UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"DISEÑO DE UN SISTEMA DE CONTROL PARA UN ROBOT DE REHABILITACIÓN"

TRABAJO FIN DE GRADO

Septiembre - 2024

AUTOR: Freddy Prieto Gonzaga

DIRECTOR/ES: Adrián Peidro Vidal

INDICE GENERAL

1. INTRODUCCIÓN 3
1.1. ANTECEDENTES 3
1.2. OBJETIVOS Y APORTACIÓN DE ESTE TRABAJO 5
1.3. ESTRUCTURA DE LA MEMORIA 6
2. DESCRIPCIÓN CINEMÁTICA Y DINÁMICA DEL ROBOT 7
2.1. SÍMBOLOS Y VARIABLES CINEMÁTICAS EMPLEADAS 7
2.2. RESTRICCIONES CINEMÁTICAS APLICADAS AL ROBOT 8
2.3. MODELADO DINÁMICO DEL ROBOT 9
3. ESTUDIO Y DISEÑO DEL CONTROL EN EL ESPACIO DE ESTADOS 12
3.1. SISTEMA SISO CON DOS ESTADOS 13
3.2. SISTEMA SISO CON DOS ESTADOS Y PARTE INTEGRAL 16
3.3. SISTEMA MIMO CON SEIS ESTADOS CRÍTICAMENTE AMORTIGUADO 20
3.4. SISTEMA MIMO CON SEIS ESTADOS SUBAMORTIGUADO 26
4. APLICACIÓN DEL SISTEMA DE CONTROL AL ROBOT NO-LINEAL 29
4.1. EXPERIMENTO 1 CON EL CONTROLADOR CRÍTICAMENTE AMORTIGUADO 30
4.2. EXPERIMENTO 2 CON EL CONTROLADOR CRÍTICAMENTE AMORTIGUADO 31
4.3. EXPERIMENTO 3 CON CONTROLADOR SUBAMORTIGUADO 32
4.4. EXPERIMENTO 4 CON CONTROLADOR SUBAMORTIGUADO 33
5. CONCLUSIONES Y TRABAJOS FUTUROS 35
6. ANEXOS 37
6.1. ANEXO I: ECUACIONES CINEMÁTICAS DEL ROBOT DESCRITAS EN MATLAB 37
6.2. ANEXO II: MODELO DINÁMICO DEL ROBOT DESCRITO EN JAVA 38
7. BIBLIOGRAFÍA 44

1. INTRODUCCIÓN

1.1.ANTECEDENTES

La rehabilitación y la asistencia para las articulaciones del cuerpo humano son procesos sanitarios cruciales en la salud y la calidad de vida de las personas. Ambos procesos ayudan a recuperar o mejorar la funcionalidad, la movilidad y la autonomía de articulaciones que han sido anteriormente afectadas. Los avances tecnológicos han permitido el uso de robots (concretamente robots paralelos) en los procesos de rehabilitación y asistencia. El uso de robots puede ayudar a facilitar dichos procesos, ya que éstos pueden llegar a ser más precisos, estables y adaptables que los métodos de rehabilitación y asistencia tradicionales.

Este trabajo se centra en los robots paralelos de rehabilitación, por lo que describir el campo de aplicación es necesario. Estos robots son diseñados para ayudar a los sujetos a recuperar la fuerza, las habilidades motoras y la funcionalidad perdida de articulaciones que han sido afectadas por lesiones, enfermedades o por procesos operatorios.

Existen diversos modelos de robots paralelos de rehabilitación los cuales pueden variar en su estructura según la cadena cinemática dispuesta. Esta cadena depende del tipo de articulaciones que posea el robot, las cuales pueden ser de rotación (R), prismáticas (P), esféricas (S) y universales (U). También pueden variar en función de la articulación humana a la cual asisten, pues hay que adaptar la estructura del robot a las partes del cuerpo en las cuales se va a anclar.

En el grupo de robots de rehabilitación de cuello hay un artículo (Lingampally & Selvakumar, 2019) que presentó un robot portátil. El robot posee tres brazos, de los cuales cada uno está conformado por una cadena cinemática RPS. Los actuadores están conectados a las articulaciones rotacionales de cada brazo. Este dispositivo también es capaz de medir la electromiografía de la superficie de los músculos del cuello. Estas medidas pueden proporcionar información acerca de la caída de la cabeza y la progresión del trastorno.

En el grupo de la muñeca el artículo (Wang & Xu, 2021) introdujo un robot paralelo blando que emplea músculos neumáticos artificiales para la rehabilitación de la misma. El robot usa una estructura de 6 brazos SPS que ofrece una opción barata, adaptable y segura. Los músculos artificiales y unos actuadores lineales mueven el robot. Por último,

un sensor electromiográfico otorga información para evaluar el progreso de la rehabilitación.

Para la rehabilitación de la cadera el artículo (Ren, Liu, Luo, & Chen, 2019) muestra un diseño de un exoesqueleto portátil para las piernas. Este robot tiene una forma antropomórfica y una estructura serie-paralelo para replicar mejor los movimientos naturales de las extremidades. Este modelo reduce los errores de coincidencias cinemáticas en múltiples direcciones que suelen tener los modelos de tipo exoesqueleto. Para mayor movilidad de la pelvis se usó en la parte paralela de la estructura un mecanismo 6-SPS.

Por último, el artículo (Hunt & Lee, 2021) expone el robot para el cual ha sido diseñado el sistema de control de este trabajo. Es un robot paralelo cuya estructura empleada para este estudio se centra en rehabilitar el hombro. Posee tres brazos de 4 barras con dos actuadores cada uno (4B-SPM). Los brazos se conectan a una plataforma móvil triangular (el efector final), la cual se mueve de forma tangente al espacio de trabajo esférico que indica la Figura 1. Puede usarse en otras articulaciones como el tobillo, la cadera y la muñeca gracias a su flexibilidad a la hora de posicionar los actuadores. También destacan el uso de componentes mecánicos sencillos, un diseño económico, que no requiere el uso de la articulación humana asistida para ninguna solución cinemática y que permite regular su rigidez para optimizar actividades como levantar peso o absorber impactos. Esto último es gracias a un método de diseño llamado acoplamiento modular de movimiento (MMC), sobre el cual se hizo un análisis de rigidez con matrices estructurales usadas en ingeniería civil. Con un modelo de rigidez y por las características previamente mencionadas, el robot 4B-SPM podría tener amplias aplicaciones en exoesqueletos.

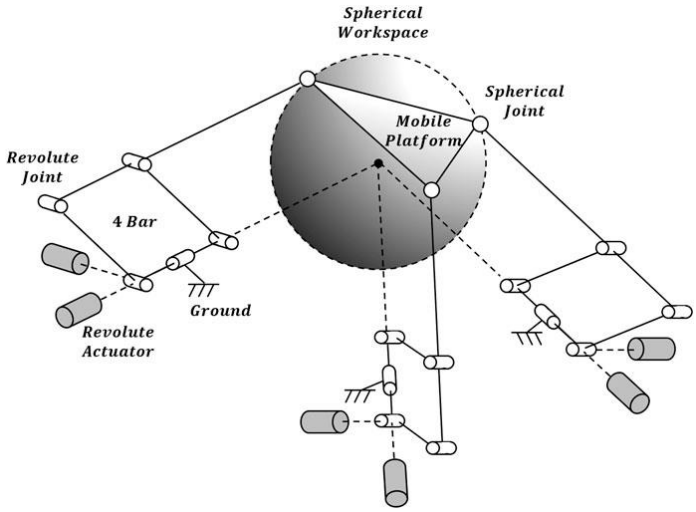

Figura 1. 4B-SPM

Controlar y minimizar los errores de posición puede optimizar la calidad del tratamiento, minimizar los efectos de las perturbaciones que entran al sistema e incluso puede evitar lesiones producidas por un mal posicionamiento del efector. Por eso es necesario diseñar un sistema de control preciso que controle el robot para que las respuestas temporal y permanente del sistema cumplan los parámetros especificados. Los robots en serie, por ejemplo, pueden usar un sistema de control desacoplado con un regulador PID para cada actuador. En cambio, los robots paralelos requieren un método de diseño distinto, ya que no se pueden controlar con precisión con el método anterior debido al acoplamiento dinámico que existe entre distintas cadenas cinemáticas. Por lo tanto, en este trabajo se propone el estudio del control en el espacio de estados, que permite tener en cuenta el acoplamiento dinámico de varios grados de libertad.

1.2. OBJETIVOS Y APORTACIÓN DE ESTE TRABAJO

Este TFG plantea el estudio del robot paralelo 4B-SPM y el estudio y diseño de un sistema de control en el espacio de estado para dicho dispositivo. Para ello se ha usado los contenidos impartidos en la asignatura Control en el Espacio de Estado, Teoría de Sistemas, Sistemas de control y los programas Matlab y Simulink.

Con respecto a las aportaciones, este trabajo pretende conseguir un sistema de control funcional para el robot. A lo largo del trabajo estarán desarrolladas distintas simulaciones con un modelo lineal y con un modelo no-lineal (más realista) del robot para demostrar la fiabilidad del sistema.

Cabe destacar que el sistema se ha diseñado con linealizaciones de las ecuaciones cinemáticas y dinámicas, por lo que el robot controlado por el mismo tendría que funcionar en torno a un punto de trabajo. También se ha reducido el número de actuadores a un motor por brazo para simplificar el cálculo.

1.3.ESTRUCTURA DE LA MEMORIA

La estructura de la memoria tendrá los siguientes apartados:

- **Capítulo 1. Introducción:** Describe el campo de la rehabilitación y algunos modelos de robots paralelos. También expone el robot seleccionado para el trabajo
- **Capítulo 2. Descripción cinemática y dinámica del robot:** Muestra los esquemas, las ecuaciones cinemáticas, dinámicas y las restricciones del robot.
- **Capítulo 3. Descripción de cálculos del control en el espacio de estados:** Comprende los cálculos realizados para el diseño del sistema de control. El cálculo está descrito de forma progresiva.
- **Capítulo 4. Aplicación del sistema al modelo no-lineal:** Muestra las pruebas realizadas del sistema calculado con una simulación de un modelo del robot no-lineal. Las comprobaciones se harán tanto en rangos de operación cercanos como lejanos al punto de trabajo establecido.
- **Capítulo 5. Conclusiones y trabajos futuros:** Se concluye la efectividad del sistema diseñado. También muestra el posible desarrollo posterior que se podría realizar.
- **Capítulo 6. Anexos:** Expone el código y los grupos de ecuaciones asociados a diferentes capítulos del trabajo.
- **Capítulo 7. Bibliografía:** Se exponen los artículos científicos y los libros de la asignatura usados.

2. DESCRIPCIÓN CINEMÁTICA Y DINÁMICA DEL ROBOT

2.1. SÍMBOLOS Y VARIABLES CINEMÁTICAS EMPLEADAS

El desarrollo de las ecuaciones cinemáticas y dinámicas requiere la identificación clara del robot mediante símbolos. Este apartado muestra y explica los símbolos empleados para describir los eslabones, los movimientos de las articulaciones y los puntos usados para describir las posiciones en el espacio del robot. Como los tres brazos son idénticos entre sí, sólo es necesario describir un brazo. Cabe destacar que los brazos están dispuestos formando un círculo dejando una separación de 120º entre cada brazo y que todas las medidas están representadas en unidades del Sistema Métrico Decimal. En nuestro caso, hemos simplificado el robot (mostrado en color rojo) para suprimir los paralelogramos articulados del robot original y sustituirlos por las barras simples comprendidas entre los puntos A y B de cada brazo.

El brazo 1 (el que posee las articulaciones A_1, B_1 y C_1) posee tres eslabones, de los cuales dos son móviles y uno es fijo. Los dos eslabones móviles están representados en la Figura 2 con líneas rojas, mientras que el eslabón fijo está representado por una línea azul. La línea azul no muestra la forma real del eslabón al que está asociado; representa la distancia entre el centro de la esfera y la articulación A_1. La esfera del dibujo muestra el espacio de trabajo del robot. La articulación C_1 se mueve de forma tangente a la superficie de esta esfera. La articulación A_1 es de tipo universal, la B_1 de tipo rotacional y la C_1 es esférica. θ_1, η_1 y ψ_1 representan los ángulos girados por cada articulación. Como la articulación A_1 es de tipo universal los ángulos θ_1 y ψ_1 representan los dos tipos de giro que esta unión puede realizar. De los tres giros, θ_1 es el único giro activo controlado por un motor eléctrico. θ_1 es un giro realizado en torno al eje X. ψ_1 y η_1 giran en torno a vectores con la misma dirección que el eje Y. La razón por la que ψ_1 indica el giro de una articulación

del robot sin simplificar es porque esta articulación ayuda a visualizar mejor el movimiento.

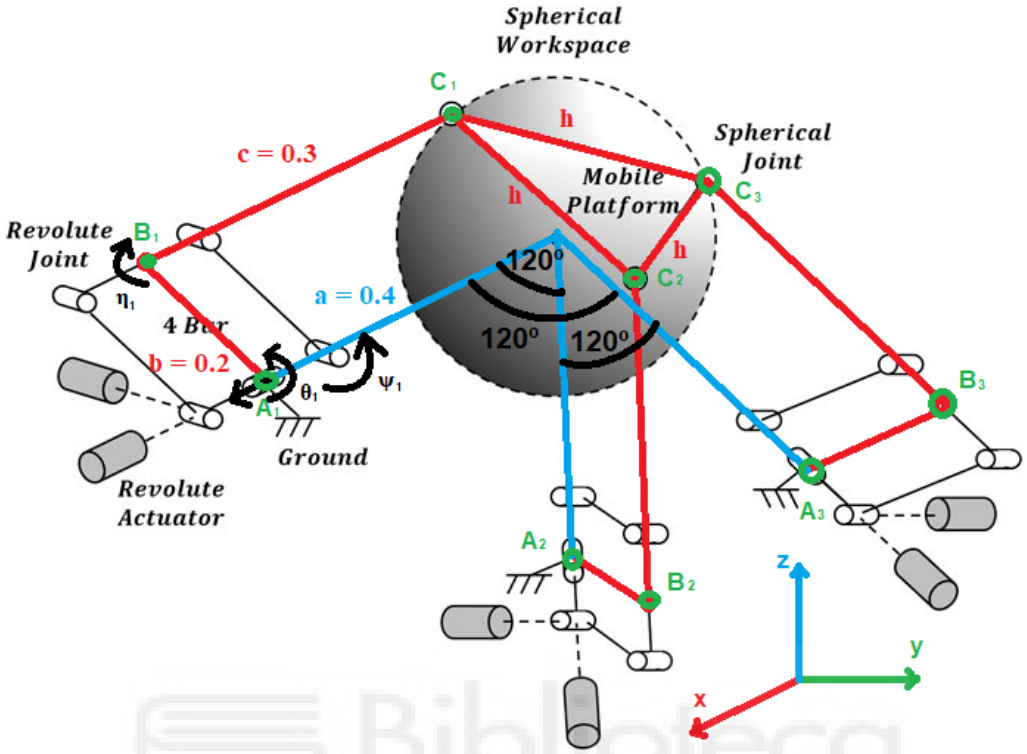

Figura 2. Dibujo del robot simplificado

2.2.RESTRICCIONES CINEMÁTICAS APLICADAS AL ROBOT

En esta sección se trata de establecer las restricciones que existen entre todos los ángulos del robot. Como el robot tiene tres grados de libertad y nueve ángulos de giro, tiene que poseer seis restricciones. Estas restricciones son las que establecen las distancias entre $C_1 = (C_{1x}, C_{1y}, C_{1z})$, $C_2 = (C_{2x}, C_{2y}, C_{2z})$ y $C_3 = (C_{3x}, C_{3y}, C_{3z})$ y las distancias de estas articulaciones al origen de coordenadas, el cual se ha ha posicionado en el centro de la esfera que encierra el espacio de trabajo. Las expresiones que describen estas expresiones son:

$$h^2 = (C_{1x} - C_{2x})(C_{1x} - C_{2x}) + (C_{1y} - C_{2y})(C_{1y} - C_{2y}) + (C_{1z} - C_{2z})(C_{1z} - C_{2z})$$

$$h^2 = (C_{3x} - C_{2x})(C_{3x} - C_{2x}) + (C_{3y} - C_{2y})(C_{3y} - C_{2y}) + (C_{3z} - C_{2z})(C_{3z} - C_{2z})$$

$$h^2 = (C_{3x} - C_{1x})(C_{3x} - C_{1x}) + (C_{3y} - C_{1y})(C_{3y} - C_{1y}) + (C_{3z} - C_{1z})(C_{3z} - C_{1z})$$

$$r^2 = {C_{1x}}^2 + {C_{1y}}^2 + {C_{1z}}^2$$

$$r^2 = {C_{2x}}^2 + {C_{2y}}^2 + {C_{2z}}^2$$

$$r^2 = {C_{3x}}^2 + {C_{3y}}^2 + {C_{3z}}^2$$

Donde r es el radio de la esfera del espacio de trabajo. Las expresiones que permiten obtener las coordenadas C_1, C_2 y C_3 están descritas en el Anexo I, donde se puede ver que C_1 depende de θ_1, η_1 y ψ_1 (ocurre de forma análoga con C_2 y C_3 con los ángulos asociados a las distintas patas).

Las tres primeras ecuaciones descritas arriba son las que imponen que la distancia entre las tres articulaciones esféricas sea h. Las ecuaciones restantes imponen que la distancia desde cualquiera de las articulaciones C hasta el origen sea igual a r, lo que implica que el efector está unido al origen mediante una articulación esférica (no se muestra en las figuras por simplicidad). En el Anexo I estas expresiones son planteadas como un sistema de ecuaciones $F(Q) = 0$, donde Q es el vector que contiene los nueve ángulos de giro del motor. Para resolver la cinemática directa del robot hay que despejar los seis ángulos pasivos (η_1, η_2, η_3 y ψ_1, ψ_2, ψ_3) del sistema de ecuaciones ideado cuando se conocen los ángulos activos θ_1, θ_2 y θ_3 (movidos por motores).

2.3. MODELADO DINÁMICO DEL ROBOT

Para obtener el modelo dinámico del dispositivo se ha empleado el procedimiento descrito en el artículo (Peidró, Quijada-Fernández, Úbeda, Puerto, Payá, & Reinoso, 2022, February). Para empezar, se han descrito dos grupos de variables cinemáticas:

- Entradas cinemáticas: $q = [\theta_1, \theta_2, \theta_3]^T$
- Salidas cinemáticas: $x = [\psi_1, \eta_1, \psi_2, \eta_2, \psi_3, \eta_3]^T$

Estas entradas y salidas están relacionadas por un sistema de ecuaciones que se denominará como:

$$C(q, x) = [C_1(q, x), \dots, C_6(q, x)]^T$$

Donde C_i son las restricciones aplicadas al sistema. En este caso se han empleado las restricciones del apartado anterior. Derivar $C(q, x)$ con respecto al tiempo genera una relación entre las velocidades de las entradas y las salidas:

$$A\dot{x} + B\dot{q} = 0$$

Donde A y B son matrices Jacobianas de las cuales se pueden definir dos tipos principales de singularidades:

- La matriz A contiene las derivadas de C_i con respecto a x_j ($\frac{\partial C_i}{\partial x_j}$). Las posiciones para las cuales $\det(A) = 0$ son singularidades de *Tipo II*. Este tipo de singularidades hace que el efector final alcance velocidades incontrolables distintas de cero en el efector final, incluso cuando las entradas son nulas.
- La matriz B contiene las derivadas de C_i con respecto a q_j ($\frac{\partial C_i}{\partial q_j}$). Las posiciones en las que $\det(B) = 0$ son singularidades de *Tipo I*. Estas singularidades producen una pérdida de la destreza del robot, lo que implica que algunas velocidades de salida son inalcanzables. Esto ocurre por lo general en los límites del espacio de trabajo del robot. Como normalmente los robots no trabajan cerca de estos límites en la práctica, se puede asumir que $\det(B) \neq 0$. Entonces $\dot{q}$ puede ser despejado de la siguiente forma:

$$\dot{q} = B^{-1}A\dot{x}$$

Si se deriva $C(q, x)$ dos veces se puede obtener una relación entre las aceleraciones de las entradas y las salidas:

$$\dot{A}\dot{x} + A\ddot{x} + \dot{B}\dot{q} + B\ddot{q} = 0$$

A partir de esta expresión se puede despejar $\ddot{q}$:

$$\ddot{q} = -B^{-1}(\dot{A}\dot{x} + A\ddot{x} + \dot{B}\dot{q})$$

Ahora el procedimiento sigue a partir de la siguiente expresión de la mecánica Lagrangiana: por cada coordenada p_i del sistema corresponde una ecuación dinámica como la descrita a continuación:

$$f_{p_i} = \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{p}_i}\right) - \frac{\partial \mathcal{L}}{\partial p_i} - \sum_{j=1}^{n} \lambda_j \frac{\partial C_j}{\partial p_i}$$

Donde $n = 3$ es el número de grados de libertad, $\mathcal{L}$ es el Lagrangiano, que es equivalente a la energía cinética del sistema menos su energía potencial, f_{p_i} es la fuerza generalizada asociada a la coordenada p_i y λ_j son los multiplicadores de Lagrange que afectan a las fuerzas restrictivas originadas por las restricciones de $C(q, x)$. Particularizando esta expresión para todas las entradas cinemáticas nos deja con:

$$\tau = \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}, \frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}, \frac{\partial \mathcal{L}}{\partial \dot{\theta}_3}\right]^T - b - B^T\lambda$$

Donde $\lambda = [\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6]^T$, $b = \left[\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}, \frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}, \frac{\partial \mathcal{L}}{\partial \dot{\theta}_3}\right]^T$ y $\tau = \left[f_{\theta_1}, f_{\theta_2}, f_{\theta_3}\right]^T$ es el vector de las fuerzas de actuación. Si particularizamos la expresión de f_{p_i} para todas las salidas cinemáticas la expresión se queda como sigue:

$$\eta = \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\psi}_1}, \frac{\partial \mathcal{L}}{\partial \dot{\eta}_1}, \frac{\partial \mathcal{L}}{\partial \dot{\psi}_2}, \frac{\partial \mathcal{L}}{\partial \dot{\eta}_2}, \frac{\partial \mathcal{L}}{\partial \dot{\psi}_3}, \frac{\partial \mathcal{L}}{\partial \dot{\eta}_3}\right]^T - d - A^T\lambda$$

Donde $d = \left[\frac{\partial \mathcal{L}}{\partial \dot{\psi}_1}, \frac{\partial \mathcal{L}}{\partial \dot{\eta}_1}, \frac{\partial \mathcal{L}}{\partial \dot{\psi}_2}, \frac{\partial \mathcal{L}}{\partial \dot{\eta}_2}, \frac{\partial \mathcal{L}}{\partial \dot{\psi}_3}, \frac{\partial \mathcal{L}}{\partial \dot{\eta}_3}\right]^T$ y $\eta = \left[f_{\psi_1}, f_{\eta_1}, f_{\psi_2}, f_{\eta_2}, f_{\psi_3}, f_{\eta_3}\right]^T$ es el vector de las fuerzas externas aplicadas sobre las direcciones de las coordenadas de salida.

En general, el Lagrangiano $\mathcal{L}$ dependerá de (q, x), y de sus derivadas $(\dot{q}, \dot{x})$. Teniendo esto en cuenta y usando la regla de la cadena para calcular la derivada con respecto al tiempo en τ Y η se obtienen las siguientes expresiones:

$$\tau = a_q\dot{q} + a_{\dot{q}}\ddot{q} + a_x\dot{x} + a_{\dot{x}}\ddot{x} - b - B^T\lambda$$

$$\eta = c_q\dot{q} + c_{\dot{q}}\ddot{q} + c_x\dot{x} + c_{\dot{x}}\ddot{x} - d - A^T\lambda$$

Donde:

- a_q, $a_{\dot{q}}$, a_x y $a_{\dot{x}}$ son las matrices Jacobianas de $\left[\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}, \frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}, \frac{\partial \mathcal{L}}{\partial \dot{\theta}_3}\right]^T$ con respecto a q, $\dot{q}$, x y $\dot{x}$, respectivamente.
- c_q, $c_{\dot{q}}$, c_x y $c_{\dot{x}}$ son las matrices Jacobianas de $\left[\frac{\partial \mathcal{L}}{\partial \dot{\psi}_1}, \frac{\partial \mathcal{L}}{\partial \dot{\eta}_1}, \frac{\partial \mathcal{L}}{\partial \dot{\psi}_2}, \frac{\partial \mathcal{L}}{\partial \dot{\eta}_2}, \frac{\partial \mathcal{L}}{\partial \dot{\psi}_3}, \frac{\partial \mathcal{L}}{\partial \dot{\eta}_3}\right]^T$ con respecto a q, $\dot{q}$, x y $\dot{x}$, respectivamente.

En el Anexo II se recopilan diversas rutinas que permiten calcular las matrices Jacobianas descritas arriba para modelar y simular la dinámica no-lineal del robot. Además, esas mismas rutinas se encargan de obtener las matrices del modelo de estado linealizado en torno a un punto de trabajo. Son estas ecuaciones las que se utilizarán para diseñar los controladores del siguiente capítulo.

3. ESTUDIO Y DISEÑO DEL CONTROL EN EL ESPACIO DE ESTADOS

Después de haber obtenido la representación interna gracias a las ecuaciones cinemáticas y dinámicas se puede proceder al diseño del sistema de control. El cálculo del sistema está dividido en cuatro partes, que consisten en cuatro sistemas ordenados de menor a mayor complejidad. Cada uno de estos controladores están diseñados para los siguientes sistemas:

- **S**istema SISO con dos estados
- Sistema SISO con dos estados con parte integral
- Sistema MIMO con seis estados críticamente amortiguado
- Sistema MIMO con seis estados subamortiguado

Para el estudio del control en el espacio de estado primero se ha trabajado con un modelo simplificado del sistema en el que sólo actúa un brazo del robot, de forma que los motores de los otros dos brazos se mantienen fijos (θ_2 y θ_3 están fijos). El sistema posterior agrega el controlador integral, que ayuda a reducir las perturbaciones y las oscilaciones no deseadas que se puedan producir en la entrada. Los dos últimos controladores estudian el control para sistemas MIMO y establecen los cálculos de diseño para los sistemas finales, los cuales se aplican al robot completo con sus tres brazos en movimiento. Los últimos sistemas tienen un esquema de control similar al sistema SISO con parte integral, pero el cálculo usado es distinto para cada caso. Es por ello por lo que se han separado en estos grupos.

Además, para el estudio de cada sistema SISO se ha tenido en cuenta la controlabilidad. Es necesario determinar este factor para determinar si el sistema es capaz de alcanzar cualquiera de los valores deseados del estado desde una posición inicial nula en un periodo finito de tiempo. Los sistemas LTI continuos son controlables si el rango de la matriz de controlabilidad es igual a n (n es el orden del sistema):

$$Q = [B\ AB\ A^2B\ \dots\ A^{n-1}B]$$

En esta expresión Q es la matriz de controlabilidad y A y B son las matrices de estado y entrada del modelo dinámico linealizado del robot. No se deben confundir con las matrices A y B del capítulo anterior, las cuales definían los dos tipos de singularidades de los robots paralelos.

El valor que un estado puede alcanzar desde un estado inicial nulo es una combinación lineal de las columnas de la matriz Q. Es necesario que haya n columnas linealmente independientes en la matriz Q para alcanzar cualquier valor en el espacio de estado.

A continuación se muestra el desglose de los cálculos junto con una descripción de los mismos para cada parte.

3.1. SISTEMA SISO CON DOS ESTADOS

Para el cálculo del sistema de control se ha usado el modelo simplificado SISO de la representación interna del sistema MIMO (robot completo). En este caso el número de actuadores se reduce a un motor, el cual es el que produce el giro θ_1. Los actuadores de los dos brazos restantes se mantienen fijos, por lo que los giros θ_2 y θ_3 son fijos también. Estas condiciones resultan en un sistema con un grado de libertad y, por tanto, con dos estados (posición y velocidad). Los valores de las matrices usadas para el cálculo de este sistema simplificado son los siguientes:

$$A = \begin{pmatrix} 0 & 1 \\ 0 & -0.025 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 \\ 12 \end{pmatrix}, \qquad C = (1 \quad 0), \quad D = (0)$$

Con estos valores la representación interna del sistema se puede expresar de la siguiente forma:

$$\frac{d}{dt}\begin{pmatrix} \Delta\theta_1 \\ \Delta\dot{\theta}_1 \end{pmatrix} = A \cdot \begin{pmatrix} \Delta\theta_1 \\ \Delta\dot{\theta}_1 \end{pmatrix} + B \cdot \Delta\tau_1$$

$$\Delta\theta_1 = C \cdot \begin{pmatrix} \Delta\theta_1 \\ \Delta\dot{\theta}_1 \end{pmatrix} + D \cdot \Delta\tau_1$$

Donde $\Delta\theta_1$ y $\Delta\dot{\theta}_1$ son el incremento del ángulo girado y de la velocidad angular del actuador, respectivamente, y $\Delta\tau_1$ es la variación de la fuerza de actuación del motor. Como hay dos variables de estado, el sistema es de orden $n = 2$.

El control de este sistema consiste en una realimentación de los estados con una matriz de ganancias K. Para calcular esta realimentación hay que tener en cuenta que la matriz A cambia, siendo la nueva matriz resultante $A_r = A - BK$. Con esta matriz se puede proceder al cálculo del polinomio característico (denominador de la FDT) y de la función de transferencia, con la cual se podrán asignar los polos deseados. Pero antes hay que llevar a cabo una transformación de las matrices A, B, C y D a lo que se conoce como Forma Controlable Canónica (CCF).

Para transformar las matrices a su CCF hay que calcular primero la matriz de controlabilidad $Q = [B\ AB]$. Esta matriz es de rango 2 (como indican los valores que se muestran más abajo), por lo que el sistema es controlable. A continuación se calcula Q^{-1} y se utiliza la última fila de esta matriz (la cual se denominará como e_n^T, **n = 2**) para calcular la matriz de transformación inversa $T_C^{-1} = \begin{pmatrix} e_2^T \\ e_2^T \cdot A \end{pmatrix}$. Los valores obtenidos son los siguientes:

$$Q = \begin{pmatrix} 0 & 12 \\ 12 & -0.3 \end{pmatrix}, \quad Q^{-1} = \begin{pmatrix} \frac{1}{480} & \frac{1}{12} \\ \frac{1}{12} & 0 \end{pmatrix}, \quad T_C^{-1} = \begin{pmatrix} \frac{1}{12} & 0 \\ 0 & \frac{1}{12} \end{pmatrix}, \quad T_C = \begin{pmatrix} 12 & 0 \\ 0 & 12 \end{pmatrix}$$

Con las matrices de transformación calculadas se puede obtener la CCF de cualquier representación inicial del sistema. Las matrices transformadas en este caso van a ser la matriz A ($\tilde{A} = T_C^{-1} \cdot A \cdot T_C$), la matriz B ($\tilde{B} = T_C^{-1} \cdot B$) y la matriz C ($\tilde{C} = C \cdot T_C$). Después se calcula $\tilde{A}_r = \check{A} - \tilde{B}\widetilde{K_C}$. Esta matriz es necesaria para el cálculo del polinomio característico y de la función de transferencia, pero aquí solo se muestra la FDT ($G_R(s)$) por simplicidad.

$$G_R(s) = \tilde{C}[s \cdot I - \tilde{A}_r]^{-1}\tilde{B}$$

Con la FDT obtenida se puede observar que es, en efecto, un sistema de segundo orden. Si igualamos esta función de transferencia a la expresión general de las FDT de segundo orden y despejamos algunas incógnitas, teniendo en cuenta que el sistema va a esta críticamente amortiguado, se pueden despejar los dos elementos de la matriz de realimentación transformada $\widetilde{K_C} = [\widetilde{K_1}\ \widetilde{K_2}]$.

$$G_R(s) = \frac{12}{s^2 + \left(\widetilde{K_2} + 0.025\right)s + \widetilde{K_1}} = \frac{k \cdot \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

$$\zeta = 1, \qquad \omega_n = \frac{4.73}{t_s}$$

Los resultados obtenidos están en función del tiempo de establecimiento t_s y son los siguientes:

$$G_R(s) = \frac{12}{s^2 + \frac{9.46}{t_s}s + \frac{22.3729}{t_s^2}}$$

$$\widetilde{K_1} = \frac{22.3729}{t_s^2},\ \widetilde{K_2} = \frac{9.46}{t_s} - 0.025$$

Con este proceso hemos conseguido colocar los polos en el lugar deseado en función del tiempo de establecimiento. Sin embargo, el numerador de la función de transferencia no se ha modificado con el proceso. Esto puede afectar a la ganancia estática y, por tanto, a la respuesta en estado permanente del sistema. Para resolver esto se calcula una ganancia $K_S = \frac{1}{k}$ que ajuste el valor de la entrada del sistema. El valor obtenido para esta ganancia en función del tiempo de establecimiento es $\boldsymbol{K_S = \frac{22.3729}{12t_s^2}}$. La forma del esquema en Matlab es la representada en la Figura 3.

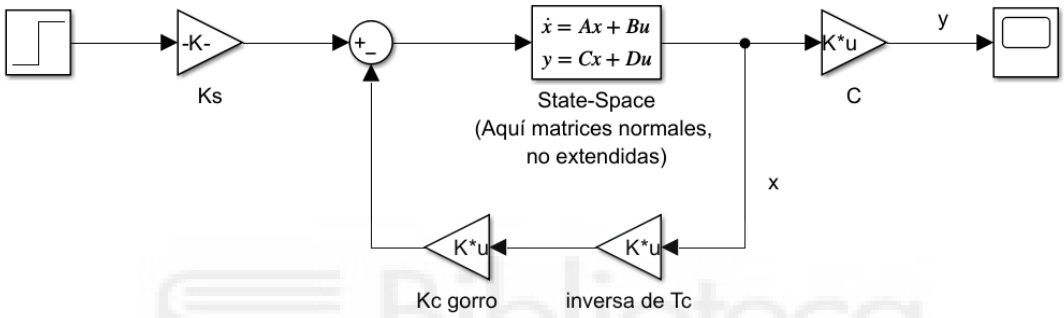

Figura 3. Sistema SISO con dos estados

En el bloque de espacio de estado se ha dado a la matriz C el valor de la matriz identidad para extraer los estados del sistema, transformarlos a CCF y realimentarlos con el ajuste de la ganancia $\widetilde{K_C}$. Posteriormente, estos estados se han multiplicado por la matriz C real para obtener la salida real del sistema. Los resultados obtenidos para una ganancia estática unitaria del sistema, para una entrada escalón unitaria y para un tiempo de establecimiento $t_S = 1$ son:

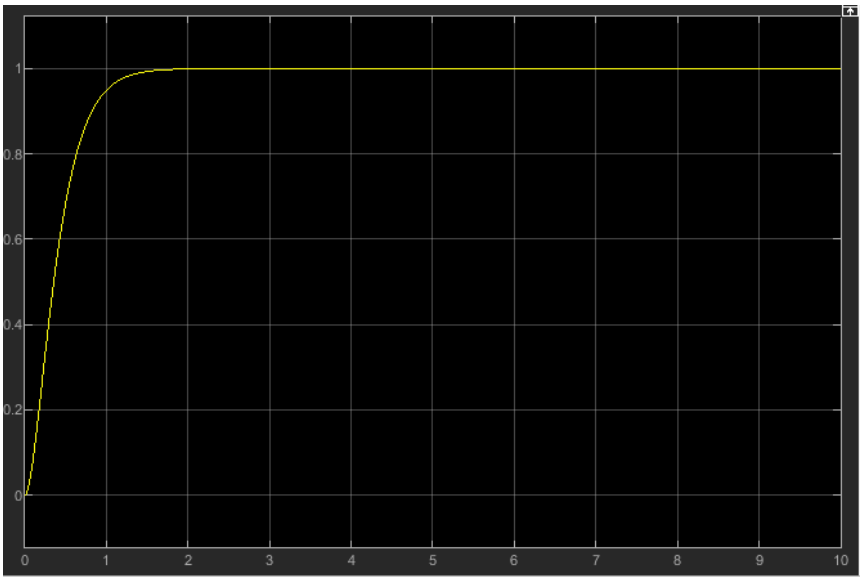

Figura 4. Salida del sistema SISO 1

La siguiente gráfica muestra los resultados para una ganancia estática k = 2, un tiempo de establecimiento $t_S = 2.5$s y para una entrada escalón unitaria:

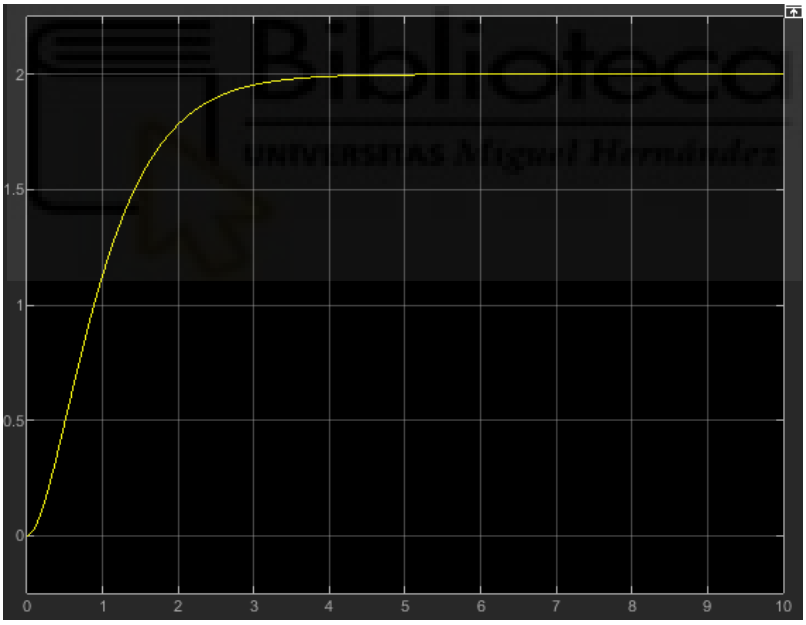

Figura 5. Salida del sistema SISO 2

3.2.SISTEMA SISO CON DOS ESTADOS Y PARTE INTEGRAL

El controlador integral se añadió teniendo en cuenta uno de los defectos del sistema anterior. La ganancia K_S que se usó para ajustar la señal de entrada del sistema también afecta a las perturbaciones y errores de la misma señal. Es por ello por lo que se decidió implementar el controlador integral, el cual sí que es un método robusto ante estas

condiciones. Este controlador se implementa junto a una segunda realimentación de los estados del sistema.

Para el cálculo de la ganancia del controlador integral K_I y de la ganancia de realimentación K_C hay que obtener la función de transferencia y el polinomio característico. La obtención del polinomio característico sirve en este apartado (y en el anterior) para hacer comprobaciones comparándolo con el denominador de la FDT. Para conseguir el polinomio característico, en vez de pasar las matrices a CCF, primero hay que generar una ecuación de estado del sistema aumentada. Las matrices usadas correspondientes a la ecuación de estado (A, B, C, D) son las mismas que en el apartado anterior.

La forma aumentada de la ecuación de estado es la que describe la Figura 7. Esta forma corresponde al esquema mostrado en la Figura 6.

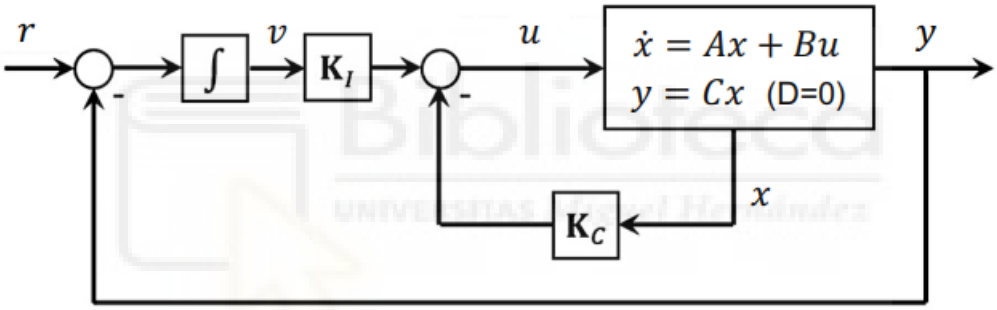

Figura 6. Esquema del sistema SISO realimentado con controlador integral

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix}}_{\dot{\hat{x}}} = \underbrace{\begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix}}_{\hat{A}} \underbrace{\begin{bmatrix} x \\ v \end{bmatrix}}_{\hat{x}} + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{\hat{B}} u + \begin{bmatrix} 0 \\ I \end{bmatrix} r$$

Figura 7. Ecuación de estado en su forma aumentada

Los valores de las matrices aumentadas obtenidos son los siguientes:

$$\hat{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -0.025 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \qquad \hat{B} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Teniendo en cuenta que la señal de control $u = -\widehat{K_C} \cdot \hat{x}$, la ecuación se puede reducir a la siguiente expresión: $\dot{\hat{x}} = (\hat{A} - \hat{B}\widehat{K_C})\hat{x} + \begin{bmatrix}0\\I\end{bmatrix} r$, siendo $(\hat{A} - \hat{B}\widehat{K_C}) = \widehat{A_r}$. Será con ésta última matriz con la que se calculará el polinomio característico. El proceso para el cálculo se realizará con las siguientes ecuaciones:

$$\widehat{K_C} = [K_C \quad -K_i] = [K_1 \quad K_2 \quad -K_i], \qquad P_R(s) = |s \cdot I - \widehat{A_r}|$$

El polinomio característico obtenido es: $s^3 + (12K_2 + 0.025)s^2 + 12K_1 s - 12K_i$. El polinomio indica que el sistema ahora tiene 3 polos. Como solo se necesitan dos polos para calcular un sistema críticamente amortiguado se puede cancelar uno de esos polos con uno de los ceros del sistema, en el caso de que los tuviera. Para comprobar esto, se calcula la función de transferencia del sistema mediante la siguiente expresión:

$$G_R(s) = C(s^2 I + s(BK_C - A) + BK_i C)^{-1} BK_i$$

Obteniéndose así la siguiente expresión:

$$G_R(s) = \frac{12K_i}{s^3 + (12K_2 + 0.025)s^2 + 12K_1 s + 12K_i}$$

El sistema no posee ceros en su función de transferencia, por lo que se ha optado por despreciar uno de los polos asignándole un valor que afecte mínimamente al comportamiento del sistema. Para ello se empleará el método de asignación de polos, con el cual se dará un mismo valor a dos polos y al tercero se le asignará un valor doce veces mayor al anterior. La expresión queda de la siguiente forma:

$$(s + \sigma)^2 (s + d) = s^3 + (12K_2 + 0.025)s^2 + 12K_1 s - 12K_i$$

$$d > 6\sigma \longrightarrow d = 12\sigma$$

Igualando los índices que acompañan a las variables 's' de mismo orden se pueden obtener los valores de K_i, K_1 y K_2 en función del polo σ:

$$K_i = \sigma^3, \qquad K_1 = \frac{25}{12}\sigma^2, \quad K_2 = \frac{14\sigma - 0.025}{12}$$

Por último, se puede despejar σ para que las ganancias queden en función del tiempo de establecimiento en vez de en función de un polo real doble. Para esto, se descarta el tercer polo y el polinomio resultante se iguala al polinomio característico de los sistemas de segundo orden: $s^2 + 2\zeta\omega_n + \omega_n^2 = (s + \sigma)^2$. Teniendo en cuenta que el sistema a

diseñar es críticamente amortiguado ($\zeta = 1$, $t_s = \frac{4.73}{\omega_n}$), se deduce que $\sigma = \omega_n = \frac{4.73}{t_s}$. Despejando los polos con este valor, las ganancias tienen la siguiente forma:

$$K_i = \frac{105.823817}{{t_s}^3}, \qquad K_1 = \frac{559.3225}{12{t_s}^2}, \qquad K_2 = \frac{\frac{66.22}{t_s} - 0.025}{12}$$

Éstos serán los valores usados en las simulaciones lineales en Simulink. El esquema utilizado es el de la Figura 8.

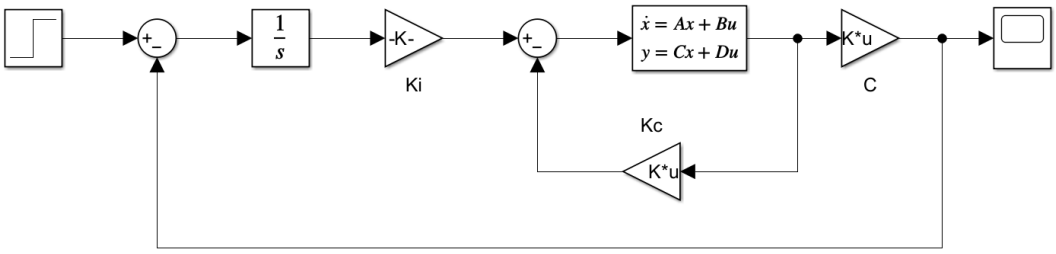

Figura 8. Esquema del sistema SISO con controlador integral

La Figura 9 representa la salida del sistema cuando se le aplica un escalón unitario y cuando el tiempo de establecimiento $t_s = 1s$:

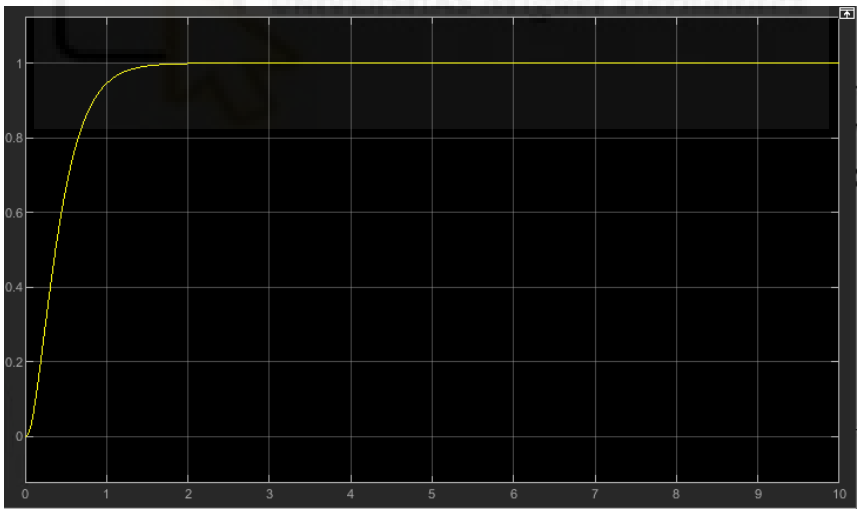

Figura 9. Salida del sistema SISO integral 1

La Figura 10 muestra el resultado obtenido cuando $t_s = 2.5s$ y cuando la entrada es un escalón unitario:

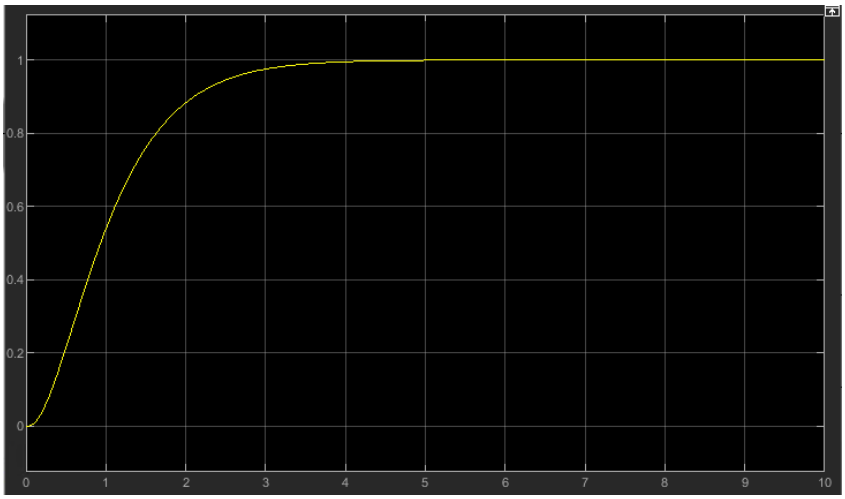

Figura 10. Salida del sistema SISO integral 2

3.3.SISTEMA MIMO CON SEIS ESTADOS CRÍTICAMENTE AMORTIGUADO

Los sistemas diseñados anteriormente han ayudado a estudiar y comprender el control en el espacio de estado. Los métodos utilizados, como la Forma Controlable Canónica y las matrices ampliadas, serán empleados en el cálculo del control para el sistema completo MIMO. Sin embargo, algunos procedimientos dentro del cálculo son distintos dentro de un sistema MIMO. Es por ello que en este apartado se desglosa el cálculo realizado para el diseño de un sistema de control críticamente amortiguado para el robot MIMO completo. Este sistema estará conformado por una realimentación de los estados a la que se le ha aplicado una ganancia y por un controlador integral cuya entrada es una segunda realimentación de la salida. El esquema del sistema es muy similar al de la Figura 6. También cabe destacar que ahora ninguno de los motores actuadores del robot se mantiene fijo, resultando en un sistema con tres grados de libertad. Las matrices utilizadas corresponden al modelo dinámico del robot linealizado en torno a su configuración de referencia ($\theta_1 = 0, \theta_2 = 0, \theta_3 = 0$) y son las expuestas a continuación:

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.025 & -0.0128 & 0.0378 \\ 0 & 0 & 0 & 0.0378 & -0.025 & -0.0128 \\ 0 & 0 & 0 & -0.0128 & 0.0378 & -0.025 \end{pmatrix},$$

$$B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 11.7701 & 2.7854 & 2.7854 \\ 2.7854 & 11.7701 & 2.7854 \\ 2.7854 & 2.7854 & 11.7701 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix},$$

$$D = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Con esta información podemos obtener las matrices ampliadas, las cuales se usarán para obtener la matriz de controlabilidad. La forma de obtener estas matrices es la misma que la vista en la Figura 7. Los valores obtenidos se muestran a continuación.

$$\hat{A} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.025 & -0.0128 & 0.0378 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0378 & -0.025 & -0.0128 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0128 & 0.0378 & -0.025 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\hat{B} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 11.7701 & 2.7854 & 2.7854 \\ 2.7854 & 11.7701 & 2.7854 \\ 2.7854 & 2.7854 & 11.7701 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Para obtener la matriz de controlabilidad $\hat{Q}$ se puede emplear la expresión $\hat{Q} = [\hat{B}\ \hat{A}\hat{B}\ \hat{A}^2\hat{B}\ \ldots\ \hat{A}^{n-1}\hat{B}]$. Sin embargo, Matlab posee una función capaz de realizar este proceso más rápido. La expresión utilizada dentro de Matlab es Q = ctrb (Aa, Ba), siendo Q = $\hat{Q}$, Aa = $\hat{A}$ y Ba = $\hat{B}$. La matriz obtenida es demasiado grande como para ser representada en este documento, por lo que los datos de esta matriz estarán representados en la Figura 11 y en la Figura 12.

9x27 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	0	11.7701	2.7854	2.7854	-0.2247	-0.1152	0.3400	-0.0031	0.0186	-0.0155	9.8174e-04	-0.0010
2	0	0	0	2.7854	11.7701	2.7854	0.3400	-0.2247	-0.1152	-0.0155	-0.0031	0.0186	3.2164e-05	9.8173e-04
3	0	0	0	2.7854	2.7854	11.7701	-0.1152	0.3400	-0.2247	0.0186	-0.0155	-0.0031	-0.0010	3.2166e-05
4	11.7701	2.7854	2.7854	-0.2247	-0.1152	0.3400	-0.0031	0.0186	-0.0155	9.8174e-04	-0.0010	3.2164e-05	-6.3335e-05	1.3986e-05
5	2.7854	11.7701	2.7854	0.3400	-0.2247	-0.1152	-0.0155	-0.0031	0.0186	3.2164e-05	9.8173e-04	-0.0010	4.9350e-05	-6.3335e-05
6	2.7854	2.7854	11.7701	-0.1152	0.3400	-0.2247	0.0186	-0.0155	-0.0031	-0.0010	3.2166e-05	9.8174e-04	1.3985e-05	4.9350e-05
7	0	0	0	0	0	0	-11.7701	-2.7854	-2.7854	0.2247	0.1152	-0.3400	0.0031	-0.0186
8	0	0	0	0	0	0	-2.7854	-11.7701	-2.7854	-0.3400	0.2247	0.1152	0.0155	0.0031
9	0	0	0	0	0	0	-2.7854	-2.7854	-11.7701	0.1152	-0.3400	0.2247	-0.0186	0.0155

Figura 11. Matriz de controlabilidad ampliada (1)

15	16	17	18	19	20	21	22	23	24	25	26	27
3.2164e-05	-6.3335e-05	1.3986e-05	4.9350e-05	1.4804e-06	2.3300e-06	-3.8104e-06	1.0001e-07	-2.2146e-07	1.2145e-07	-1.2439e-08	8.8521e-09	3.5874e-09
-0.0010	4.9350e-05	-6.3335e-05	1.3986e-05	-3.8104e-06	1.4805e-06	2.3300e-06	1.2144e-07	1.0001e-07	-2.2146e-07	3.5874e-09	-1.2439e-08	8.8521e-09
9.8174e-04	1.3985e-05	4.9350e-05	-6.3335e-05	2.3300e-06	-3.8104e-06	1.4804e-06	-2.2146e-07	1.2145e-07	1.0001e-07	8.8520e-09	3.5874e-09	-1.2439e-08
4.9350e-05	1.4804e-06	2.3300e-06	-3.8104e-06	1.0001e-07	-2.2146e-07	1.2145e-07	-1.2439e-08	8.8521e-09	3.5874e-09	6.0010e-10	7.3886e-11	-6.7399e-10
1.3986e-05	-3.8104e-06	1.4805e-06	2.3300e-06	1.2144e-07	1.0001e-07	-2.2146e-07	3.5874e-09	-1.2439e-08	8.8521e-09	-6.7399e-10	6.0010e-10	7.3885e-11
-6.3335e-05	2.3300e-06	-3.8104e-06	1.4804e-06	-2.2146e-07	1.2145e-07	1.0001e-07	8.8520e-09	3.5874e-09	-1.2439e-08	7.3890e-11	-6.7399e-10	6.0010e-10
0.0155	-9.8174e-04	0.0010	-3.2164e-05	6.3335e-05	-1.3986e-05	-4.9350e-05	-1.4804e-06	-2.3300e-06	3.8104e-06	-1.0001e-07	2.2146e-07	-1.2145e-07
-0.0186	-3.2164e-05	-9.8173e-04	0.0010	-4.9350e-05	6.3335e-05	-1.3986e-05	3.8104e-06	-1.4805e-06	-2.3300e-06	-1.2144e-07	-1.0001e-07	2.2146e-07
0.0031	0.0010	-3.2166e-05	-9.8174e-04	-1.3985e-05	-4.9350e-05	6.3335e-05	-2.3300e-06	3.8104e-06	-1.4804e-06	2.2146e-07	-1.2145e-07	-1.0001e-07

Figura 12. Matriz de controlabilidad ampliada (2)

A continuación, se puede construir otra matriz L a partir de columnas linealmente independientes de $\hat{Q}$. Esta matriz representará el número de estados que controlará cada entrada del sistema. El número de columnas a utilizar es igual al número de estados aumentados, siendo esto después de incluir los controladores integrales. Para este caso, el orden del sistema aumentado es $\hat{n} = 9$, por lo que se pueden seleccionar nueve columnas. También hay que tener en cuenta que cada columna está asociada a una entrada distinta. Las columnas de la serie [1, 4, 7, 10, 13, …] corresponden a la entrada 1, las de la serie [2, 5, 8, 11, 14, …] corresponden a la entrada 2, y las columnas de la serie [3, 6, 9, 12, 15, …] se asocian con la entrada 3 (las 3 tres entradas son los tres actuadores). Para agrupar estas columnas dentro de la matriz L hay que seleccionar un número determinado de columnas (menor que el orden del sistema aumentado) asociadas a la entrada 1 y hay que posicionarlas juntas dentro de la matriz L. Después hay que hacer lo mismo con las columnas asociadas a la entrada 2 y posicionar este grupo de columnas detrás del primer grupo de columnas. El proceso de la entrada 2 se repite con la entrada 3. La suma de las columnas seleccionadas no puede superar el orden del sistema ampliado. En este caso para que cada entrada controle el mismo número de estados se van a seleccionar tres columnas por entrada. La siguiente expresión utilizada en Matlab genera la matriz L y muestra las columnas utilizadas: L = Q (:, [1, 4, 7, 2, 5, 8, 3, 6, 9]).

	1	2	3	4	5	6	7	8	9
1	0	11.7701	-0.2247	0	2.7854	-0.1152	0	2.7854	0.3400
2	0	2.7854	0.3400	0	11.7701	-0.2247	0	2.7854	-0.1152
3	0	2.7854	-0.1152	0	2.7854	0.3400	0	11.7701	-0.2247
4	11.7701	-0.2247	-0.0031	2.7854	-0.1152	0.0186	2.7854	0.3400	-0.0155
5	2.7854	0.3400	-0.0155	11.7701	-0.2247	-0.0031	2.7854	-0.1152	0.0186
6	2.7854	-0.1152	0.0186	2.7854	0.3400	-0.0155	11.7701	-0.2247	-0.0031
7	0	0	-11.7701	0	0	-2.7854	0	0	-2.7854
8	0	0	-2.7854	0	0	-11.7701	0	0	-2.7854
9	0	0	-2.7854	0	0	-2.7854	0	0	-11.7701

Figura 13. Valores de la matriz L

Ahora hay que calcular la matriz de transformación ${T_C}^{-1}$. La fórmula para obtener la matriz de transformación que se usará en este apartado está especificada al final de este párrafo. Las filas especificadas en la expresión corresponden a la inversa de la matriz L. Las filas son seleccionadas en función del número de estados que controla cada entrada. Como en este caso cada entrada controla el mismo número de estados, se seleccionan las filas 3, 6 y 9. Si la entrada 1, la entrada 2 y la entrada 3 controlasen cada una cuatro, tres y dos estados, respectivamente, las filas a seleccionar serían la fila 4, la fila 7 y la fila 9.

$$ {T_C}^{-1} = \begin{bmatrix} fila3 \\ fila3 \cdot \hat{A} \\ fila3 \cdot \hat{A}^2 \\ fila6 \\ fila6 \cdot \hat{A} \\ fila6 \cdot \hat{A}^2 \\ fila9 \\ fila9 \cdot \hat{A} \\ fila9 \cdot \hat{A}^2 \end{bmatrix} $$

Con esta matriz podemos pasar las matrices aumentadas a su Forma Canónica Controlable. Las matrices que se transformarán son $\hat{A}$ y $\hat{B}$ mediante las siguientes expresiones.

$$ \tilde{\hat{A}} = {T_C}^{-1} \cdot \hat{A} \cdot T_C, \qquad \tilde{\hat{B}} = {T_C}^{-1} \cdot \hat{B} $$

Siendo sus valores obtenidos los siguientes:

	1	2	3	4	5	6	7	8	9
1	0	1.0000	0	0	-3.3941e-17	0	0	-2.6278e-17	0
2	0	0	1.0000	0	0	-3.3941e-17	0	0	-2.6278e-17
3	0	0	-0.0250	0	0	-0.0128	0	0	0.0378
4	0	1.7844e-17	0	0	1	0	0	1.9673e-17	0
5	0	0	1.7844e-17	0	0	1	0	0	1.9673e-17
6	0	0	0.0378	0	0	-0.0250	0	0	-0.0128
7	0	3.9027e-17	0	0	6.9322e-18	0	0	1	0
8	0	0	3.9027e-17	0	0	6.9322e-18	0	0	1
9	0	0	-0.0128	0	0	0.0378	0	0	-0.0250

Figura 14. Valores de A ampliada en su forma CCF

	1	2	3
1	0	0	0
2	0	0	0
3	1.0000	5.6265e-17	-2.6278e-17
4	0	0	0
5	0	0	0
6	-1.5850e-17	1.0000	1.9673e-17
7	0	0	0
8	0	0	0
9	-2.4607e-18	6.2443e-17	1

Figura 15. Valores de B ampliada en su forma CCF

Después, hay que construir la matriz que contiene los polos deseados en su forma CCF, la cual nos ayudará a extraer los valores de las ganancias. Para los sistemas MIMO hay muchas maneras de construir esta matriz, a la cual denominaremos como $\tilde{\tilde{A}}_R$. Esta matriz tiene el mismo número de filas y columnas que $\tilde{\tilde{A}}$, pero hay dos cambios:

- Existen bloques diagonales dentro de la matriz (submatrices) que, según su orden, afectan al control de cada entrada sobre el sistema. Todo lo que esté fuera de estos bloques es igual a cero.
- En la última fila de cada bloque hay que poner los coeficientes correspondientes al polinomio que contiene los polos deseados.

En este caso se ha seleccionado la opción equilibrada, la cual contiene tres bloques diagonales de orden $n = 3$. También se usará el mismo polinomio deseado en cada bloque, pues cada bloque afecta a un brazo del robot. Como todos los brazos son iguales, los polos deseados también pueden serlo para cada uno. A continuación se muestran el polinomio deseado en función de t_s (es el mismo polinomio deseado que se usa en el apartado anterior) y la forma de la matriz $\tilde{\tilde{A}}_R$:

$$\tilde{\tilde{A}}_R = \begin{pmatrix} 0 & 1 & 0 & & & & & & \\ 0 & 0 & 1 & & 0_{3x3} & & & 0_{3x3} & \\ -\alpha_0 & -\alpha_1 & -\alpha_2 & & & & & & \\ & & & 0 & 1 & 0 & & & \\ & 0_{3x3} & & 0 & 0 & 1 & & 0_{3x3} & \\ & & & -\alpha_0 & -\alpha_1 & -\alpha_2 & & & \\ & & & & & & 0 & 1 & 0 \\ & 0_{3x3} & & & 0_{3x3} & & 0 & 0 & 1 \\ & & & & & & -\alpha_0 & -\alpha_1 & -\alpha_2 \end{pmatrix}$$

$$P_R(s) = s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0 = s^3 + \frac{66.22}{t_s} s^2 + \frac{559.3225}{{t_s}^2} s + \frac{1269.885804}{{t_s}^3}$$

Con esta matriz y con las anteriores podemos concluir con el último paso. Hay que calcular las ganancias a partir de la siguiente fórmula: $\tilde{\tilde{B}} \cdot \tilde{\tilde{K}} = R$, siendo $R = \tilde{\tilde{A}} - \tilde{\tilde{A}}_R$. Para despejar las ganancias es necesario que $\tilde{\tilde{B}}$ sea cuadrada. Como no es el caso, hay que realizar otro procedimiento. En este caso se ha optado por quitar las filas nulas que conforman $\tilde{\tilde{B}}$. Las filas de igual número también serán suprimidas en la matriz R puesto que también son nulas. Como se puede observar en la Figura 15, las filas nulas son la 1, la 2, la 4, la 5, la 7 y la 8, así que esas serán retiradas. Después de despejar $\tilde{\tilde{K}}$, habría que deshacer la CCF, por lo que se usa la siguiente expresión: $\hat{K} = \tilde{\tilde{K}} \cdot {T_C}^{-1}$. Para terminar, se pueden diferenciar las ganancias con esta fórmula: $\hat{K} = [K_{C_{3x6}} \; -K_{i_{3x3}}]$ (estas matrices no serán representadas en este documento, pues su tamaño dificulta su exposición). Con esto ya se puede construir un esquema en Simulink (Figura 16) con el que empezar a simular.

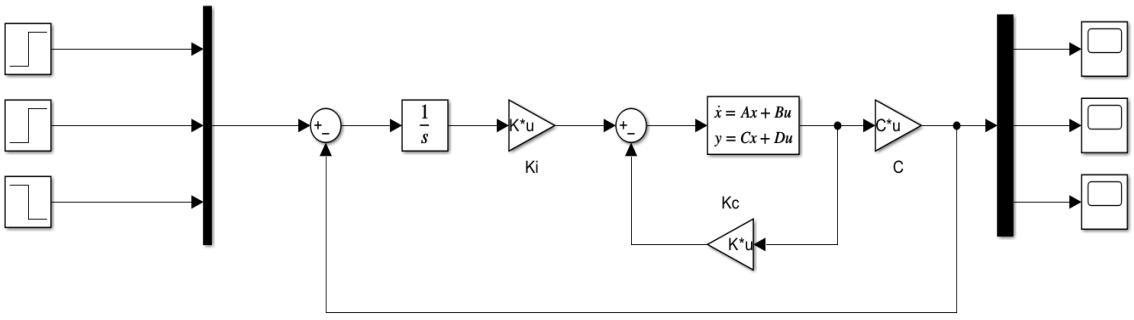

Figura 16. Esquema del sistema MIMO

Los valores de las salidas obtenidos para un tiempo de establecimiento $t_s = 1s$ y para unas entradas escalón $u_1 = 1$, $u_2 = 2$ y $u_3 = -2$ son los siguientes:

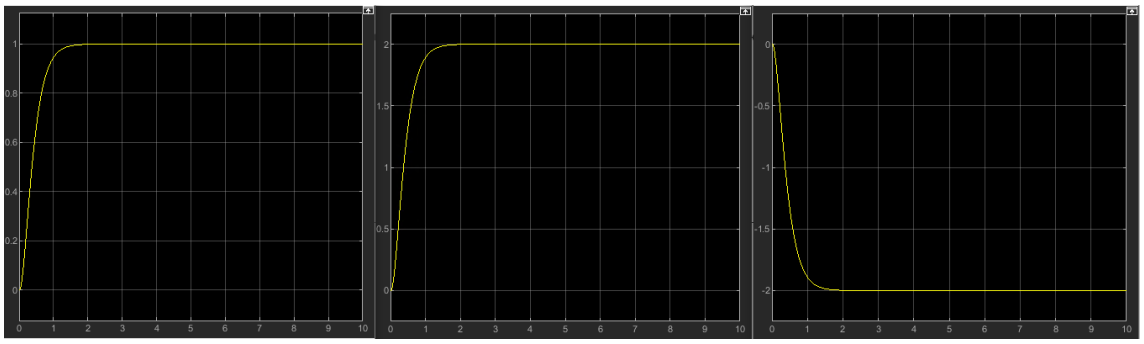

Figura 17. Salidas 1, 2 y 3 del sistema MIMO críticamente amortiguado (1)

Los resultados para un tiempo de establecimiento $t_s = 2.5s$ y para unas entradas escalón $u_1 = 1.5$, $u_2 = 3$ y $u_3 = -1$ son:

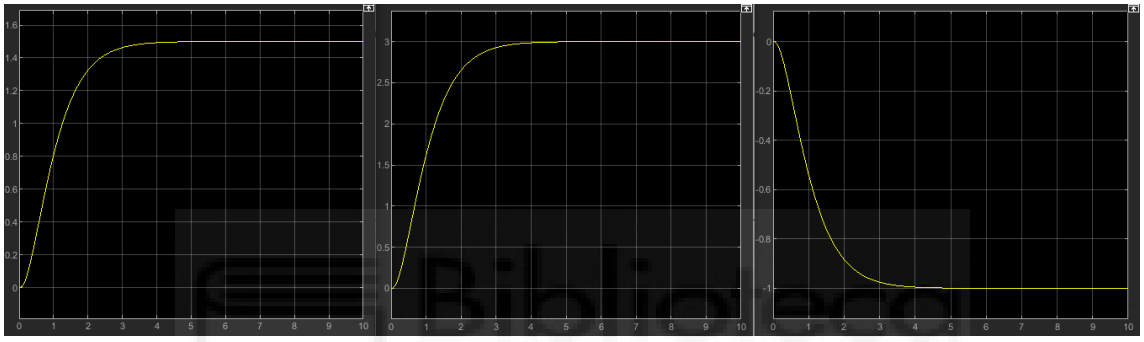

Figura 18. Salidas 1, 2 y 3 del sistema MIMO críticamente amortiguado (2)

3.4. SISTEMA MIMO CON SEIS ESTADOS SUBAMORTIGUADO

Para concluir con el estudio del control en el espacio de estado, también se ha procedido con el diseño de un sistema de control MIMO subamortiguado. El procedimiento de desarrollo de este sistema es el mismo que se ha empleado en el sistema de control MIMO críticamente amortiguado. Pero como los polos deseados en este caso son distintos, hay que diseñar nuevamente una matriz $\tilde{\tilde{A}}_R$. Las matrices $\tilde{\tilde{A}}$ y $\tilde{\tilde{B}}$ contienen los mismos valores que en el apartado anterior y como su proceso de obtención ya ha sido descrito no se desglosará en este apartado.

En este caso también se busca que cada bloque de la matriz $\tilde{\tilde{A}}_R$ contenga los mismos polos y, por tanto, el mismo orden. Para obtener los coeficientes del polinomio característico deseado hay que desarrollar esta función de manera distinta. Se sabe que el orden de este polinomio es $n = 3$, por lo que posee tres polos. Con el método de asignación de polos se asignarán dos polos a dos valores complejos ($s = -\sigma \pm \omega_d j$), mientras que al polo

restante se le asignará un valor que afecte despreciablemente el valor final de las salidas ($s = -d = -12\sigma$). El polinomio resultante es el siguiente:

$$(s + \sigma + \omega_d j)(s + \sigma - \omega_d j)(s + d) \\ = s^3 + (2\sigma + d)s^2 + (\sigma^2 + {\omega_d}^2 + 2\sigma d)s + d(\sigma^2 + {\omega_d}^2)$$

Para una mejor representación se pueden sustituir los polos y dejar el polinomio en función del tiempo de establecimiento y de la sobreoscilación. Para ello, se tienen en cuenta las siguientes igualdades: $t_s = \frac{\pi}{\sigma}$, $M_p = e^{\frac{-\pi}{tg(\theta)}}$, $tg(\theta) = \frac{\omega_d}{\sigma}$. Ahora el polinomio característico deseado se puede expresar de la siguiente forma:

$$P_R(s) = s^3 + \left(\frac{14\pi}{{t_s}^2}\right)s^2 + \left(\frac{25\pi^2 \cdot (\ln(M_p))^2 + \pi^4}{{t_s}^2 \cdot (\ln(M_p))^2}\right)s + \frac{12\pi^3 \cdot (\ln(M_p))^2 + 12\pi^5}{{t_s}^3 \cdot (\ln(M_p))^2}$$

Con los coeficientes de este polinomio se puede construir $\tilde{\tilde{A}}_R$ y se puede llevar a cabo la obtención de la matriz de ganancias $\widetilde{\tilde{K}}$ con la expresión utilizada en el apartado anterior: $\tilde{\tilde{B}} \cdot \widetilde{\tilde{K}} = R$, siendo $R = \tilde{\tilde{A}} - \tilde{\tilde{A}}_R$. Se vuelven a suprimir las filas nulas de $\tilde{\tilde{B}}$, las filas equivalentes en R, se despeja $\widetilde{\tilde{K}}$ y se transforma la matriz mediante la expresión $\widehat{K} = \widetilde{\tilde{K}} \cdot {T_C}^{-1}$. Sabiendo que $\widehat{K} = [K_{C_{3x6}} \ -K_{i_{3x3}}]$ se puede empezar a simular. El esquema usado para simular es el de la Figura 16.

Para unas entradas escalón $u_1 = 1, u_2 = 2$ y $u_3 = -2$, con un tiempo de establecimiento $t_s = 1s$ y con una sobreoscilación $M_p = 0.2$ se obtienen los siguientes valores de las salidas del sistema (Figura 19):

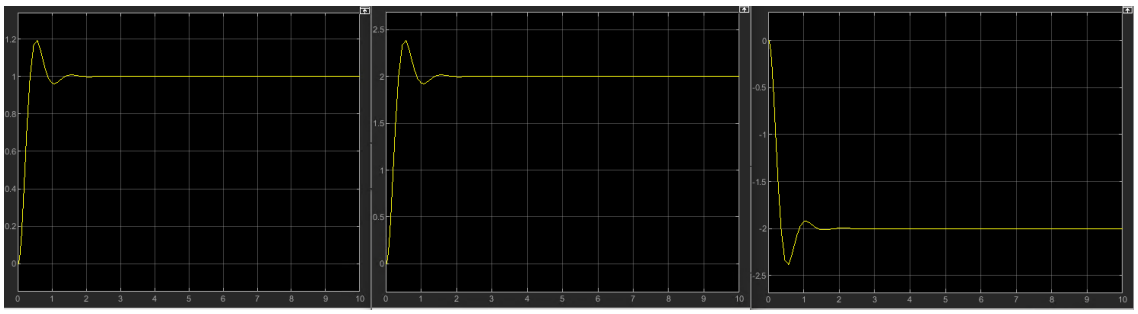

Figura 19. Salidas 1, 2 y 3 del sistema MIMO subamortiguado (1)

Para unas entradas escalón $u_1 = 1.5, u_2 = 3$ y $u_3 = -1$, con un tiempo de establecimiento $t_s = 2.5s$ y con una sobreoscilación $M_p = 0.4$ los resultados obtenidos son los mostrados en la Figura 20:

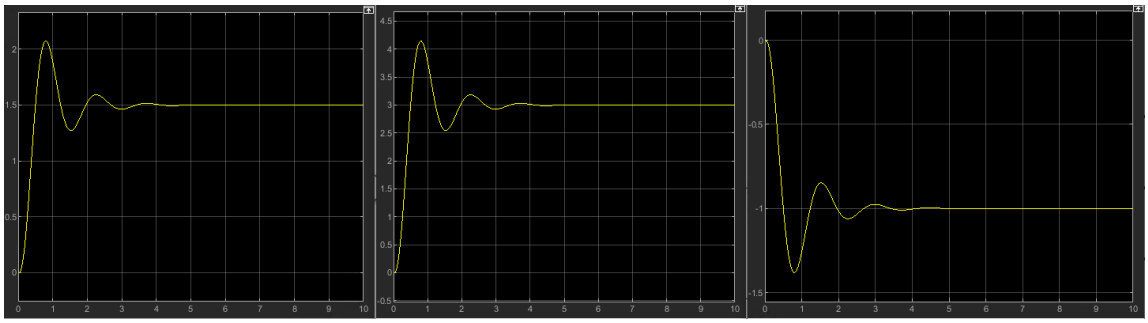

Figura 20. Salidas 1, 2 y 3 del sistema MIMO subamortiguado (2)

Para unas entradas escalón $u_1 = 1$, $u_2 = -1.5$ y $u_3 = -3$, con un tiempo de establecimiento $t_s = 1.5s$ y con una sobreoscilación $M_p = 0.5$ se obtienen los siguientes valores en las salidas (Figura 21):

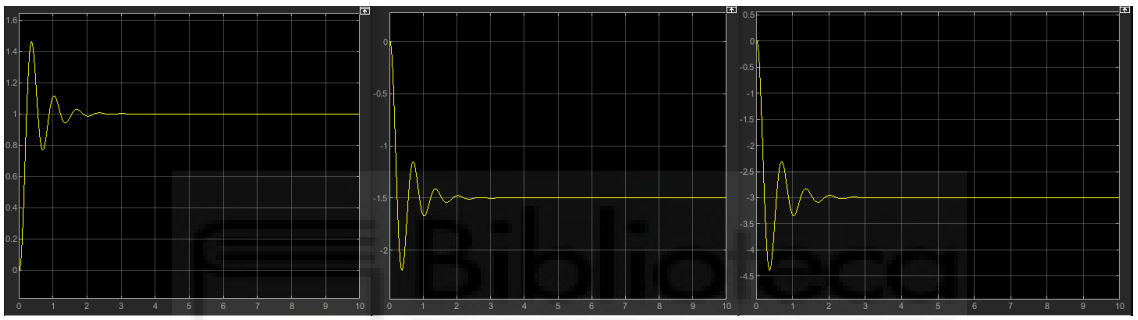

Figura 21. Salidas 1, 2 y 3 del sistema MIMO subamortiguado (3)

4. APLICACIÓN DEL SISTEMA DE CONTROL AL ROBOT NO-LINEAL

Para demostrar el correcto funcionamiento de los dos sistemas de control MIMO diseñados se va a realizar una prueba aplicando estos controladores a un modelo del robot no-lineal. Este modelo tiene un funcionamiento más realista en comparación con las simulaciones realizadas en Simulink, ya que en la práctica los robots paralelos no suelen tener un comportamiento lineal.

Para las simulaciones de este apartado se ha utilizado el programa Easy Java Simulations (EJS) con las siguientes condiciones:

- Las simulaciones del robot parten de la configuración de referencia (analizada en la sección 3) donde todos los ángulos activos (θ_1, θ_2, θ_3) son nulos.
- Todos los brazos del robot son móviles, ya que ninguno de los motores actuadores está fijo. Por lo tanto, el sistema tiene tres grados de libertad (sistema MIMO)
- Las fuerzas gravitatorias que afectan a las masas del robot se desprecian y se supone que estas masas son puntuales.

A continuación se exponen los resultados de las salidas de las cuatro simulaciones realizadas (dos por cada sistema de control).

4.1. EXPERIMENTO 1 CON EL CONTROLADOR CRÍTICAMENTE AMORTIGUADO

Para este caso se ha pedido al robot que sus salidas se desplacen ($\theta_1 = 0.1$, $\theta_2 = 0.1$, $\theta_3 = -0.1$) radianes de su punto de trabajo y se ha establecido el tiempo de establecimiento en $t_s = 2.5$ segundos. Los resultados obtenidos son los de la Figura 22:

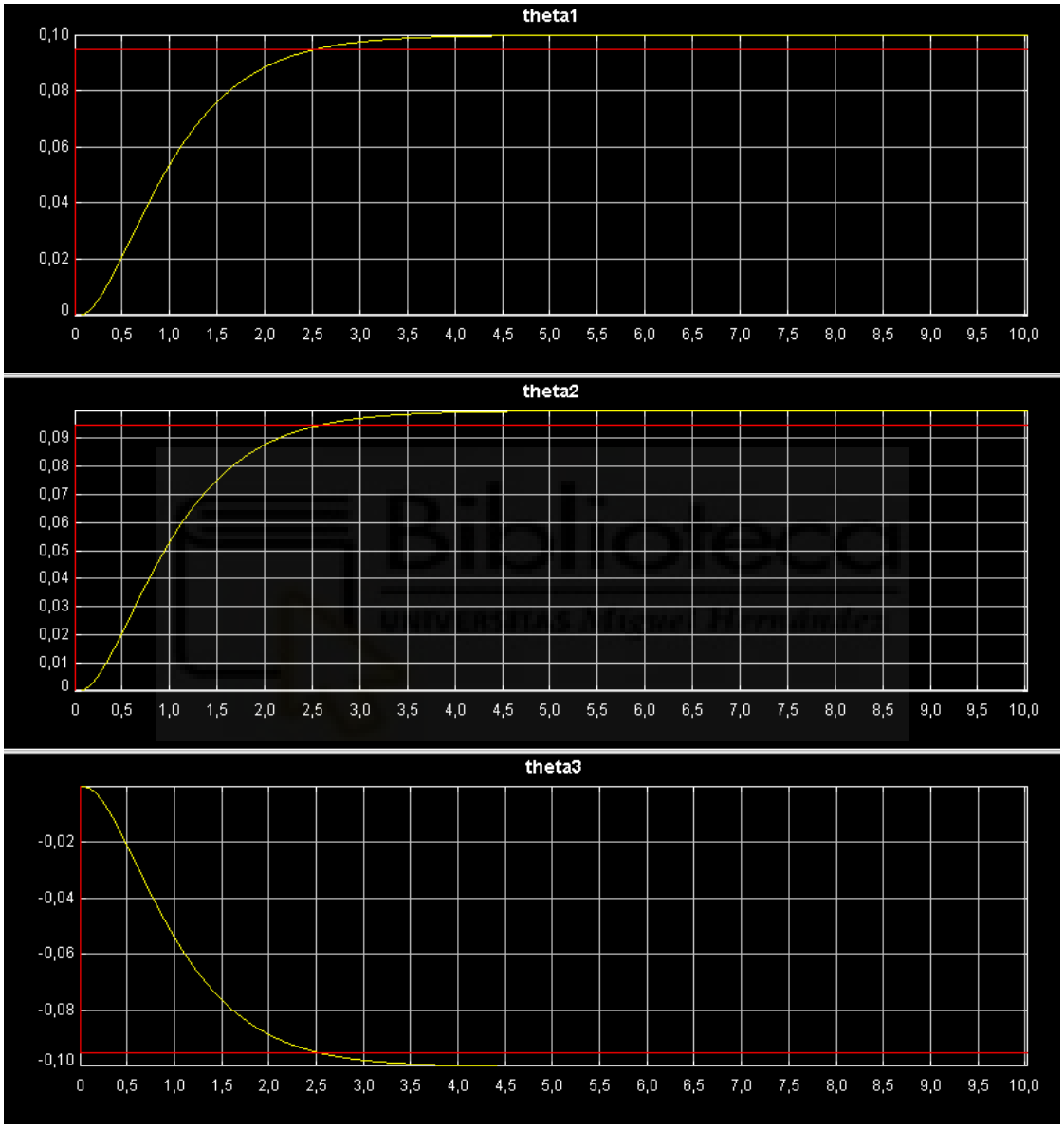

Figura 22. Salidas del sistema MIMO críticamente amortiguado (no-lineal) (1)

Las líneas rojas horizontales muestran la banda de establecimiento cuyos valores corresponden al ±5% del valor final. Se puede comprobar que las salidas alcanzan valores muy cercanos a esta banda en el tiempo de establecimiento fijado. Esto indica que el controlador realiza el comportamiento deseado (es válido). Hay que tener en cuenta que las salidas no se han alejado mucho del punto de trabajo.

4.2. EXPERIMENTO 2 CON EL CONTROLADOR CRÍTICAMENTE AMORTIGUADO

En este apartado el robot desplaza sus salidas ($\theta_1 = 0.7$, $\theta_2 = 0.7$, $\theta_3 = -0.7$) radianes de su punto de trabajo. El tiempo de establecimiento es $t_s = 2.5$ segundos. Las gráficas obtenidas se muestran en la Figura 23:

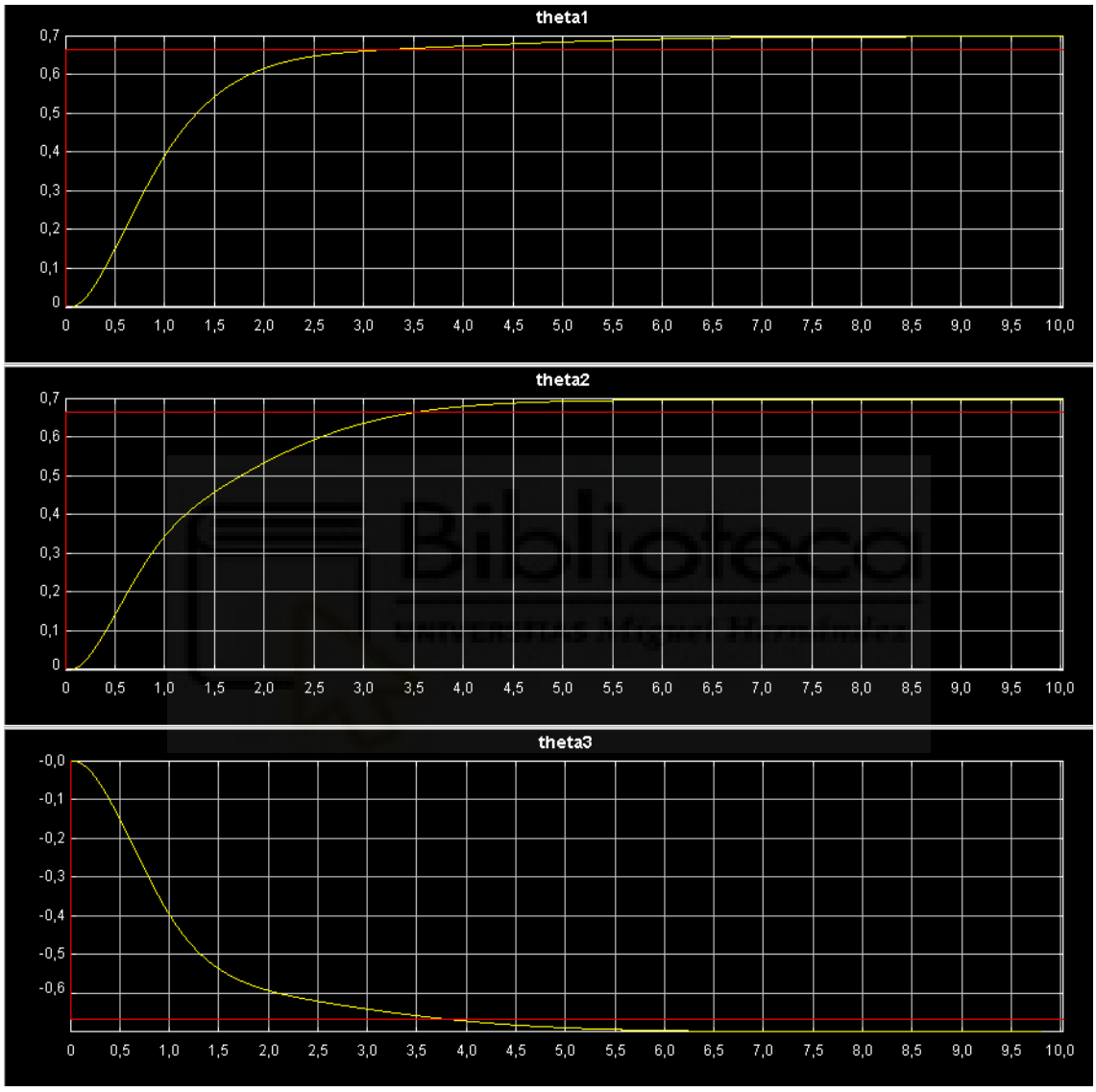

Figura 23. Salidas del sistema MIMO críticamente amortiguado (no-lineal) (2)

Para este caso las salidas no cumplen el comportamiento deseado. Las señales tardan más de 2.5 segundos en atravesar los límites de la banda de establecimiento (por ejemplo θ_2 tarda unos 3.5 segundos en establecerse). Esto se debe a que el controlador, al estar diseñado para un modelo linealizado, es una buena aproximación de la dinámica del robot solamente para pequeñas desviaciones de las salidas en torno al punto de trabajo. El controlador diseñado no es válido para estos valores.

4.3. EXPERIMENTO 3 CON CONTROLADOR SUBAMORTIGUADO

En esta simulación se pide de nuevo que el robot se mueva ($\theta_1 = 0.1$, $\theta_2 = 0.1$, $\theta_3 = -0.1$) radianes de su punto de trabajo. Como el controlador usado es el subamortiguado, las condiciones establecidas restantes son $t_s = 2.5s$ y la sobreoscilación $M_p = 0.4$. Los resultados son los expuestos en la Figura 24:

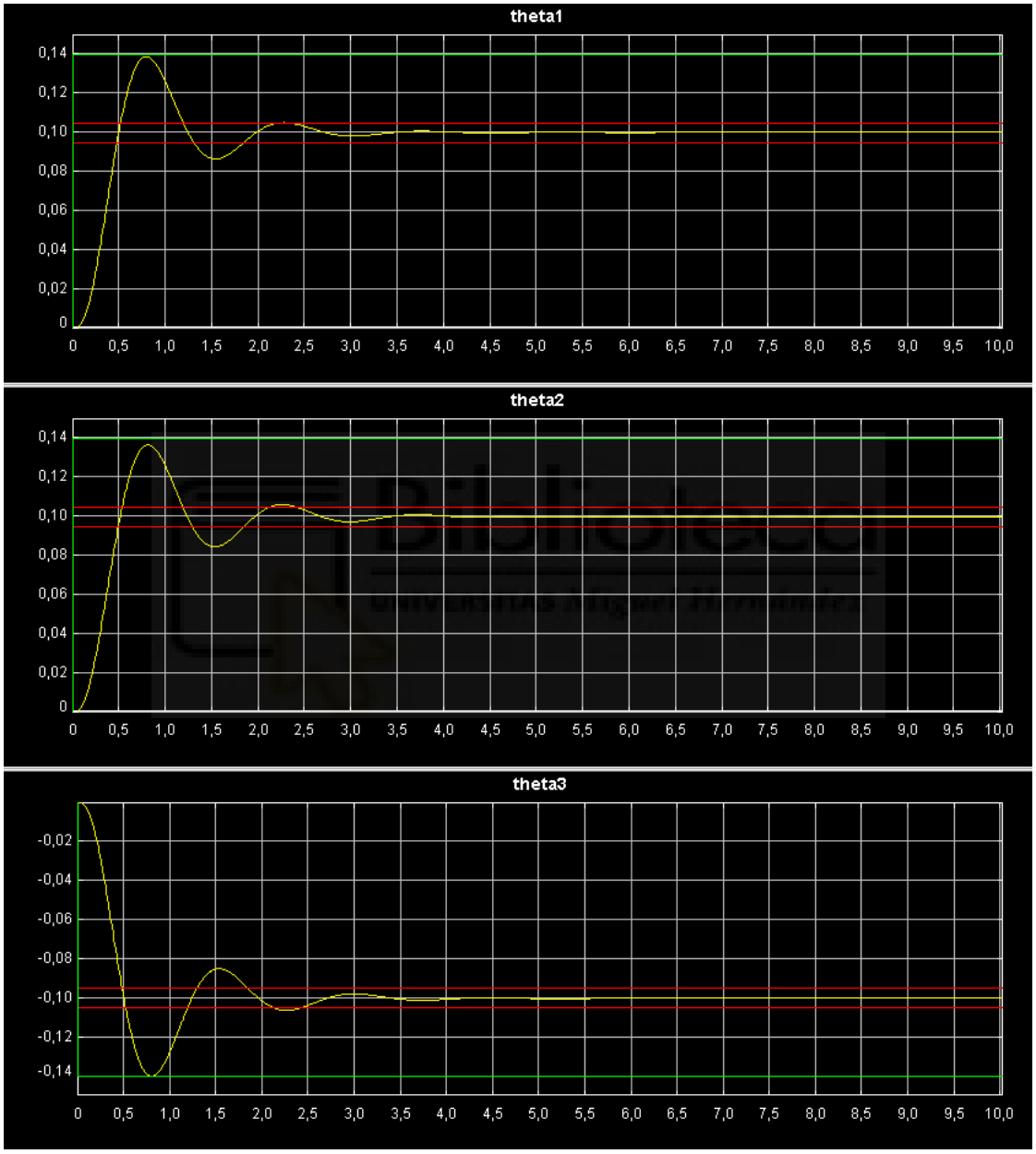

Figura 24. Salidas del sistema MIMO subamortiguado (no-lineal) (2)

Las líneas rojas encierran la banda de establecimiento y la línea verde indica la máxima sobreoscilación. En este caso, el controlador diseñado al alejarse poco del punto de trabajo también alcanza muy aproximadamente el comportamiento deseado. El controlador es válido.

4.4. EXPERIMENTO 4 CON CONTROLADOR SUBAMORTIGUADO

Por último, se pide al robot que desplace sus salidas ($\theta_1 = 0.7$, $\theta_2 = 0.7$, $\theta_3 = -0.7$) radianes de su punto de trabajo. El tiempo de establecimiento y la sobreoscilación tienen el mismo valor que en el apartado anterior ($t_s = 2.5s$, $M_p = 0.4$).

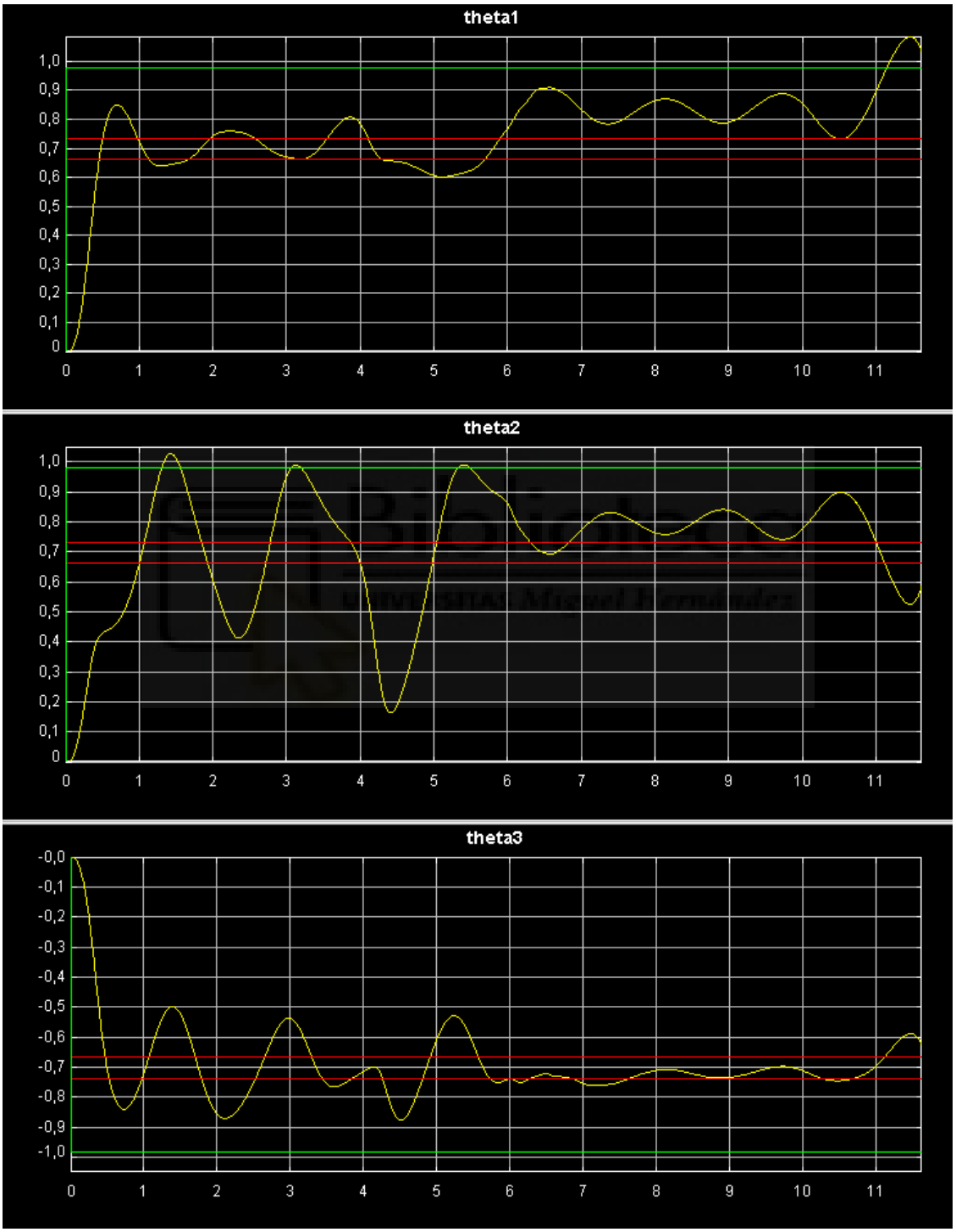

Figura 25. Salidas del sistema MIMO subamortiguado (no-lineal) (2)

En este caso el sistema de control no ha logrado estabilizar las salidas en ningún punto de la simulación. No se logra alcanzar ni el comportamiento transitorio ni el permanente deseado. Ocurre lo mismo que en el apartado 4.2: el robot se aleja demasiado del punto de trabajo fijado. Nótese que, debido a la sobreoscilación, la desviación de las salidas es mayor que la deseada, lo cual contribuye a alejar aún mas al robot del punto de trabajo y a disminuir aún más la validez del controlador si cabe. El controlador no es válido para las salidas solicitadas.

5. CONCLUSIONES Y TRABAJOS FUTUROS

A lo largo del trabajo hemos realizado un estudio y diseño de los sistemas de control en el espacio de estado del robot 4B-SPM. Los objetivos relacionados con este estudio han sido el diseño de sistemas de control funcionales para este robot, el estudio de los usos de la robótica en el campo de la rehabilitación y el desarrollo de las competencias relacionadas con las asignaturas de Teoría de Sistemas, Sistemas de Control y Control en el Espacio de Estado.

Las conclusiones a las que he llegado con este trabajo en base a los resultados obtenidos son los siguientes:

- El diseño de un sistema de control para un robot requiere de los conocimientos de diversos campos de estudio. Por ejemplo, este trabajo ha requerido los conocimientos relacionados con la robótica, la cinemática, la dinámica, la informática y la teoría de sistemas. También hay que tener en cuenta el campo de aplicación, que en este caso es la rehabilitación de articulaciones humanas. Que todos estos detalles influyan a la hora de llevar a cabo el diseño de los sistemas de control del robot ha cambiado mi perspectiva acerca del proceso.
- El modelado de los sistemas de control se aborda de múltiples formas. Para aplicar el control en un sistema podemos emplear controladores integrales, controladores diferenciales, PID, realimentaciones del sistema con ganancias. Sin embargo, el cálculo para el diseño de estos controladores varía en cada caso: el sistema es continuo, discreto, se trabaja en el dominio de la frecuencia, el sistema es SISO, MIMO, etc. En este trabajo hemos empleado el control en el espacio de estado porque funciona bien con los sistemas con varios grados de libertad.
- Los sistemas de control diseñados funcionan correctamente cuando trabajan cerca del punto de trabajo establecido. Esto es debido a que los cálculos se han realizado con un modelo linealizado del robot, el cual funciona con un comportamiento muy aproximado al modelo no-lineal (más realista) cuando el robot no se aleja excesivamente de este punto. Cuando lo aplicamos al modelo no-lineal se puede comprobar con los resultados que los sistemas son válidos cuando se solicita al robot desplazamientos cercanos al punto de trabajo. En otros casos hemos comprobado que el robot no cumple con las condiciones transitorias fijadas e incluso no ha podido mantener unas salidas estables ni ha podido alcanzar un comportamiento permanente.

- He podido comprobar la extensión de los usos de la robótica en campos médicos. Existen muchos dispositivos orientados a rehabilitar y asistir a distintas articulaciones del cuerpo humano. Los artículos relacionados con estos robots también apuntan a estudiar y mejorar los diseños con optimizaciones de los comportamientos relacionados con la rigidez y la docilidad, modificaciones de los robots originales, el uso de materiales con distintas propiedades, etc.

Con respecto a los trabajos futuros, se podrían realizar estudios del control no-lineal de tipo CTC (Computed-Torque Control) para comparar nuevamente los resultados con el modelo linealizado. También se podría variar el diseño cinemático y dinámico del robot para evaluar qué modelo tiene mayor utilidad dentro de la rehabilitación. Incluso se podría realizar un estudio acerca del cambio del control cerca de las singularidades del robot. Pero el que me resultó más llamativo es la construcción de un prototipo real en el que probar los sistemas de control diseñados. Los componentes mecánicos son sencillos, haciendo que el montaje del prototipo resulte más accesible para un Trabajo de Fin de Grado.

En definitiva, los sistemas de control diseñados para el robot 4B-SPM son válidos. Su comportamiento se aproxima significativamente al deseado para las condiciones establecidas, y las simulaciones lo demuestran en los distintos entornos establecidos.

6. ANEXOS

6.1. ANEXO I: ECUACIONES CINEMÁTICAS DEL ROBOT DESCRITAS EN MATLAB

```
function retorno = F(Q)

theta1 = Q(1);
psi1 = Q(2);
eta1 = Q(3);
theta2 = Q(4);
psi2 = Q(5);
eta2 = Q(6);
theta3 = Q(7);
psi3 = Q(8);
eta3 = Q(9);

alpha2 = 0;
alpha3 = 0;
beta2 = 0;
beta3 = 0;

a = 0.4; % distancia origen-A
b = 0.2; % distancia A-B
c = 0.3; % distancia B-C
r = 0.2; % radio esfera
h = 0.25; % lado del efector final triangular (distancia entre Ci y Cj)

C1x = - c*cos(eta1)*cos(psi1) + c*sin(eta1)*sin(psi1) - b*sin(psi1) + a;
C1y = c*cos(eta1)*sin(theta1)*sin(psi1) + c*sin(eta1)*sin(theta1)*cos(psi1) - b*sin(theta1)*cos(psi1);
C1z = - c*cos(eta1)*cos(theta1)*sin(psi1) - c*sin(eta1)*cos(theta1)*cos(psi1) + b*cos(theta1)*cos(psi1);

% Leg 2

A2x = a*sin(alpha2)*cos(beta2);
A2y = a*sin(alpha2)*sin(beta2);
A2z = a*cos(alpha2);

C2x = cos(alpha2)*cos(beta2)*(c*cos(eta2)*cos(theta2)*sin(psi2) ...
    + c*sin(eta2)*cos(theta2)*cos(psi2) - b*cos(theta2)*cos(psi2)) ...
    - sin(alpha2)*cos(beta2)*(c*cos(eta2)*cos(psi2) - c*sin(eta2)*sin(psi2) ...
    + b*sin(psi2)) - sin(beta2)*(c*cos(eta2)*sin(theta2)*sin(psi2) ...
    + c*sin(eta2)*sin(theta2)*cos(psi2) - b*sin(theta2)*cos(psi2)) + A2x;
C2y = cos(alpha2)*sin(beta2)*(c*cos(eta2)*cos(theta2)*sin(psi2) ...
    + c*sin(eta2)*cos(theta2)*cos(psi2) - b*cos(theta2)*cos(psi2)) ...
    - sin(alpha2)*sin(beta2)*(c*cos(eta2)*cos(psi2) - c*sin(eta2)*sin(psi2) ...
    + b*sin(psi2)) + cos(beta2)*(c*cos(eta2)*sin(theta2)*sin(psi2) ...
    + c*sin(eta2)*sin(theta2)*cos(psi2) - b*sin(theta2)*cos(psi2)) + A2y;
C2z = - cos(alpha2)*(c*cos(eta2)*cos(psi2) - c*sin(eta2)*sin(psi2) + b*sin(psi2)) ...
    - sin(alpha2)*(c*cos(eta2)*cos(theta2)*sin(psi2) + c*sin(eta2)*cos(theta2)*cos(psi2) ...
    - b*cos(theta2)*cos(psi2)) + A2z;

% Leg 3

A3x = a*sin(alpha3)*cos(beta3);
A3y = a*sin(alpha3)*sin(beta3);
A3z = a*cos(alpha3);

C3x = cos(alpha3)*cos(beta3)*(c*cos(eta3)*cos(theta3)*sin(psi3) ...
    + c*sin(eta3)*cos(theta3)*cos(psi3) - b*cos(theta3)*cos(psi3)) ...
    - sin(alpha3)*cos(beta3)*(c*cos(eta3)*cos(psi3) - c*sin(eta3)*sin(psi3) ...
    + b*sin(psi3)) - sin(beta3)*(c*cos(eta3)*sin(theta3)*sin(psi3) ...
    + c*sin(eta3)*sin(theta3)*cos(psi3) - b*sin(theta3)*cos(psi3)) + A3x;
C3y = cos(alpha3)*sin(beta3)*(c*cos(eta3)*cos(theta3)*sin(psi3) ...
    + c*sin(eta3)*cos(theta3)*cos(psi3) - b*cos(theta3)*cos(psi3)) ...
    - sin(alpha3)*sin(beta3)*(c*cos(eta3)*cos(psi3) - c*sin(eta3)*sin(psi3) ...
    + b*sin(psi3)) + cos(beta3)*(c*cos(eta3)*sin(theta3)*sin(psi3) ...
    + c*sin(eta3)*sin(theta3)*cos(psi3) - b*sin(theta3)*cos(psi3)) + A3y;
C3z = - cos(alpha3)*(c*cos(eta3)*cos(psi3) - c*sin(eta3)*sin(psi3) + b*sin(psi3)) ...
    - sin(alpha3)*(c*cos(eta3)*cos(theta3)*sin(psi3) + c*sin(eta3)*cos(theta3)*cos(psi3) ...
    - b*cos(theta3)*cos(psi3)) + A3z;

retorno = zeros(6,1);
retorno(1) = (C1x-C2x)*(C1x-C2x) + (C1y-C2y)*(C1y-C2y) + (C1z-C2z)*(C1z-C2z) - h*h;
retorno(2) = (C3x-C2x)*(C3x-C2x) + (C3y-C2y)*(C3y-C2y) + (C3z-C2z)*(C3z-C2z) - h*h;
retorno(3) = (C3x-C1x)*(C3x-C1x) + (C3y-C1y)*(C3y-C1y) + (C3z-C1z)*(C3z-C1z) - h*h;
retorno(4) = C1x*C1x + C1y*C1y + C1z*C1z - r*r;
retorno(5) = C2x*C2x + C2y*C2y + C2z*C2z - r*r;
retorno(6) = C3x*C3x + C3y*C3y + C3z*C3z - r*r;
```

6.2. ANEXO II: MODELO DINÁMICO DEL ROBOT DESCRITO EN JAVA

```
// Notas:
// 1) Las simulaciones de control sobre el robot no-lineal se han realizado
//    en Easy Java Simulations (EJS) para explotar las facilidades que este
//    software gratuito ofrece a efectos de representar fácilmente el robot.
// 2) alpha y beta son coordenadas esféricas de los ejes de rotación de
//    los ángulos theta2 y theta3, porque se ha programado para que
//    funcione con una disposición arbitraria de dichos ejes. Sin embargo,
//    en las pruebas del TFG se han tomado los 3 ejes de (theta1,theta2,theta3)
//    como coplanarios y formando ángulos de 120deg entre ellos. Esto se
//    corresponde con: alpha2=alpha3=90deg, beta2=120deg, beta3=-120deg.
// 3) El modelo dinámico no-lineal considerado desprecia la gravedad y supone
//    que las masas están concentradas en masas puntuales en las articulaciones
//    B1...B3 y C1...C3. Por simplicidad, se consideran masas unitarias.
// 4) Los numerosos cálculos matriciales involucrados en este código se han realizado
//    mediante la librería gratuita JaMa para Java (Java Matrix Package).

Matrix positions(Matrix Q) {
  // Esta función calcula las posiciones de todas la articulaciones B y C
  // del robot, a partir de los todos los ángulos (tanto actuados como pasivos).
  double B1x, B1y, B1z, C1x, C1y, C1z;
  double A2x, A2y, A2z, B2x, B2y, B2z, C2x, C2y, C2z;
  double A3x, A3y, A3z, B3x, B3y, B3z, C3x, C3y, C3z;

  double psi1, eta1, psi2, eta2, psi3, eta3, theta1, theta2, theta3;
  psi1 = Q.get(0,0);
  eta1 = Q.get(1,0);
  psi2 = Q.get(2,0);
  eta2 = Q.get(3,0);
  psi3 = Q.get(4,0);
  eta3 = Q.get(5,0);
  theta1 = Q.get(6,0);
  theta2 = Q.get(7,0);
  theta3 = Q.get(8,0);

  // Leg 1

  C1x = - c*Math.cos(eta1)*Math.cos(psi1) + c*Math.sin(eta1)*Math.sin(psi1) - b*Math.sin(psi1) + a;
  C1y = c*Math.cos(eta1)*Math.sin(theta1)*Math.sin(psi1) +
c*Math.sin(eta1)*Math.sin(theta1)*Math.cos(psi1) - b*Math.sin(theta1)*Math.cos(psi1);
  C1z = - c*Math.cos(eta1)*Math.cos(theta1)*Math.sin(psi1) -
c*Math.sin(eta1)*Math.cos(theta1)*Math.cos(psi1) + b*Math.cos(theta1)*Math.cos(psi1);

  B1x = a - b*Math.sin(psi1);
  B1y = - b*Math.sin(theta1)*Math.cos(psi1);
  B1z = b*Math.cos(theta1)*Math.cos(psi1);

  // Leg 2

  A2x = a*Math.sin(alpha2)*Math.cos(beta2);
  A2y = a*Math.sin(alpha2)*Math.sin(beta2);
  A2z = a*Math.cos(alpha2);

  B2x = - b*Math.cos(alpha2)*Math.cos(beta2)*Math.cos(theta2)*Math.cos(psi2) -
b*Math.sin(alpha2)*Math.cos(beta2)*Math.sin(psi2) + b*Math.sin(beta2)*Math.sin(theta2)*Math.cos(psi2)
+ A2x;
  B2y = - b*Math.cos(alpha2)*Math.sin(beta2)*Math.cos(theta2)*Math.cos(psi2) -
b*Math.sin(alpha2)*Math.sin(beta2)*Math.sin(psi2) - b*Math.cos(beta2)*Math.sin(theta2)*Math.cos(psi2)
+ A2y;
  B2z = - b*Math.cos(alpha2)*Math.sin(psi2) + b*Math.sin(alpha2)*Math.cos(theta2)*Math.cos(psi2) +
A2z;

  C2x = Math.cos(alpha2)*Math.cos(beta2)*(c*Math.cos(eta2)*Math.cos(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.cos(theta2)*Math.cos(psi2) - b*Math.cos(theta2)*Math.cos(psi2)) -
Math.sin(alpha2)*Math.cos(beta2)*(c*Math.cos(eta2)*Math.cos(psi2) - c*Math.sin(eta2)*Math.sin(psi2) +
b*Math.sin(psi2)) - Math.sin(beta2)*(c*Math.cos(eta2)*Math.sin(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.sin(theta2)*Math.cos(psi2) - b*Math.sin(theta2)*Math.cos(psi2)) + A2x;
  C2y = Math.cos(alpha2)*Math.sin(beta2)*(c*Math.cos(eta2)*Math.cos(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.cos(theta2)*Math.cos(psi2) - b*Math.cos(theta2)*Math.cos(psi2)) -
Math.sin(alpha2)*Math.sin(beta2)*(c*Math.cos(eta2)*Math.cos(psi2) - c*Math.sin(eta2)*Math.sin(psi2) +
b*Math.sin(psi2)) + Math.cos(beta2)*(c*Math.cos(eta2)*Math.sin(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.sin(theta2)*Math.cos(psi2) - b*Math.sin(theta2)*Math.cos(psi2)) + A2y;
  C2z = - Math.cos(alpha2)*(c*Math.cos(eta2)*Math.cos(psi2) - c*Math.sin(eta2)*Math.sin(psi2) +
b*Math.sin(psi2)) - Math.sin(alpha2)*(c*Math.cos(eta2)*Math.cos(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.cos(theta2)*Math.cos(psi2) - b*Math.cos(theta2)*Math.cos(psi2)) + A2z;

  // Leg 3

  A3x = a*Math.sin(alpha3)*Math.cos(beta3);
  A3y = a*Math.sin(alpha3)*Math.sin(beta3);
  A3z = a*Math.cos(alpha3);

  B3x = - b*Math.cos(alpha3)*Math.cos(beta3)*Math.cos(theta3)*Math.cos(psi3) -
b*Math.sin(alpha3)*Math.cos(beta3)*Math.sin(psi3) + b*Math.sin(beta3)*Math.sin(theta3)*Math.cos(psi3)
+ A3x;
```

```
  B3y = - b*Math.cos(alpha3)*Math.sin(beta3)*Math.cos(theta3)*Math.cos(psi3) -
b*Math.sin(alpha3)*Math.sin(beta3)*Math.sin(psi3) - b*Math.cos(beta3)*Math.sin(theta3)*Math.cos(psi3)
+ A3y;
  B3z = - b*Math.cos(alpha3)*Math.sin(psi3) + b*Math.sin(alpha3)*Math.cos(theta3)*Math.cos(psi3) +
A3z;

  C3x = Math.cos(alpha3)*Math.cos(beta3)*(c*Math.cos(eta3)*Math.cos(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.cos(theta3)*Math.cos(psi3) - b*Math.cos(theta3)*Math.cos(psi3)) -
Math.sin(alpha3)*Math.cos(beta3)*(c*Math.cos(eta3)*Math.cos(psi3) - c*Math.sin(eta3)*Math.sin(psi3) +
b*Math.sin(psi3)) - Math.sin(beta3)*(c*Math.cos(eta3)*Math.sin(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.sin(theta3)*Math.cos(psi3) - b*Math.sin(theta3)*Math.cos(psi3)) + A3x;
  C3y = Math.cos(alpha3)*Math.sin(beta3)*(c*Math.cos(eta3)*Math.cos(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.cos(theta3)*Math.cos(psi3) - b*Math.cos(theta3)*Math.cos(psi3)) -
Math.sin(alpha3)*Math.sin(beta3)*(c*Math.cos(eta3)*Math.cos(psi3) - c*Math.sin(eta3)*Math.sin(psi3) +
b*Math.sin(psi3)) + Math.cos(beta3)*(c*Math.cos(eta3)*Math.sin(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.sin(theta3)*Math.cos(psi3) - b*Math.sin(theta3)*Math.cos(psi3)) + A3y;
  C3z = - Math.cos(alpha3)*(c*Math.cos(eta3)*Math.cos(psi3) - c*Math.sin(eta3)*Math.sin(psi3) +
b*Math.sin(psi3)) - Math.sin(alpha3)*(c*Math.cos(eta3)*Math.cos(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.cos(theta3)*Math.cos(psi3) - b*Math.cos(theta3)*Math.cos(psi3)) + A3z;

  return new Matrix(new double[][]
{{B1x,B1y,B1z},{B2x,B2y,B2z},{B3x,B3y,B3z},{C1x,C1y,C1z},{C2x,C2y,C2z},{C3x,C3y,C3z}});
}

Matrix velocities(Matrix Q, Matrix Q_d) {
  // Esta función calcula las velocidades de todas la articulaciones B y C
  // del robot, a partir de los todos los ángulos (tanto actuados como pasivos),
  // y de las derivadas de dichos ángulos.
  Matrix retorno = new Matrix(6,3);
  Matrix aux, dQ;
  double step = 0.00001;
  Matrix P0 = positions(Q);

  for(int i=0;i<9;i++){
    dQ = new Matrix(9,1);
    dQ.set(i,0,step);
    aux = positions(Q.plus(dQ)).minus(P0);
    aux = aux.times(Q_d.get(i,0)/step);
    retorno = retorno.plus(aux);
  }

  return retorno;
}

public double L(Matrix Q, Matrix Q_d){
  // Esta función calcula el Lagrangiano como diferencia entre la energía cinética
  // y la energía potencial del robot (aunque despreciamos la gravedad por ahora).
  // Las derivadas parciales del Lagrangiano integran las matrices que constituyen
  // el modelo no-lineal del robot, como así muestra la función "compute_accels".

  Matrix V = velocities(Q,Q_d);
  Matrix v;

  double[] m = new double[] {1,1,1,1,1,1}; // Masas unitarias en B1...B3 y C1...C3

  double K = 0;

  for(int i=0;i<6;i++){
    v = V.getMatrix(i,i,0,2);
    K = K + m[i]*(v.times(v.transpose())).get(0,0);
  }
  K = K*0.5;

  // De momento sin energia potencial (despreciamos gravedad)

  return K;

}

double[] compute_accels(Matrix Q, Matrix Q_d, Matrix tau){
  // Esta es la función principal del cálculo dinámico. Sus cometidos son:
  // - En primer lugar, se calculan las matrices del modelo de estado no-lineal,
  //   siguiendo la notación del artículo (Peidró, Quijada-Fernández et al., 2022).
  // - En segundo lugar, conocidas las matrices del modelo no-lineal, se despejan
  //   las aceleraciones de todos los ángulos (tanto actuados como pasivos),
  //   conociendo el estado actual del robot (Q y su derivada temporal Q_d),
  //   y conociendo también los pares de control aplicados a sus actuadores (tau).
  //   Además de despejar las aceleraciones, también se despejan los multiplicadores
  //   de Lagrange (lambdas), aunque éstos no se usan.
  // - Por último, a partir de las matrices del modelo no-lineal, se estiman las
  //   matrices del modelo de estado linealizado, que son el punto de partida
  //   para el diseño de controladores en el TFG. Estas matrices linealizadas
  //   han sido denotadas en el código como Ass y Bss (ss: state-space).

  double aux;
  double step = 0.00001;
  double step_squared_4 = 4*step*step;
```

```
  Matrix h1 = new Matrix(9,1);
  Matrix h2 = new Matrix(9,1);

  Matrix a_q = new Matrix(3,3);
  for(int i=0;i<3;i++){ // Recorre filas: q1_d ... qn_d (velocidades)
    h1 = new Matrix(9,1);
    h1.set(6+i,0,step);
    for(int j=0;j<3;j++){ // Recorre columnas: q1 ... qn (posiciones)
      h2 = new Matrix(9,1);
      h2.set(6+j,0,step);
      aux = L(Q.plus(h2),Q_d.plus(h1)) - L(Q.plus(h2),Q_d.minus(h1)) - L(Q.minus(h2),Q_d.plus(h1)) +
L(Q.minus(h2),Q_d.minus(h1));
      a_q.set(i,j,aux);
    }
  }
  a_q = a_q.times(1.0/step_squared_4);

  Matrix a_q_d = new Matrix(3,3);
  for(int i=0;i<3;i++){ // Recorre filas: q1_d ... qn_d (velocidades)
    h1 = new Matrix(9,1);
    h1.set(6+i,0,step);
    for(int j=0;j<3;j++){ // Recorre columnas: q1_d ... qn_d (velocidades)
      h2 = new Matrix(9,1);
      h2.set(6+j,0,step);
      aux = L(Q,(Q_d.plus(h1)).plus(h2)) - L(Q,(Q_d.minus(h1)).plus(h2)) -
L(Q,(Q_d.plus(h1)).minus(h2)) + L(Q,(Q_d.minus(h1)).minus(h2));
      a_q_d.set(i,j,aux);
    }
  }
  a_q_d = a_q_d.times(1.0/step_squared_4);

  Matrix a_x = new Matrix(3,6);
  for(int i=0;i<3;i++){ // Recorre filas: q1_d ... qn_d (velocidades)
    h1 = new Matrix(9,1);
    h1.set(6+i,0,step);
    for(int j=0;j<6;j++){ // Recorre columnas: x1 ... xn (posiciones)
      h2 = new Matrix(9,1);
      h2.set(0+j,0,step);
      aux = L(Q.plus(h2),Q_d.plus(h1)) - L(Q.plus(h2),Q_d.minus(h1)) - L(Q.minus(h2),Q_d.plus(h1)) +
L(Q.minus(h2),Q_d.minus(h1));
      a_x.set(i,j,aux);
    }
  }
  a_x = a_x.times(1.0/step_squared_4);

  Matrix a_x_d = new Matrix(3,6);
  for(int i=0;i<3;i++){ // Recorre filas: q1_d ... qn_d (velocidades)
    h1 = new Matrix(9,1);
    h1.set(6+i,0,step);
    for(int j=0;j<6;j++){ // Recorre columnas: x1_d ... xn_d (velocidades)
      h2 = new Matrix(9,1);
      h2.set(0+j,0,step);
      aux = L(Q,(Q_d.plus(h1)).plus(h2)) - L(Q,(Q_d.minus(h1)).plus(h2)) -
L(Q,(Q_d.plus(h1)).minus(h2)) + L(Q,(Q_d.minus(h1)).minus(h2));
      a_x_d.set(i,j,aux);
    }
  }
  a_x_d = a_x_d.times(1.0/step_squared_4);

  Matrix c_q = new Matrix(6,3);
  for(int i=0;i<6;i++){ // Recorre filas: x1_d x2_d ... (velocidades)
    h1 = new Matrix(9,1);
    h1.set(i,0,step);
    for(int j=0;j<3;j++){ // Recorre columnas: q1 q2 ...  (posiciones)
      h2 = new Matrix(9,1);
      h2.set(6+j,0,step);
      aux = L(Q.plus(h2),Q_d.plus(h1)) - L(Q.plus(h2),Q_d.minus(h1)) - L(Q.minus(h2),Q_d.plus(h1)) +
L(Q.minus(h2),Q_d.minus(h1));
      c_q.set(i,j,aux);
    }
  }
  c_q = c_q.times(1.0/step_squared_4);

  Matrix c_q_d = new Matrix(6,3);
  for(int i=0;i<6;i++){ // Recorre filas: x1_d x2_d ... (velocidades)
    h1 = new Matrix(9,1);
    h1.set(i,0,step);
    for(int j=0;j<3;j++){ // Recorre columnas: q1_d q2_d ... (velocidades)
      h2 = new Matrix(9,1);
      h2.set(6+j,0,step);
      aux = L(Q,(Q_d.plus(h1)).plus(h2)) - L(Q,(Q_d.minus(h1)).plus(h2)) -
L(Q,(Q_d.plus(h1)).minus(h2)) + L(Q,(Q_d.minus(h1)).minus(h2));
      c_q_d.set(i,j,aux);
    }
  }
  c_q_d = c_q_d.times(1.0/step_squared_4);

  Matrix c_x = new Matrix(6,6);
```

```
  for(int i=0;i<6;i++){ // Recorre filas: x1_d x2_d ... (velocidades)
    h1 = new Matrix(9,1);
    h1.set(i,0,step);
    for(int j=0;j<6;j++){ // Recorre columnas: x1 x2 ... (posiciones)
      h2 = new Matrix(9,1);
      h2.set(0+j,0,step);
      aux = L(Q.plus(h2),Q_d.plus(h1)) - L(Q.plus(h2),Q_d.minus(h1)) - L(Q.minus(h2),Q_d.plus(h1)) +
L(Q.minus(h2),Q_d.minus(h1));
      c_x.set(i,j,aux);
    }
  }
  c_x = c_x.times(1.0/step_squared_4);

  Matrix c_x_d = new Matrix(6,6);
  for(int i=0;i<6;i++){ // Recorre filas: x1_d x2_d ... (velocidades)
    h1 = new Matrix(9,1);
    h1.set(i,0,step);
    for(int j=0;j<6;j++){ // Recorre columnas: x1_d x2_d ... (velocidades)
      h2 = new Matrix(9,1);
      h2.set(0+j,0,step);
      aux = L(Q,(Q_d.plus(h1)).plus(h2)) - L(Q,(Q_d.minus(h1)).plus(h2)) -
L(Q,(Q_d.plus(h1)).minus(h2)) + L(Q,(Q_d.minus(h1)).minus(h2));
      c_x_d.set(i,j,aux);
    }
  }
  c_x_d = c_x_d.times(1.0/step_squared_4);

  Matrix b = new Matrix(3,1);
  for(int i=0;i<3;i++){ // Recorre filas: q1 q2 ... (posiciones)
    h1 = new Matrix(9,1);
    h1.set(6+i,0,step);
    aux = L(Q.plus(h1),Q_d) - L(Q,Q_d);
    b.set(i,0,aux);
  }
  b = b.times(1.0/step);

  Matrix d = new Matrix(6,1);
  for(int i=0;i<6;i++){ // Recorre filas: x1 x2 ... (posiciones)
    h1 = new Matrix(9,1);
    h1.set(i,0,step);
    aux = L(Q.plus(h1),Q_d) - L(Q,Q_d);
    d.set(i,0,aux);
  }
  d = d.times(1.0/step);

  Matrix d_q_d = new Matrix(6,3);
  for(int i=0;i<6;i++){ // Recorre filas: x1 x2 ... (posiciones)
    h1 = new Matrix(9,1);
    h1.set(i,0,step);
    aux = L(Q.plus(h1),Q_d) - L(Q,Q_d);
    d.set(i,0,aux);
  }
  d = d.times(1.0/step);

  // Las matrices AB a continuación son las Jacobianas que nos permiten
  // definir los dos tipos de singularidades de los robots paralelos, siguiendo
  // la notación empleada en (Peidró, Quijada-Fernández et al., 2022).
  Matrix AB = J_full(Q);
  Matrix A = AB.getMatrix(0,5,0,5);
  Matrix B = AB.getMatrix(0,5,6,8);
  Matrix G = new Matrix(6,3);
  try{
    G = (A.inverse()).times(B); G = G.times(-1);
  }
  catch(Exception e){
    System.out.println("A no invertible: singularidad de tipo paralelo.");
  }
  Matrix GT = G.transpose();
  Matrix N1, N2, N3, N4;
  N1 = a_q; N2 = a_x.times(G); N3 = GT.times(c_q); N4 = GT.times(c_x).times(G);
  Matrix N = N1.minus(N2).minus(N3).plus(N4);
  Matrix M1, M2, M3, M4;
  M1 = a_q_d; M2 = a_x_d.times(G); M3 = GT.times(c_q_d); M4 = GT.times(c_x_d).times(G);
  Matrix M = M1.minus(M2).minus(M3).plus(M4);

  // Seguidamente se reduce el modelo y se extraen las matrices del modelo de estado linealizado.
  Matrix Ass = new Matrix(6,6);
  Ass.setMatrix(0,2,3,5,Matrix.identity(3,3));
  Ass.setMatrix(3,5,3,5, ( (M.inverse()).times(N) ).times(-1) );
  Matrix Bss = new Matrix(6,3);
  Bss.setMatrix(3,5,0,2,M.inverse());
  /*
  // Se printean para anotarlas y proceder a realizar manualmente
  // los cálculos de diseño de reguladores en el TFG.
  Ass.print(7,7);
  Bss.print(7,7);
  */
```

```
  // Lo que queda es el cálculo de aceleraciones del sistema no-lineal,
  // para proceder a su integración numérica y a simular la dinámica no-lineal
  // del robot con los controladores lineales diseñados en el TFG.
  Matrix Mnl = new Matrix(15,15);
  Mnl.setMatrix(0,2,0,2,a_q_d); Mnl.setMatrix(0,2,3,8,a_x_d);
Mnl.setMatrix(0,2,9,14,B.transpose().times(-1));
  Mnl.setMatrix(3,8,0,2,c_q_d); Mnl.setMatrix(3,8,3,8,c_x_d);
Mnl.setMatrix(3,8,9,14,A.transpose().times(-1));
  Mnl.setMatrix(9,14,0,2,B); Mnl.setMatrix(9,14,3,8,A);

  Matrix q_d_col = new Matrix(new double[][] {{Q_d.get(6,0)},{Q_d.get(7,0)},{Q_d.get(8,0)}});
  Matrix x_d_col = new Matrix(new double[][]
{{Q_d.get(0,0)},{Q_d.get(1,0)},{Q_d.get(2,0)},{Q_d.get(3,0)},{Q_d.get(4,0)},{Q_d.get(5,0)}});
  Matrix Nln = new Matrix(15,1);
  Nln.setMatrix(0,2,0,0, tau.minus(a_q.times(q_d_col)).minus(a_x.times(x_d_col)).plus(b) );
  Nln.setMatrix(3,8,0,0, (c_q.times(q_d_col).times(-1)).minus(c_x.times(x_d_col)).plus(d) );
  // Nln.setMatrix(9,14,0,0, -A_d x_d - B_d q_d ); // Por simplicidad, despreciamos este término de
velocidades cuadráticas suponiendo velocidades pequeñas.

  Matrix accels_lambdas = (Mnl.inverse()).times(Nln);

  return accels_lambdas.getColumnPackedCopy();
}

Matrix J_full(Matrix q) {
  // Esta función calcula la Jacobiana completa (full), que es la Jacobiana de todas las
  // restricciones del robot respecto a todos los ángulos (tanto actuados como pasivos).

  Matrix retorno = new Matrix(6,9);

  Matrix F0 = F_full(q);

  double step = 0.00001;
  for(int i=0;i<9;i++){
    Matrix dq = new Matrix(9,1);
    dq.set(i,0,step);
    Matrix F1 = F_full(q.plus(dq));
    retorno.setMatrix(0,5,i,i,F1.minus(F0));
  }

  retorno = retorno.times(1.0/step);

  return retorno;
}

Matrix F_full(Matrix q) {
  // Esta función calcula todas las restricciones del robot.

  double A1x, A1y, A1z, B1x, B1y, B1z, C1x, C1y, C1z;
  double A2x, A2y, A2z, B2x, B2y, B2z, C2x, C2y, C2z;
  double A3x, A3y, A3z, B3x, B3y, B3z, C3x, C3y, C3z;

  double psi1, eta1, psi2, eta2, psi3, eta3, theta1, theta2, theta3;
  psi1 = q.get(0,0);
  eta1 = q.get(1,0);
  psi2 = q.get(2,0);
  eta2 = q.get(3,0);
  psi3 = q.get(4,0);
  eta3 = q.get(5,0);
  theta1 = q.get(6,0);
  theta2 = q.get(7,0);
  theta3 = q.get(8,0);

  // Leg 1

  C1x = - c*Math.cos(eta1)*Math.cos(psi1) + c*Math.sin(eta1)*Math.sin(psi1) - b*Math.sin(psi1) + a;
  C1y = c*Math.cos(eta1)*Math.sin(theta1)*Math.sin(psi1) +
c*Math.sin(eta1)*Math.sin(theta1)*Math.cos(psi1) - b*Math.sin(theta1)*Math.cos(psi1);
  C1z = - c*Math.cos(eta1)*Math.cos(theta1)*Math.sin(psi1) -
c*Math.sin(eta1)*Math.cos(theta1)*Math.cos(psi1) + b*Math.cos(theta1)*Math.cos(psi1);

  B1x = a - b*Math.sin(psi1);
  B1y = - b*Math.sin(theta1)*Math.cos(psi1);
  B1z = b*Math.cos(theta1)*Math.cos(psi1);

  // Leg 2

  A2x = a*Math.sin(alpha2)*Math.cos(beta2);
  A2y = a*Math.sin(alpha2)*Math.sin(beta2);
  A2z = a*Math.cos(alpha2);

  B2x = - b*Math.cos(alpha2)*Math.cos(beta2)*Math.cos(theta2)*Math.cos(psi2) -
b*Math.sin(alpha2)*Math.cos(beta2)*Math.sin(psi2) + b*Math.sin(beta2)*Math.sin(theta2)*Math.cos(psi2)
+ A2x;
  B2y = - b*Math.cos(alpha2)*Math.sin(beta2)*Math.cos(theta2)*Math.cos(psi2) -
b*Math.sin(alpha2)*Math.sin(beta2)*Math.sin(psi2) - b*Math.cos(beta2)*Math.sin(theta2)*Math.cos(psi2)
+ A2y;
```

```
  B2z = - b*Math.cos(alpha2)*Math.sin(psi2) + b*Math.sin(alpha2)*Math.cos(theta2)*Math.cos(psi2) +
A2z;

  C2x = Math.cos(alpha2)*Math.cos(beta2)*(c*Math.cos(eta2)*Math.cos(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.cos(theta2)*Math.cos(psi2) - b*Math.cos(theta2)*Math.cos(psi2)) -
Math.sin(alpha2)*Math.cos(beta2)*(c*Math.cos(eta2)*Math.cos(psi2) - c*Math.sin(eta2)*Math.sin(psi2) +
b*Math.sin(psi2)) - Math.sin(beta2)*(c*Math.cos(eta2)*Math.sin(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.sin(theta2)*Math.cos(psi2) - b*Math.sin(theta2)*Math.cos(psi2)) + A2x;
  C2y = Math.cos(alpha2)*Math.sin(beta2)*(c*Math.cos(eta2)*Math.cos(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.cos(theta2)*Math.cos(psi2) - b*Math.cos(theta2)*Math.cos(psi2)) -
Math.sin(alpha2)*Math.sin(beta2)*(c*Math.cos(eta2)*Math.cos(psi2) - c*Math.sin(eta2)*Math.sin(psi2) +
b*Math.sin(psi2)) + Math.cos(beta2)*(c*Math.cos(eta2)*Math.sin(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.sin(theta2)*Math.cos(psi2) - b*Math.sin(theta2)*Math.cos(psi2)) + A2y;
  C2z = - Math.cos(alpha2)*(c*Math.cos(eta2)*Math.cos(psi2) - c*Math.sin(eta2)*Math.sin(psi2) +
b*Math.sin(psi2)) - Math.sin(alpha2)*(c*Math.cos(eta2)*Math.cos(theta2)*Math.sin(psi2) +
c*Math.sin(eta2)*Math.cos(theta2)*Math.cos(psi2) - b*Math.cos(theta2)*Math.cos(psi2)) + A2z;

  // Leg 3

  A3x = a*Math.sin(alpha3)*Math.cos(beta3);
  A3y = a*Math.sin(alpha3)*Math.sin(beta3);
  A3z = a*Math.cos(alpha3);

  B3x = - b*Math.cos(alpha3)*Math.cos(beta3)*Math.cos(theta3)*Math.cos(psi3) -
b*Math.sin(alpha3)*Math.cos(beta3)*Math.sin(psi3) + b*Math.sin(beta3)*Math.sin(theta3)*Math.cos(psi3)
+ A3x;
  B3y = - b*Math.cos(alpha3)*Math.sin(beta3)*Math.cos(theta3)*Math.cos(psi3) -
b*Math.sin(alpha3)*Math.sin(beta3)*Math.sin(psi3) - b*Math.cos(beta3)*Math.sin(theta3)*Math.cos(psi3)
+ A3y;
  B3z = - b*Math.cos(alpha3)*Math.sin(psi3) + b*Math.sin(alpha3)*Math.cos(theta3)*Math.cos(psi3) +
A3z;

  C3x = Math.cos(alpha3)*Math.cos(beta3)*(c*Math.cos(eta3)*Math.cos(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.cos(theta3)*Math.cos(psi3) - b*Math.cos(theta3)*Math.cos(psi3)) -
Math.sin(alpha3)*Math.cos(beta3)*(c*Math.cos(eta3)*Math.cos(psi3) - c*Math.sin(eta3)*Math.sin(psi3) +
b*Math.sin(psi3)) - Math.sin(beta3)*(c*Math.cos(eta3)*Math.sin(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.sin(theta3)*Math.cos(psi3) - b*Math.sin(theta3)*Math.cos(psi3)) + A3x;
  C3y = Math.cos(alpha3)*Math.sin(beta3)*(c*Math.cos(eta3)*Math.cos(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.cos(theta3)*Math.cos(psi3) - b*Math.cos(theta3)*Math.cos(psi3)) -
Math.sin(alpha3)*Math.sin(beta3)*(c*Math.cos(eta3)*Math.cos(psi3) - c*Math.sin(eta3)*Math.sin(psi3) +
b*Math.sin(psi3)) + Math.cos(beta3)*(c*Math.cos(eta3)*Math.sin(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.sin(theta3)*Math.cos(psi3) - b*Math.sin(theta3)*Math.cos(psi3)) + A3y;
  C3z = - Math.cos(alpha3)*(c*Math.cos(eta3)*Math.cos(psi3) - c*Math.sin(eta3)*Math.sin(psi3) +
b*Math.sin(psi3)) - Math.sin(alpha3)*(c*Math.cos(eta3)*Math.cos(theta3)*Math.sin(psi3) +
c*Math.sin(eta3)*Math.cos(theta3)*Math.cos(psi3) - b*Math.cos(theta3)*Math.cos(psi3)) + A3z;

  Matrix retorno = new Matrix(6,1);
  retorno.set(0,0, (C1x-C2x)*(C1x-C2x) + (C1y-C2y)*(C1y-C2y) + (C1z-C2z)*(C1z-C2z) - h*h );
  retorno.set(1,0, (C3x-C2x)*(C3x-C2x) + (C3y-C2y)*(C3y-C2y) + (C3z-C2z)*(C3z-C2z) - h*h );
  retorno.set(2,0, (C3x-C1x)*(C3x-C1x) + (C3y-C1y)*(C3y-C1y) + (C3z-C1z)*(C3z-C1z) - h*h );
  retorno.set(3,0, C1x*C1x + C1y*C1y + C1z*C1z - r*r );
  retorno.set(4,0, C2x*C2x + C2y*C2y + C2z*C2z - r*r );
  retorno.set(5,0, C3x*C3x + C3y*C3y + C3z*C3z - r*r );

  return retorno;
}
```

7. BIBLIOGRAFÍA

[1] Lingampally, P. K., & Selvakumar, A. A. (2019). Kinematic and Workspace Analysis of a Parallel Rehabilitation Device for Head-Neck Injured Patients. *FME Transactions*, *47*(3).

[2] Wang, Y., & Xu, Q. (2021). Design and testing of a soft parallel robot based on pneumatic artificial muscles for wrist rehabilitation. *Scientific Reports*, *11*(1), 1273.

[3] Ren, B., Liu, J., Luo, X., & Chen, J. (2019). On the kinematic design of anthropomorphic lower limb exoskeletons and their matching movement. *International Journal of Advanced Robotic Systems*, *16*(5), 1729881419875908.

[4] Hunt, J., & Lee, H. (2021). Optimizing the rigid or compliant behavior of a novel parallel-actuated architecture for exoskeleton robot applications. *Frontiers in Robotics and AI*, *8*, 596958.

[5] Peidró, A., Quijada-Fernández, A., Úbeda, D., Puerto, R., Payá, L., & Reinoso, Ó. (2022, February). Imperfect Dynamic Modeling of Parallel Robots Eases the Crossing of Type-II Singularities. In *2022 IEEE 17th International Conference on Advanced Motion Control (AMC)* (pp. 124-131). IEEE.

[6] Dominguez, S., Campoy, P., & Sebastián, J. M. (2000). *Control en el Espacio de Estado*. Sección de Publicaciones de la Escuela Técnica Superior de Ingenieros Industriales, Universidad Politécnica de Madrid.

[7] Ogata, K. (2003). *Ingeniería de control moderna*. Pearson educación.

[8] Kuo, B. C. (1996). *Sistemas de control automático*. Pearson Educación.