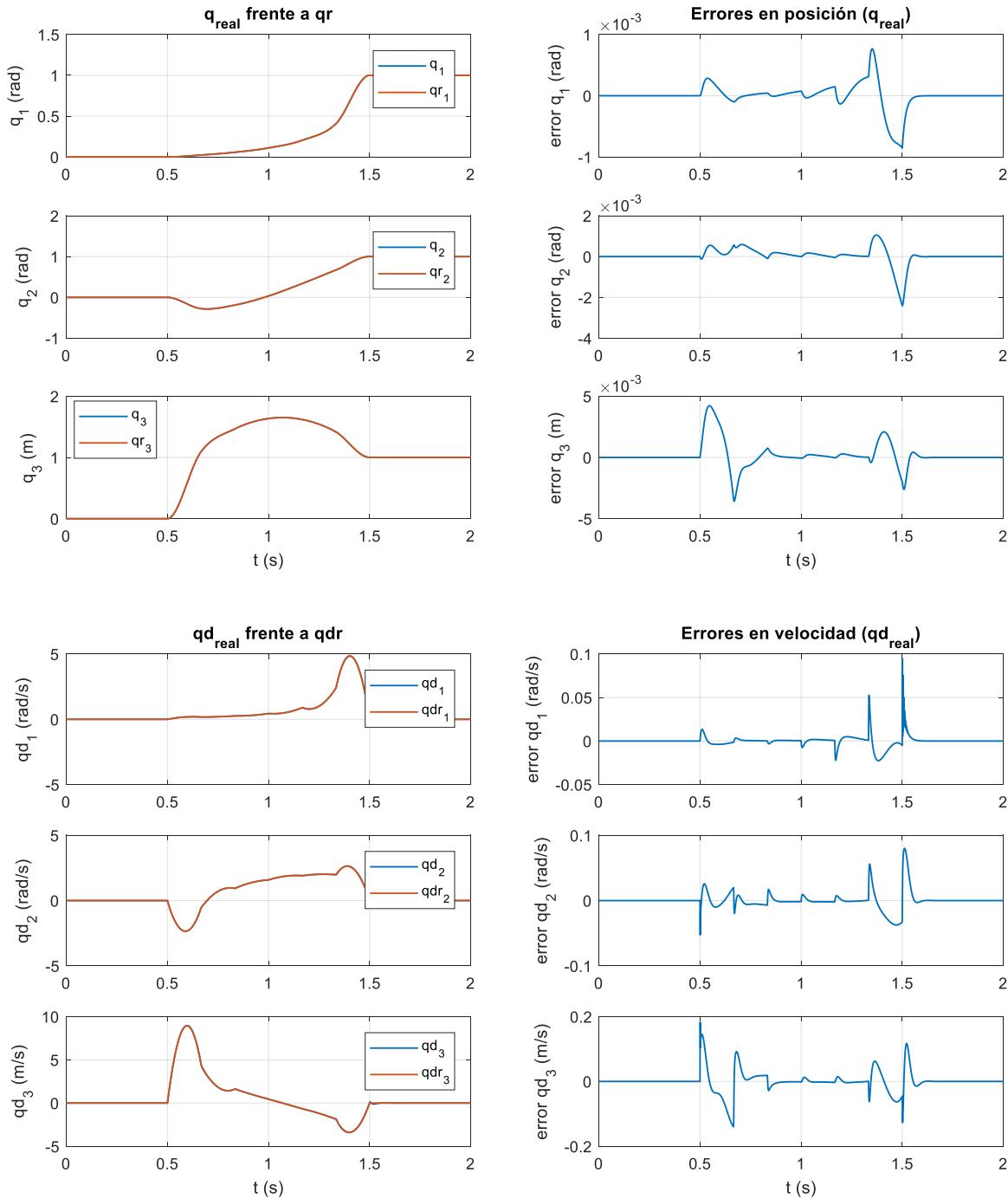


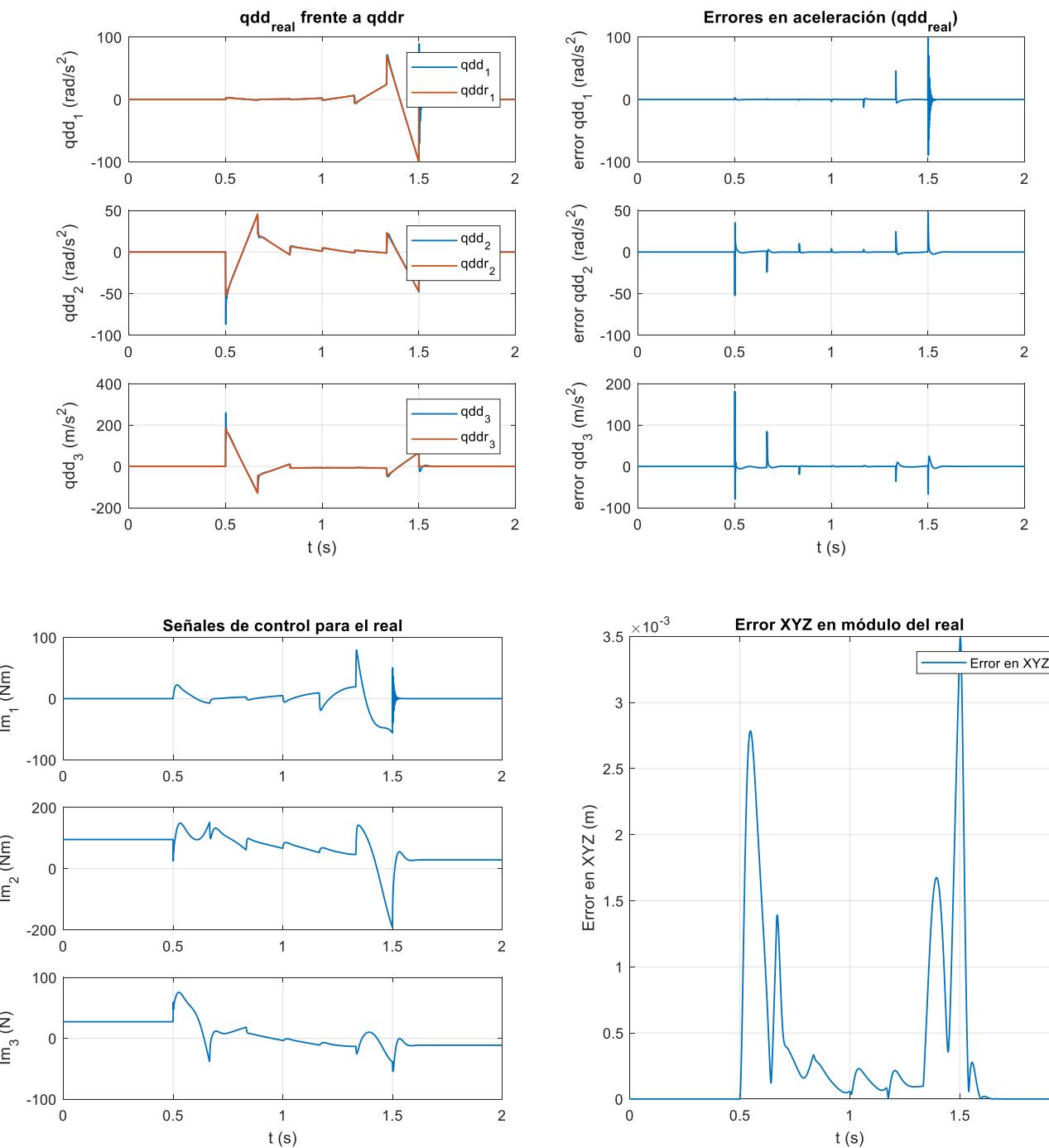


Se aprecia un error en régimen permanente debido a que contamos la gravedad como perturbación constante. Por otra parte, aparecen errores significativos al comienzo y al final del movimiento. Esto se debe a que los altos valores de aceleración hacen que los acoplamientos no sean despreciables.

3.2. PD con gravedad compensada

Para poder eliminar el error en régimen permanente debido a la gravedad también podría optarse por añadir efecto integral, pero es preferible precompensar dicho efecto gravitatorio. Por ello, se propone sumar a la señal de control otra señal de control que solo dependa de la gravedad, de manera que mediante esa función se tenga en cuenta el efecto gravitatorio que no se incluye previamente.





Como se puede observar, el error en régimen debido a la gravedad permanente ha desaparecido, sin embargo, aún se puede ver la presencia de errores en el transitorio, con picos debidos a la cuantización de la trayectoria, además de los términos de Coriolis despreciados al linealizar.

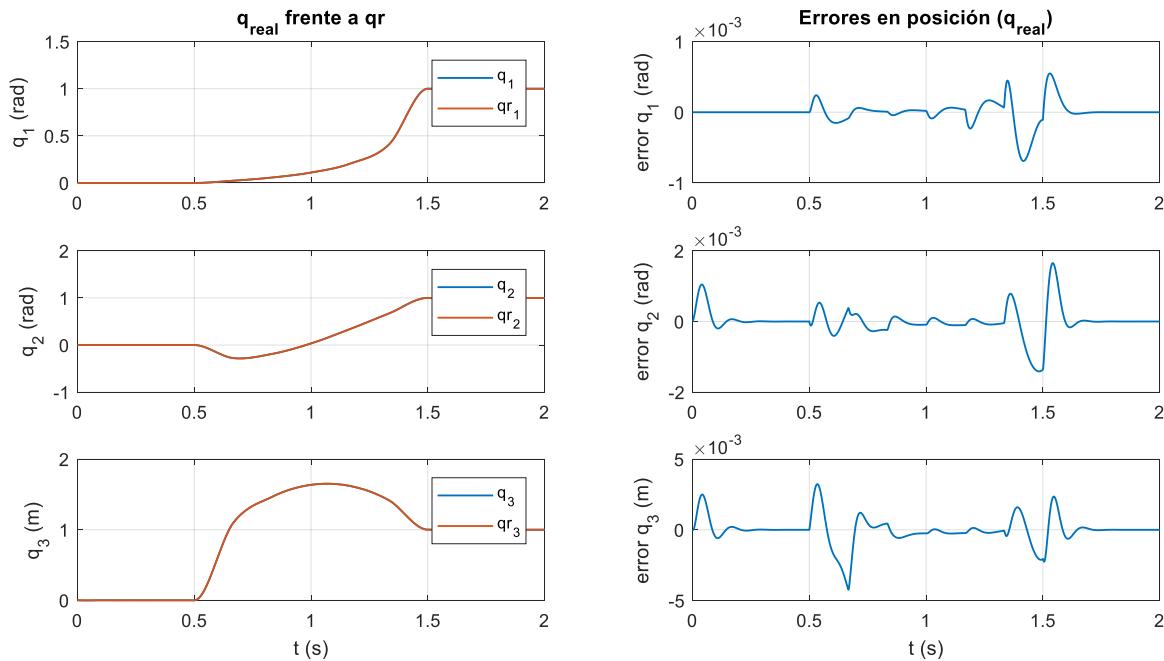
3.3.PID sin cancelación

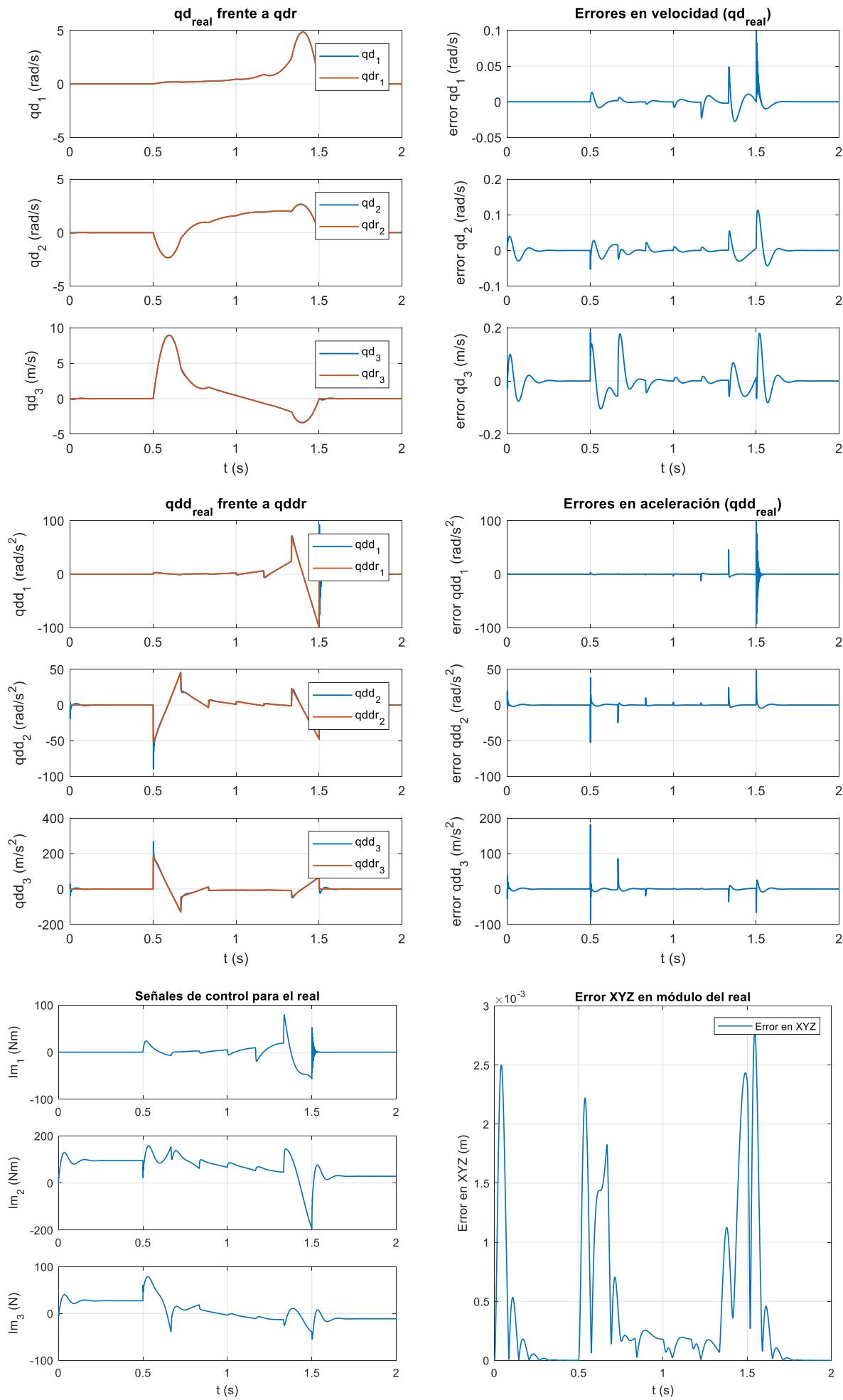
El cálculo de las constantes de control es relativamente directo, por lo que sabiendo el tiempo de subida que usaremos (igual que en pd) y que el bode del modelo es el mismo que antes, resulta directo obtener:

Articulación j	K_P	K_I	K_D
1	5.9024e+04	8.1741e+05	1.0655e+03
2	7.4307e+04	1.0291e+06	1.3414e+03
3	9.7956e+03	1.3566e+05	176.8322

Bajo estas condiciones, podemos apreciar que el error en régimen permanente es nulo, pero sobreoscila (véase el inicio de las articulaciones afectadas por la gravedad, 2 y 3).

Se omitirá la compensación de gravedad para el PID, pues apenas se aprecian diferencias entre compensarla y no (excepto las oscilaciones del permanente). Esto puede deberse a que al introducir el efecto integral en un PID sin cancelación de dinámica el propio integrador corrija la perturbación de la gravedad sin necesidad de necesitar compensador.





3.5. Control con precompensación dinámica

En los apartados anteriores se observaron las mejoras de compensar el efecto gravitatorio, pero, además de esto, se puede compensar la matriz de Coriolis y de fricción viscosa de los motores, compensando así los efectos dinámicos del modelo. Para ello es necesario disponer del modelo completo del robot.

Si se realiza este cambio, se observa que el control, en su salida, puede expresarse como $\tau = M(q) \cdot \ddot{q}$

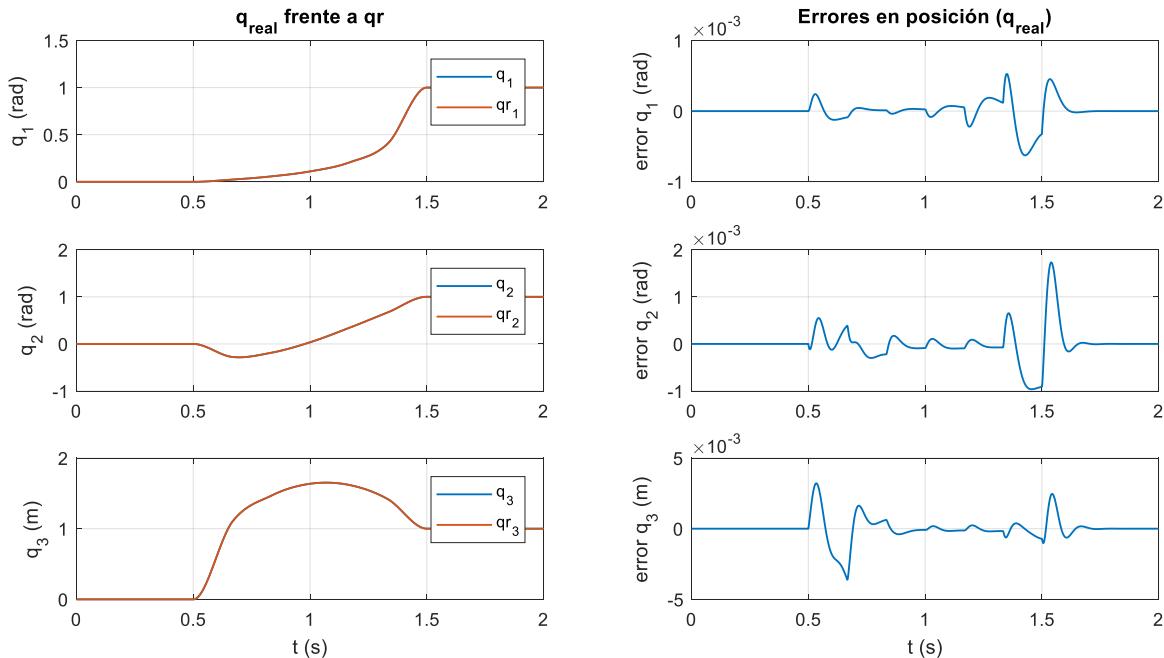
Los nuevos modelos linealizados son:

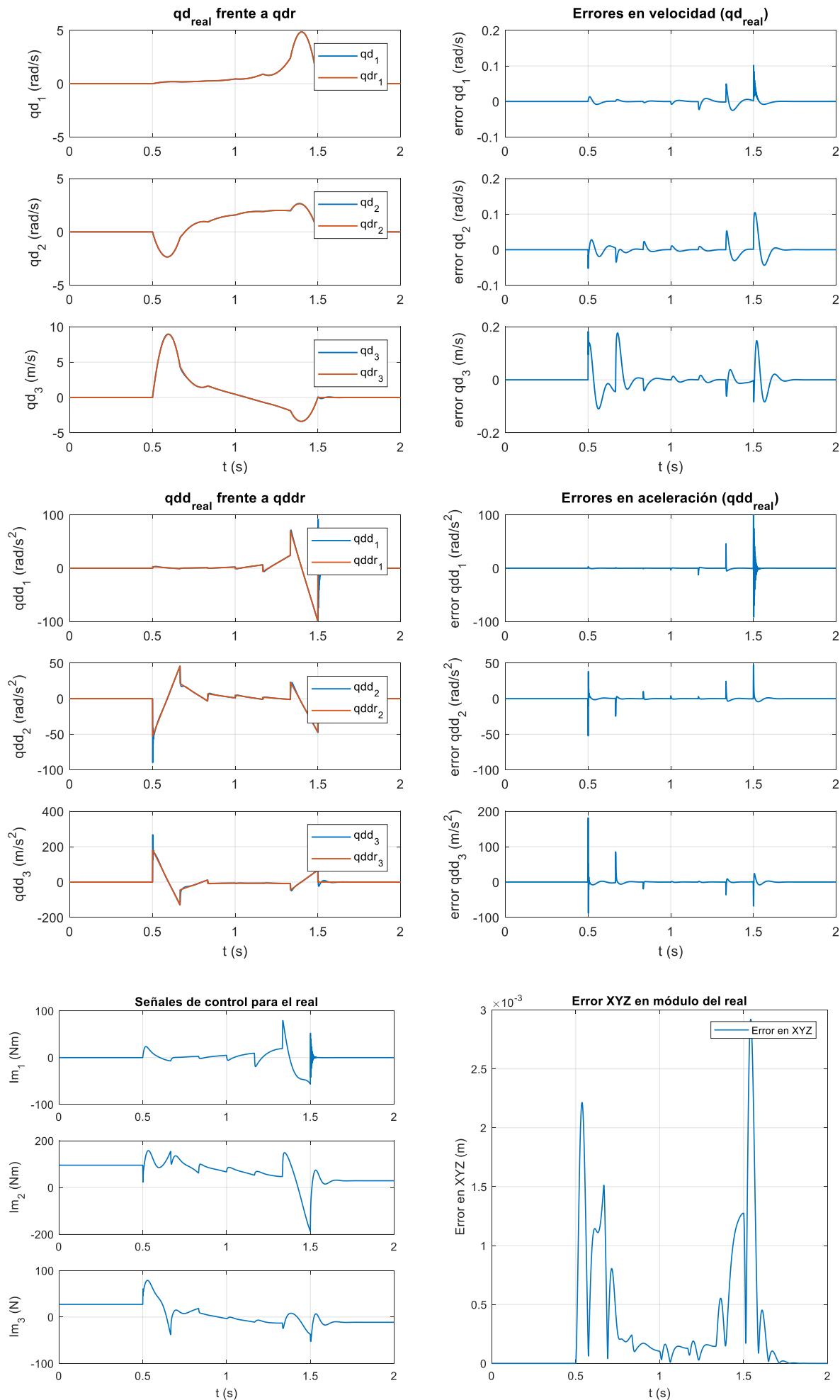
$$G_1(s) = \frac{\Delta q_1}{\Delta \tau_1} = \frac{0.1418}{s^2} \quad \left(\frac{\text{rad}}{\text{Nm}} \right)$$

$$G_2(s) = \frac{0.1132}{s^2} \quad \left(\frac{\text{rad}}{\text{Nm}} \right)$$

$$G_3(s) = \frac{0.8644}{s^2} \quad \left(\frac{\text{rad}}{\text{Nm}} \right)$$

El controlador PID por técnicas frecuenciales para cada articulación resulta igual que el anterior.





3.6. Control con par calculado

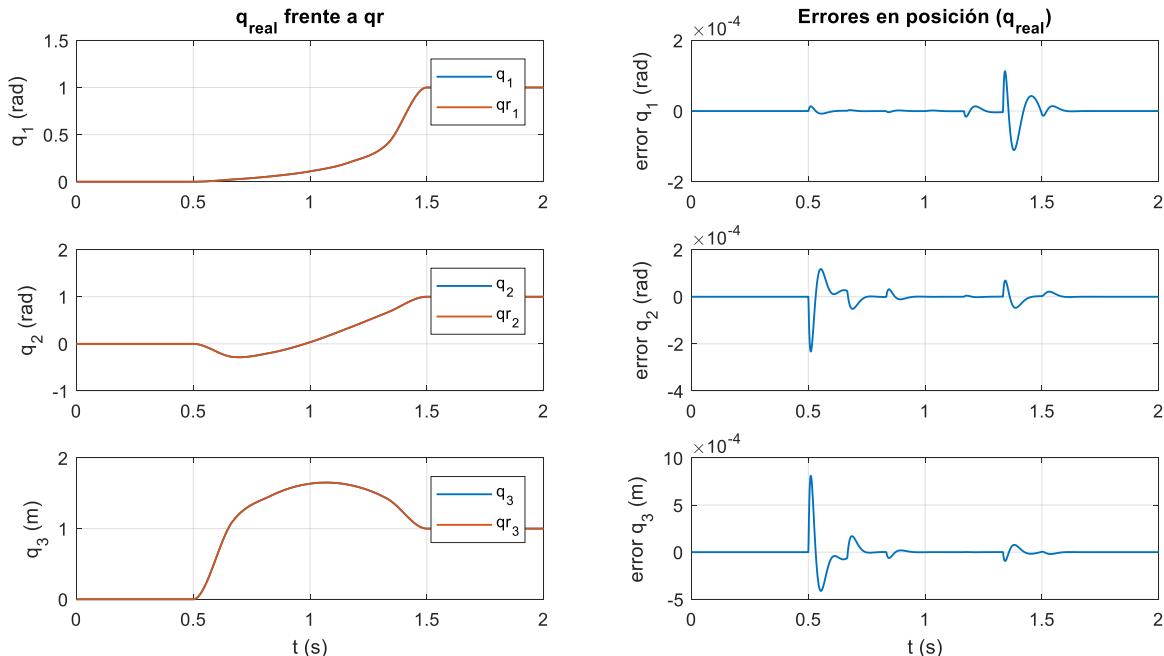
Para ello, se linealiza el modelo por realimentación. Se obtiene, para cada articulación, un modelo desacoplado que se corresponde con un doble integrador.

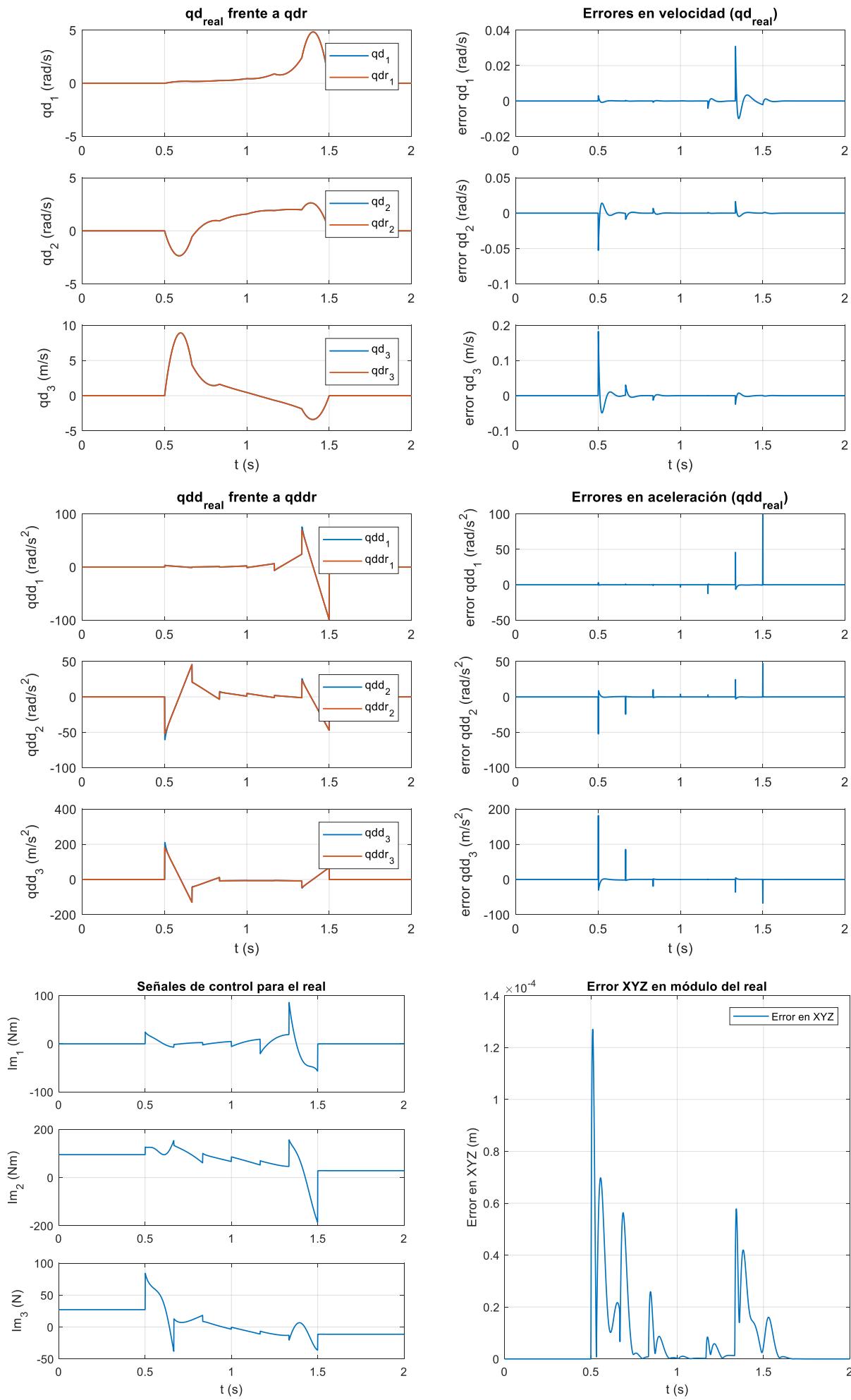
Si se realiza este cambio, se observa que el control, en su salida, puede expresarse como $\tau = \ddot{q}$

En este caso han de recalcularse las constantes de control (las mismas para cada articulación, pues la respuesta en frecuencia será igual si el modelo de cada uno es idéntico al otro).

Articulación j	K_P	K_I	K_D
1	8.4339e+03	1.1680e+05	152.2514
2	8.4339e+03	1.1680e+05	152.2514
3	8.4339e+03	1.1680e+05	152.2514

Se obtiene como resultado:





Medidas reales

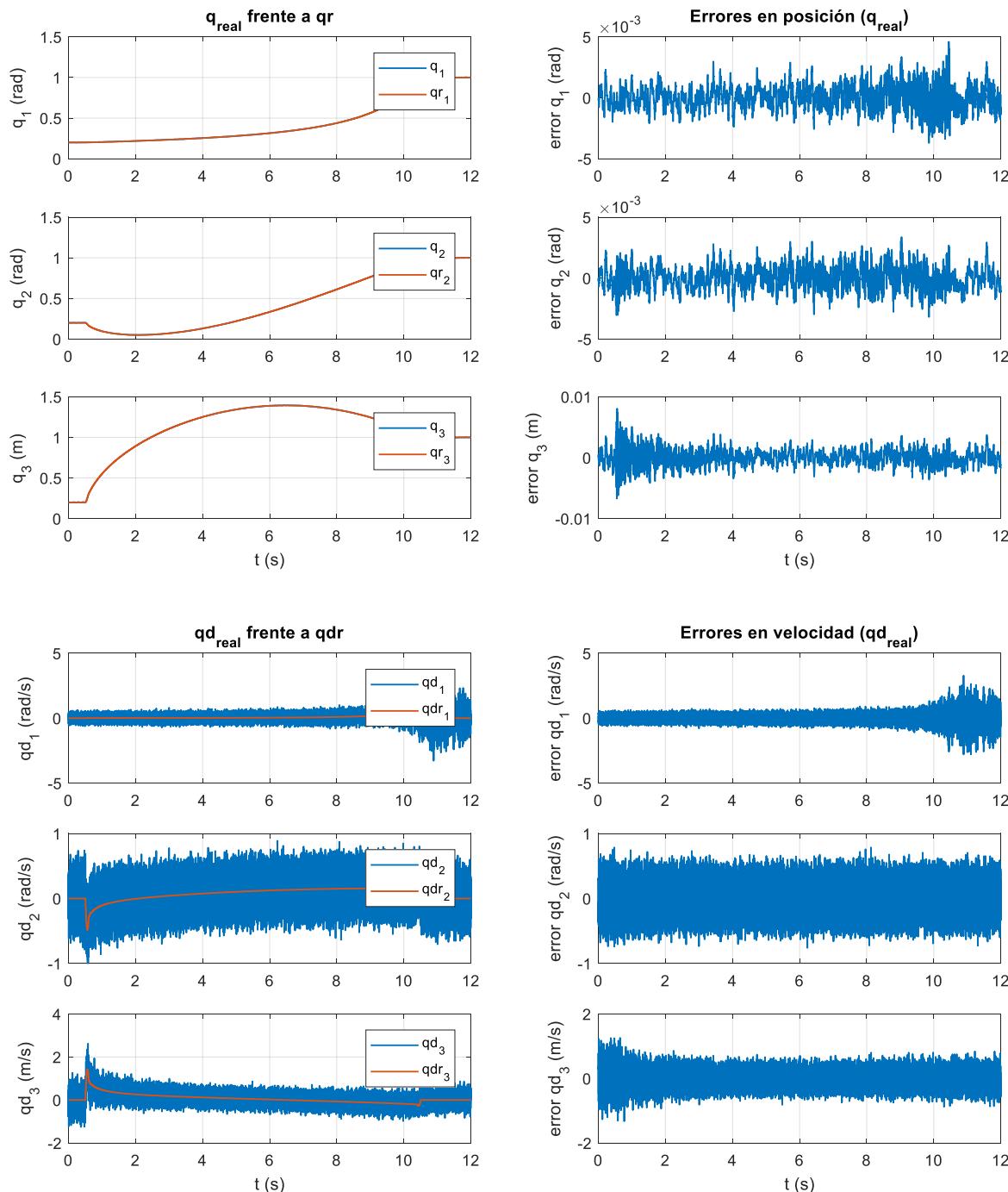
Se procede a controlar ahora el sistema, tanto mediante controladores de apartados anteriores como diseñados nuevamente, implementando el ruido natural propio de los sistemas reales, cuantización del encoder (9 bits) y ruido en el tacómetro (2% del valor máximo de la velocidad sin ruido).

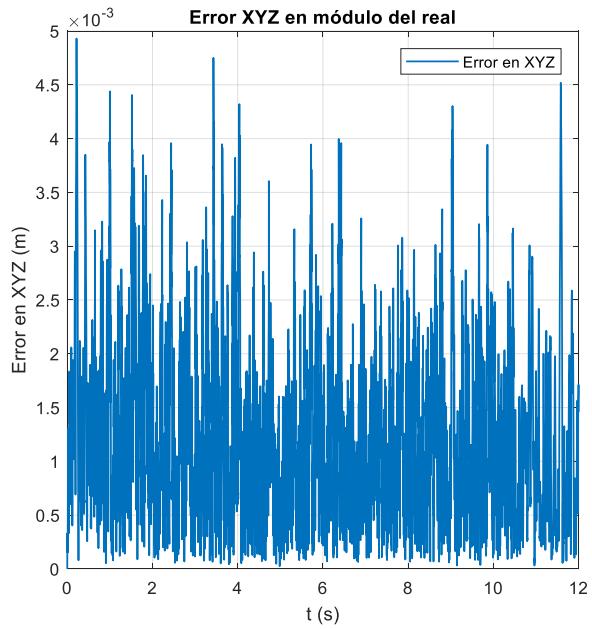
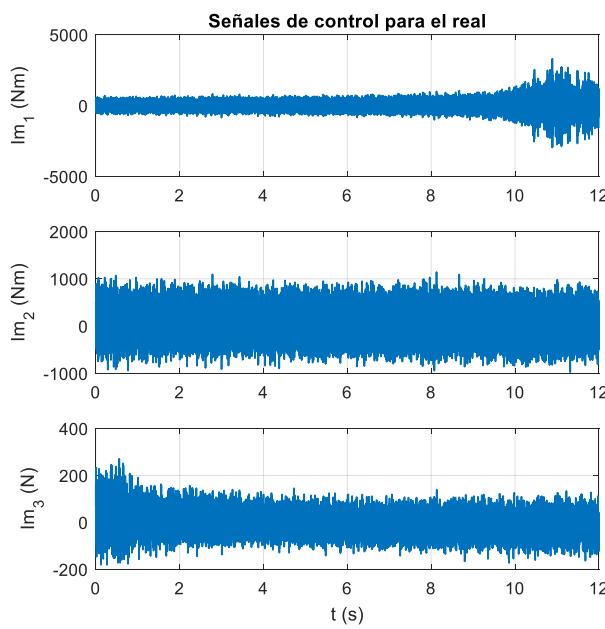
Puesto que la aceleración no se mide en un robot real, se omite la medida de esta en la simulación.

En caso de necesitarla necesariamente, puede obtenerse la aceleración del generador de trayectorias, lo cual no será exactamente la aceleración del brazo.

3.7. PD con compensación de gravedad

Se calcula la señal de control para cada articulación en función de los valores medidos de las articulaciones. Se ha cambiado la referencia con respecto a los resultados anteriores para poder apreciar mejor los errores en el permanente.





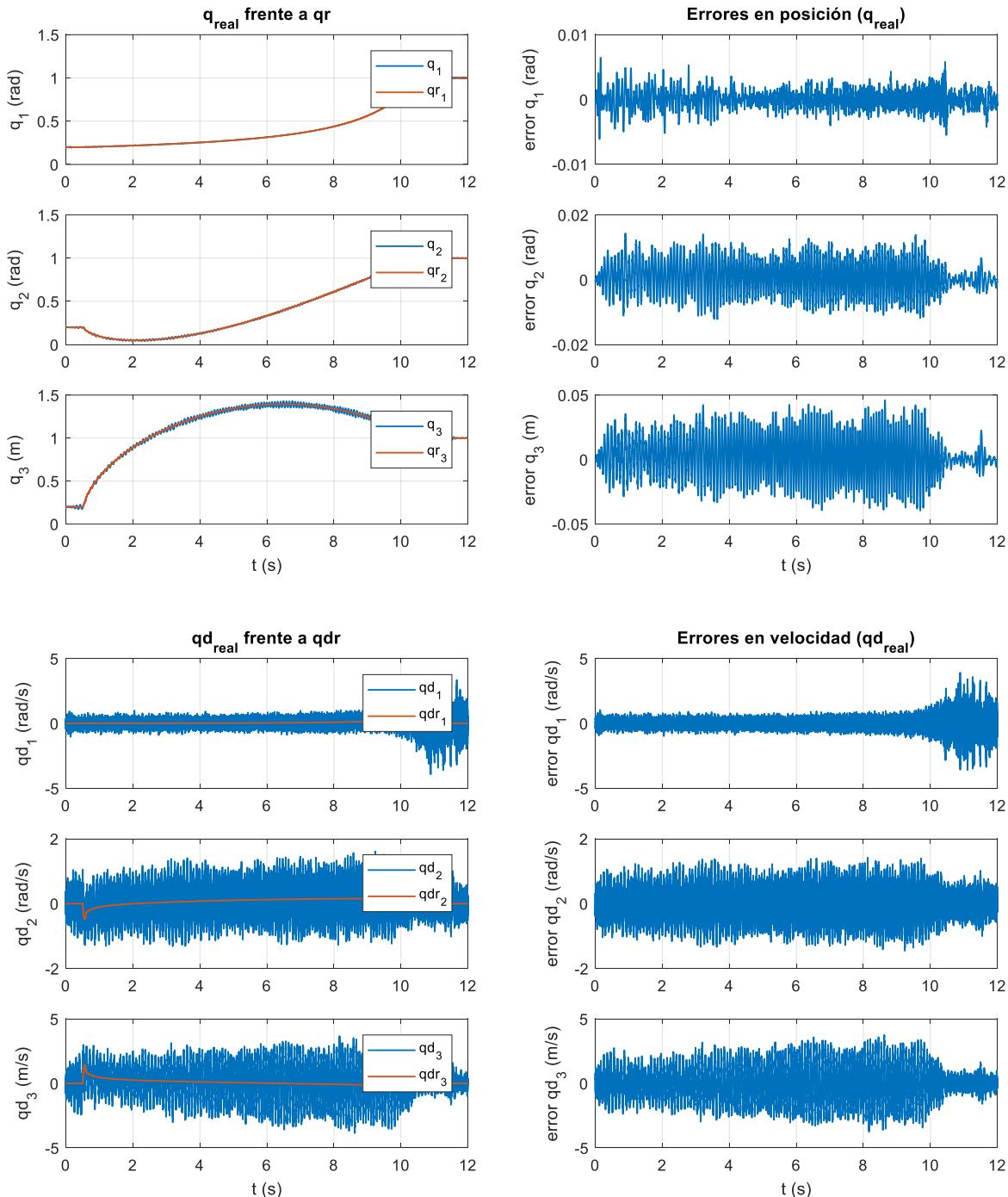
Se aprecia un error debido al que el ruido añadido. Podemos apreciar este efecto tanto en posición como en velocidad.

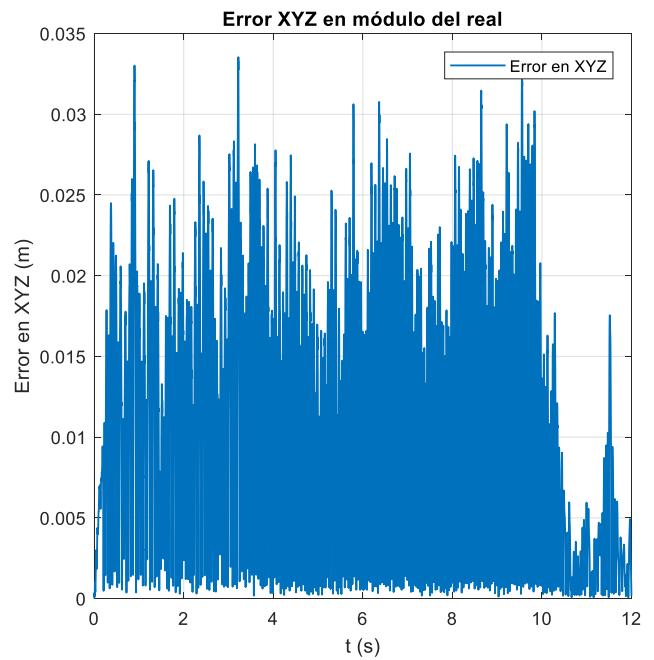
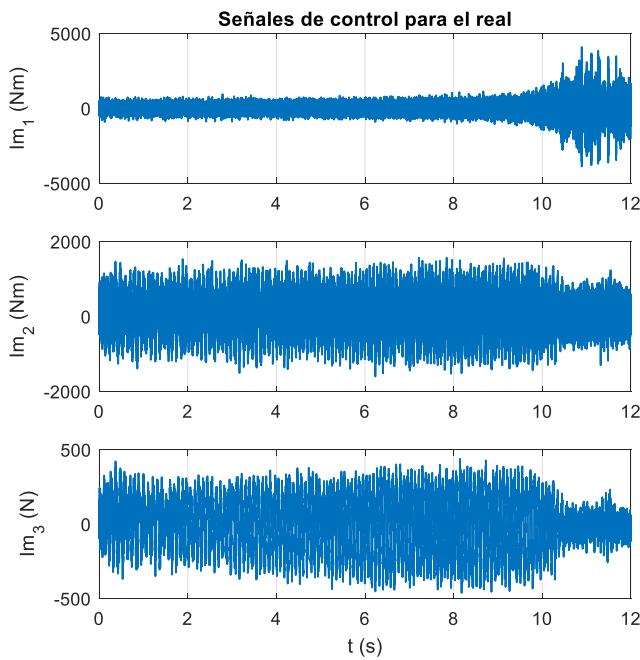
Dicho error podría reducirse filtrando la velocidad. Se ha probado a filtrarla, mediante un filtro Butterworth con la correspondiente frecuencia de corte, pero la diferencia lograda era prácticamente nula. Para la señal de control sería necesario un filtro y una limitación.

También hay ruido debido a que el modelo que estamos utilizando es el estimado con medidas no reales, por lo que al compensar la gravedad se introduce un pequeño error. Sin embargo, este es prácticamente despreciable con respecto al introducido por el ruido. El cuantizador también añade ruido, pero nuevamente el ruido del tacómetro es dominante.

3.8. PID con compensación de gravedad

Se puede apreciar claramente que el control empeora tanto en posición como en velocidad.

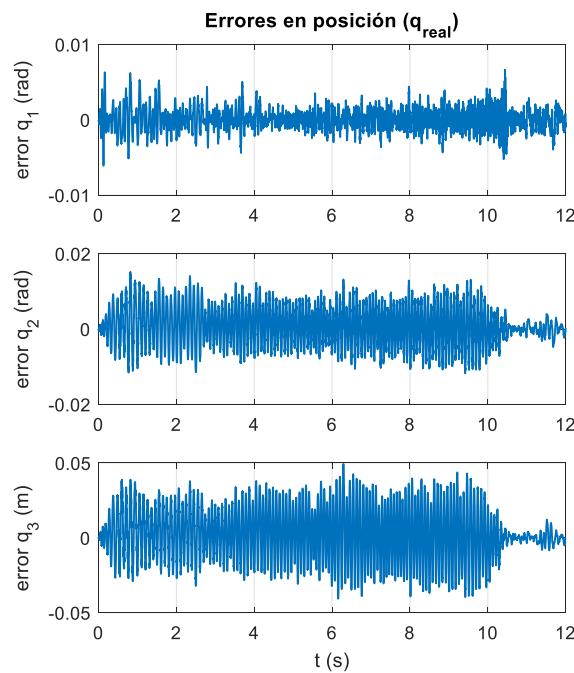
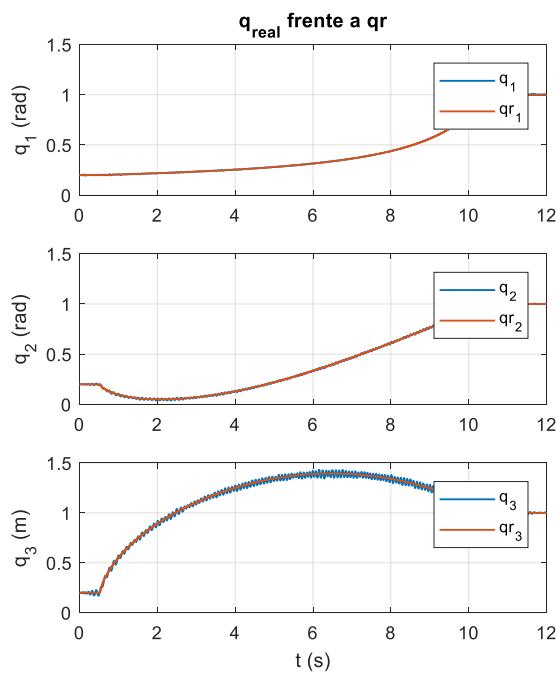


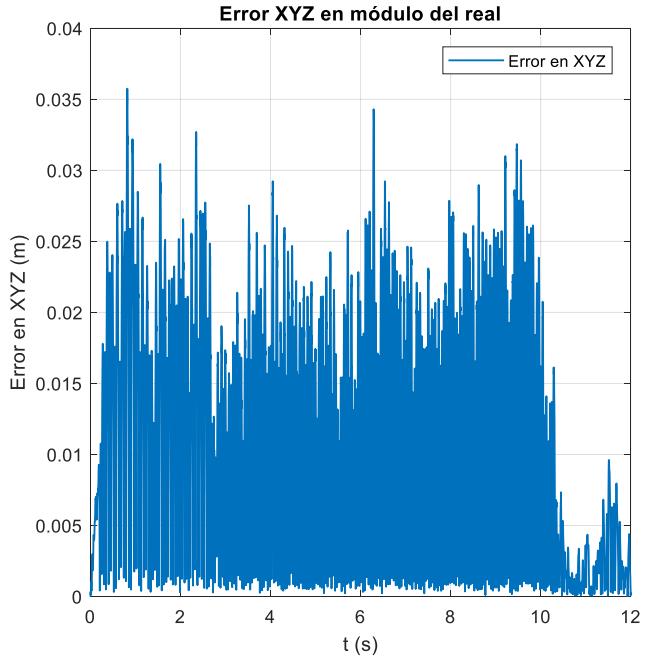
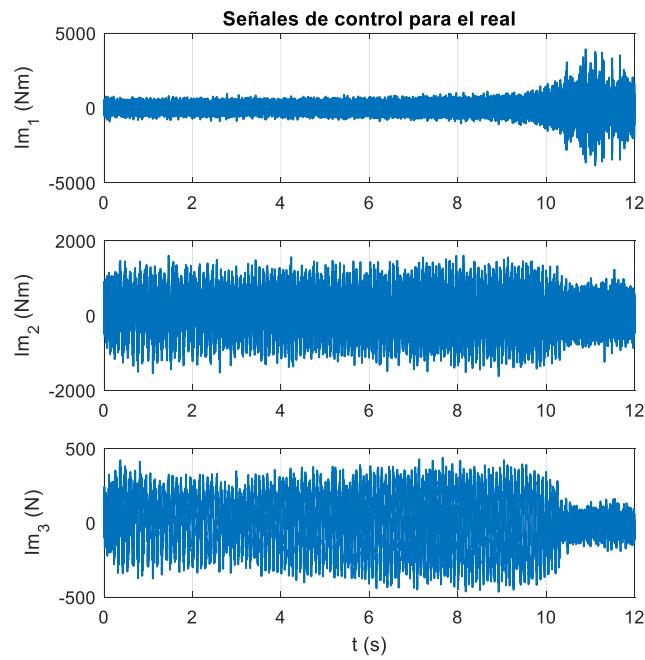
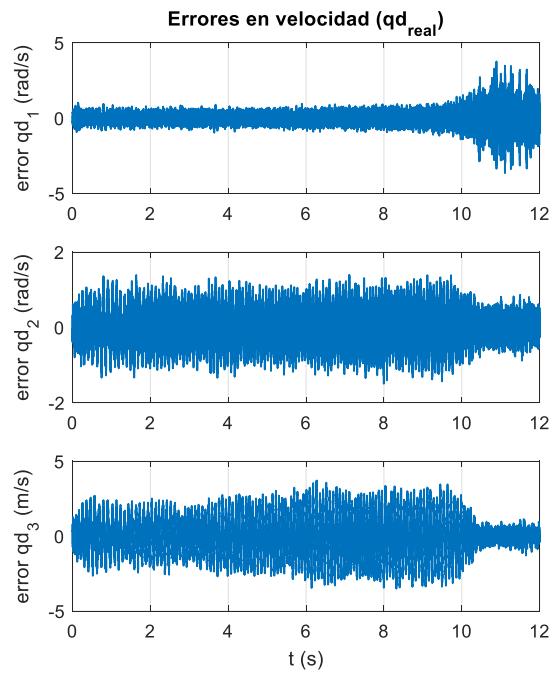
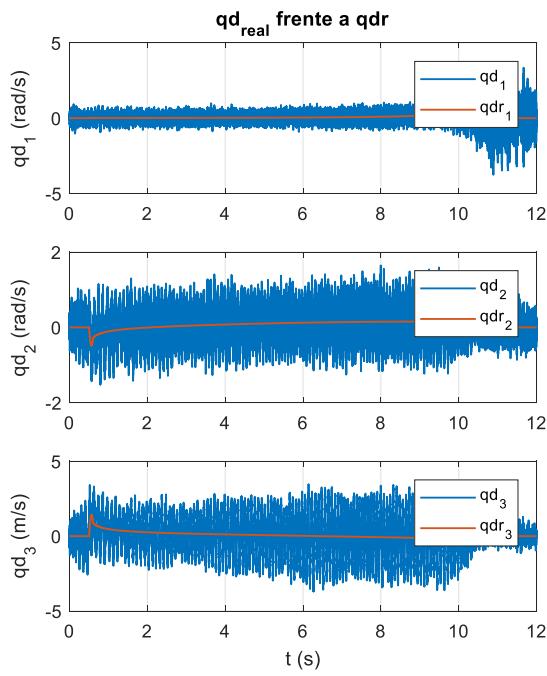


Al igual que ocurrió en el PID del modelo exacto, se aprecian sobreoscilaciones no deseadas. Esto se debe a la introducción de efecto integral con ruido en velocidad. Con un ajuste más fino de los parámetros del controlador (T_i menor) podría lograrse reducir algo dicha sobreoscilación.

3.9. PID con FeedForward (precom. dinámica)

Además de que la realimentación está hecha con las aceleraciones de referencia, se controla a baja velocidad, por lo que la diferencia con los resultados del apartado anterior es nula.

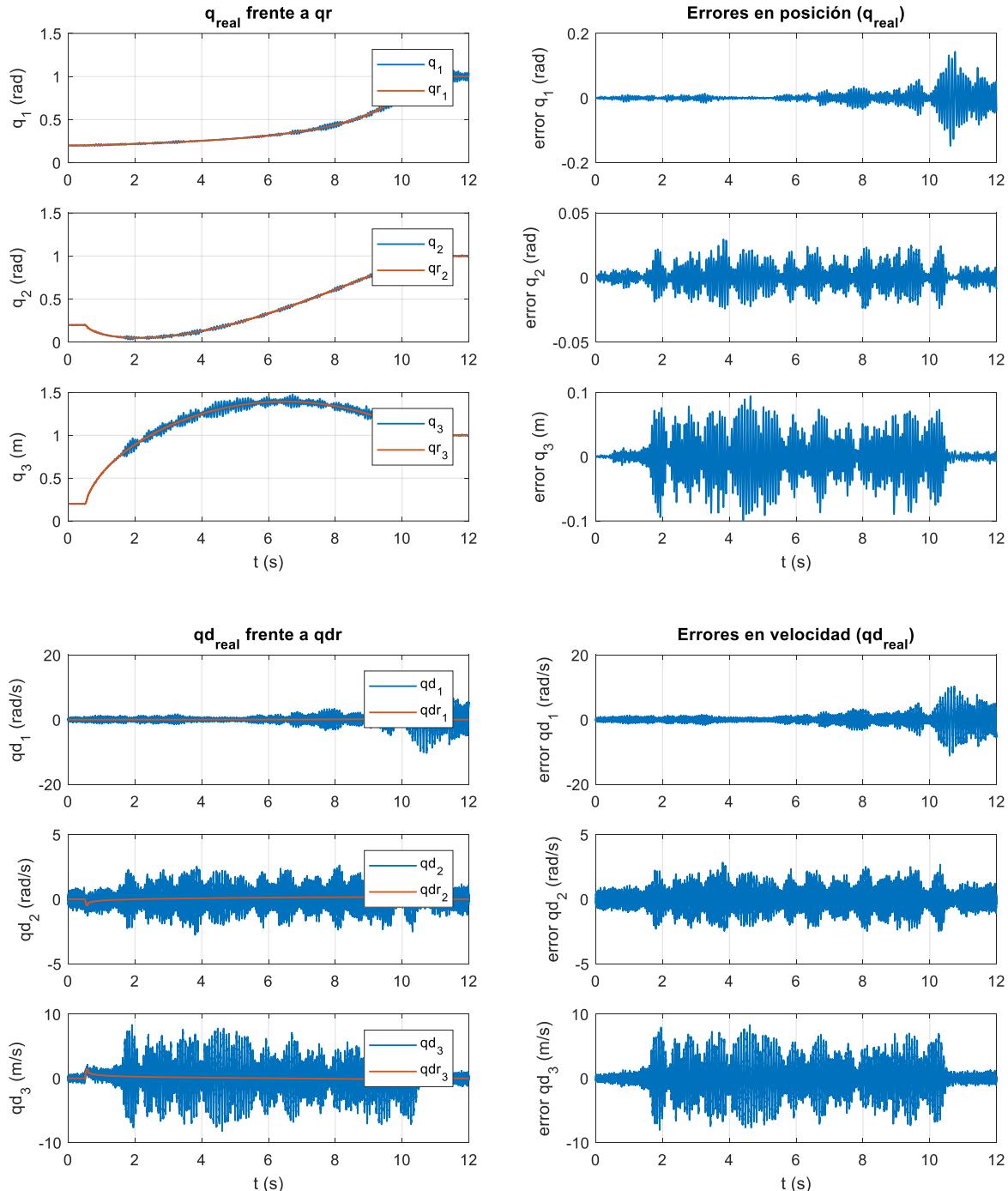


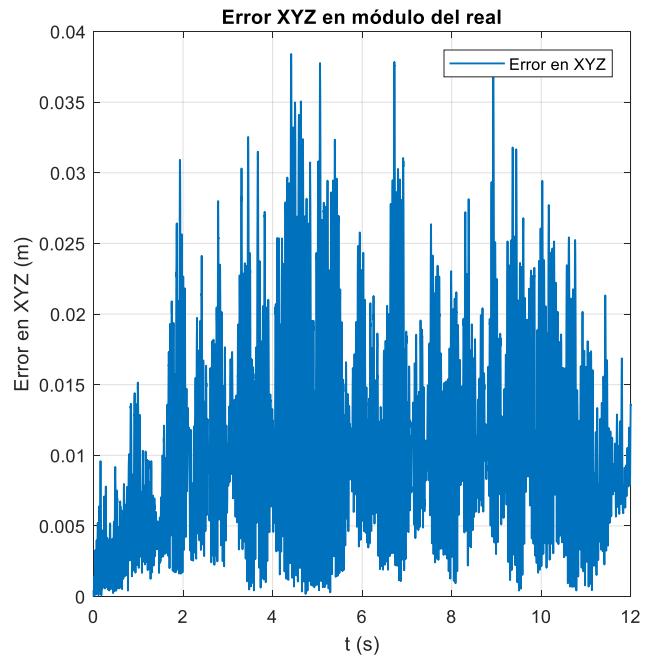
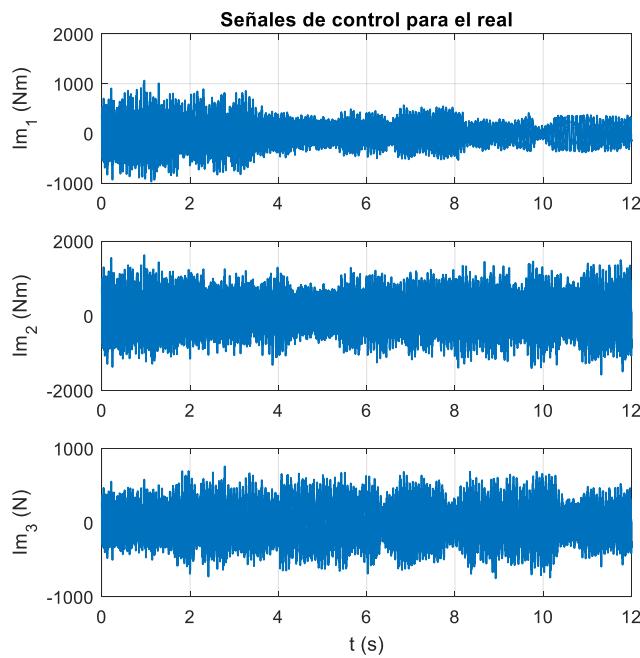


3.10. PID con par calculado precompensando con referencias

Al realimentar los valores de referencias, controlamos con valores que no son los reales del sistema en ese instante, por lo que inevitablemente, cuanto menos se parezca el modelo utilizado al real, mayores serán los efectos de esta diferencia.

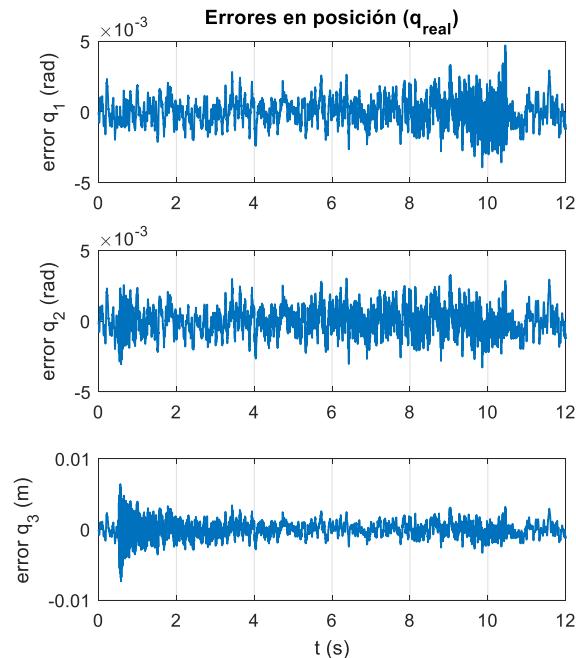
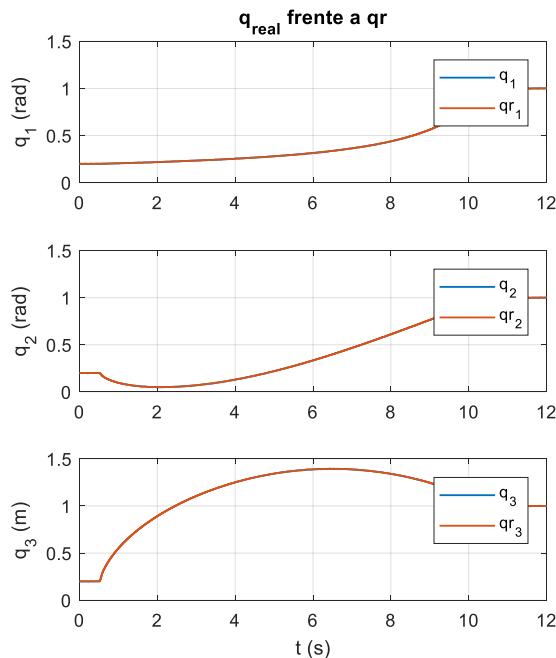
En este caso, se aprecian sobreoscilaciones durante la referencia posiblemente debidas al efecto integral del controlador.

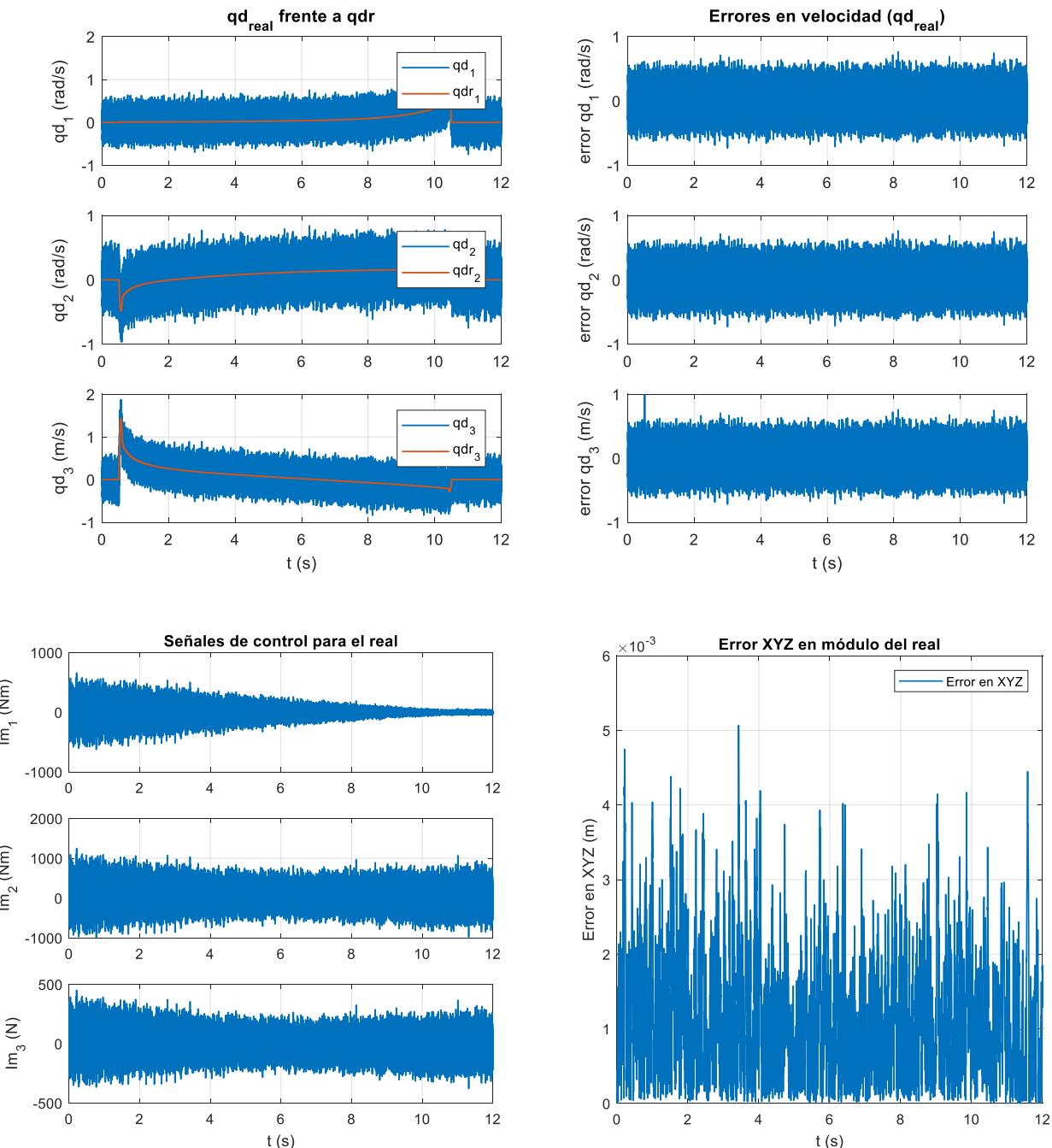




3.12. PD con par calculado precompensando con referencias

Se puede apreciar un error mucho menor al caso con PID, por lo que podría intuirse que el error tan grande antes cometido es debido definitivamente a la introducción de efecto integral. De todos los controladores realizados, este es sin duda el que mejor resultados nos ofrece, con un mínimo error en posición.





4. Conclusiones

- En todos los casos se ha considerado la gravedad como una perturbación mantenida. Dicha perturbación se puede corregir mediante la precompensación de la gravedad o mediante el efecto integral. En cualquier caso, se ha demostrado que la mejor opción es precompensar.
- Se ha optado por mostrar resultados a velocidades bajas, pues linealizamos en torno a este aspecto. Sin embargo, podría haberse especificado en el generador de trayectorias un tiempo menor para el recorrido, suponiendo una degradación a la hora de controlar el sistema en velocidad. Esto se reduce al compensar los efectos dinámicos de este.
- Los resultados mostrados en todo momento han sido utilizando accionamiento directo, es decir, con reductoras iguales a 1. Sin embargo, se han trabajado también los casos con reductora igual a 25, apreciándose en gran parte la mejoría del control en posición y velocidad.
- El tiempo de muestreo aplicado en todo momento ha sido de 1 milisegundo, lo general en las controladoras actuales, si el tiempo de muestreo fuese algo mayor, es posible que el control se realizase adecuadamente, aunque perdiendo prestaciones. En caso de disponer de un tiempo de muestreo mucho mayor, han de rehacerse los controles, pues se ha trabajado en todo momento con respecto a la frecuencia de corte de 150 rad/s.
- La incertidumbre del modelo empeora considerablemente el control, pues para precompensar cualquier factor, se requiere que el modelo sea lo más próximo al real. Sin embargo, la incertidumbre que se comete es relativamente pequeña (no superior al 10), no siendo entonces la mayor de las fuentes de error. Todo esto depende de a la velocidad a la que se requiera controlar el sistema, pues, si el modelo es bueno, puede controlarse bien a velocidades tanto altas como bajas. Sin embargo, al no ser el modelo que se dispone lo suficientemente fino, en velocidades altas podría dar problemas, pues no se podría precompensar los efectos dinámicos correctamente al no disponer del modelo exacto, lo cual degradaría considerablemente el control, pudiendo volver el sistema inestable (no es el punto de operación buscado).
- El aspecto que más error introduce en el control es el ruido de la velocidad. Se ha probado a aumentar el número de bits del encoder, notándose la cuantización en la posición de las gráficas, pero sin afectar apenas a la calidad del control. Sin embargo, se ha intentado reducir el ruido de la velocidad mediante técnicas de filtrado. Pese a eso, el ruido seguía siendo muy grande por lo que no se añadió la etapa de filtrado. Tal y como se comentó en el apartado de identificación, es posible que no se haya elegido adecuadamente la frecuencia del ruido. En cualquier caso para obtener un control deseable convendría mejorar este aspecto, sea mediante un filtro digital o analógico.
- Se han implementado tanto controladores PD como controladores PID, siendo necesario para estos últimos añadir un algoritmo de antiwindup en el control, pues el ruido de la velocidad afectaba al efecto integral en exceso, volviéndose inestable. Sin embargo, el PID aporta ruido y oscilación en el permanente no deseado, por lo que el mejor controlador para el sistema es PD par calculado realimentado con referencia.
- El error observado en las gráficas puede deberse a:
 1. El modelo utilizado difiere del real, pues los parámetros dinámicos no son exactos. Esto proporciona errores no contemplados anteriormente, corregibles según la técnica de control empleada, siendo considerablemente peor la precompensación.
 2. La implementación del control en tiempo discreto supone una degradación de las prestaciones.
 3. Se trabaja con accionamiento directo.
 4. Se realimenta con ruido en el tacómetro y cuantización en la posición y trayectoria a seguir.
 5. En el caso del PID, se amplifica el error debido al ruido en velocidad.