UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"Control de un robot redundante en el espacio de estados"

TRABAJO FIN DE GRADO

Diciembre -2025

AUTOR: Jose Daniel Neyra Timoran

DIRECTOR: Adrian Peidro Vidal

INDICE

0. Agradecimiento……1
1. Introducción al estudio……..…2
   1.1 Antecedentes……...2
   1.2 Objetivos y contribución……...……3
   1.3 Estructura de la memoria……4
2. Modelado de la cinemática y dinámica del robot……..6
   2.1 Obtención de la ecuación de estados……..15
3. Diseño de un controlador MIMO……17
   3.1 Sistema discreto equivalente……...17
   3.2 Controlabilidad y observabilidad del sistema discreto equivalente……18
   3.3 Diseño del controlador por retroalimentación e integral……24
      3.3.1 Introducción……..24
      3.3.2 Construcción del sistema extendido……..26
      3.3.3 Controlabilidad y la matriz de control del sistema discreto…...30
      3.3.4 Cálculo de la matriz de transformación a forma canónica controlable……31
      3.3.5 Polinomio característico requerido del sistema extendido……40
      3.3.6 Diseño de las matrices $\tilde{\tilde{G}}_R$ y $\tilde{\tilde{H}}_R$ ……43
      3.3.7 Cálculo de Kc y Ki……45
      3.3.8 Implementación en el esquema y simulación……49
         3.3.8.1 Diseño del controlador discreto linealizado……..49
         3.3.8.2 Prueba del controlador discreto sobre el continuo linealizado……52
         3.3.8.3 Sistema no lineal discreto……55
         3.3.8.4 Diseño del observador discreto linealizado……...62
         3.3.8.5 Unión del observador y controlador……..65
4. Análisis de la observabilidad cuando $q_3 = 0$ y $q_3 = \frac{\pi}{4}$……..67
5. Conclusiones……...70
6. Trabajos futuros……..71
7. Bibliografía y referencias……...……72

AGRADECIMIENTOS

Quisiera empezar agradeciendo a mi tutor(Adrian Peidro) por su compromiso, guía y paciencia durante la realización de este estudio, además de agradecer al resto de los docentes por su dedicación y empeño a lo largo de toda la trayectoria universitaria.

Además, deseo manifestar mi más sincero agradecimiento a mi familia, especialmente a mis progenitores. A mi madre, por enseñarme desde el principio la importancia de la responsabilidad; sin su ejemplo y respaldo constante, estoy seguro de que no habría alcanzado el nivel en el que estoy hoy, y estoy consciente de que todavía hay un extenso camino por recorrer. A mi padre, por su inagotable paciencia y por estar siempre presente cuando más lo he requerido, particularmente en las situaciones más complicadas del grado. Su soporte ha resultado esencial.

Este logro no representa un término, sino el inicio de nuevos desafíos. Cada avance realizado ha sido una enseñanza, y cada dificultad vencida, una oportunidad para evolucionar. Con esperanza, dedicación y el respaldo de aquellos que me rodean, afronto con vigor todo lo que está por llegar.

1. INTRODUCCIÓN AL ESTUDIO

1.1 Antecedentes

Como inicio, se proporcionará un panorama general de los robots manipuladores, enfocándonos en los que muestran redundancia, como el sistema que se está investigando en esta memoria. En el ámbito de la robótica, la redundancia se refiere a la presencia de varios grados de libertad (GDL) que superan los requeridos para realizar una tarea concreta. Esta característica, aunque implica una mayor complejidad en el estudio y manejo del sistema, también brinda beneficios considerables en cuanto a flexibilidad, superación de barreras, incremento de la exactitud y optimización energética.

El robot que se examinará es un manipulador redundante planar con tres grados de libertad, formado por dos articulaciones prismáticas y una articulación rotacional. Como el que se muestra en la siguiente Ilustración 1.

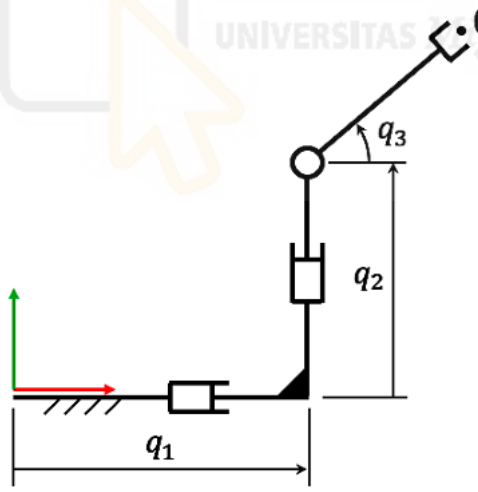


Ilustración 1

Esta arquitectura facilita el análisis de las especificaciones del control de sistemas mecánicos con acoplamientos dinámicos no triviales y con más entradas que salidas, lo que transforma el diseño del controlador en un reto técnico fascinante y educativo.

La teoría clásica de control está más pensada para sistemas con una única entrada y única salida. Para controlar sistemas que, como los robots manipuladores, poseen diversas entradas y salidas entre las que existe un elevado grado de acoplamiento dinámico, se vuelve más apta la teoría de control moderna o en el espacio de estados.

En el ámbito de control de robots, son numerosos los ejemplos de control en el espacio de estados, en donde al igual que este TFG se evidenciarán los resultados obtenidos con su utilización(Sempere Ruiz ,2022). En (Peidró et al., 2024) se presenta una revisión de otros tantos trabajos de aplicación de la teoría de control en el espacio de estados a robots manipuladores, especialmente de tipo paralelo, y se presenta un laboratorio remoto para realización de experimentos con robots paralelos reales a través de internet.

Es importante tener claro cuál es la necesidad principal de optar por este tipo de control. En primer lugar, el control en el espacio de estados desempeña un papel crucial en nuestro TFG, ya que, al presentar nuestro robot varias entradas y salidas, el control clásico no sería adecuado. Por ello, el control moderno se convierte en una herramienta fundamental.

1.2 Objetivos y contribución

El objetivo principal de este Trabajo es diseñar e implementar un sistema de control en el espacio de estados para un robot manipulador redundante con tres grados de libertad. Se pretende desarrollar un controlador que permita estabilizar el sistema, garantizar el seguimiento de referencias deseadas y cumplir con una serie de especificaciones, tanto en régimen transitorio como en régimen permanente.

Para alcanzar este objetivo, se abordan distintas técnicas de control dentro del marco del modelado en espacio de estados. En concreto, aplicarán las siguientes tecnicas:

- Control por asignación de polos: Se diseña un controlador mediante realimentación del estado que permite situar los polos del sistema en posiciones deseadas del plano complejo, ajustando así su comportamiento dinámico (tiempo de establecimiento, sobreoscilación, etc.).

- Control integral: Se amplía el sistema con integradores que permiten eliminar el error en régimen permanente en las variables de salida. Este control integral se incorpora al diseño del controlador mediante un sistema extendido.
- Control discreto: El modelo continuo del sistema se discretiza para poder implementar el controlador en un entorno digital. El diseño de las matrices del sistema discreto (G, H) se realiza a partir de las matrices continuas (A, B) mediante técnicas de discretización por muestreo.
- Diseño de observador: Dado que no siempre es posible medir directamente todos los estados del sistema, se diseña un observador que permite estimarlos a partir de las salidas disponibles. Se aplica también una transformación a forma canónica observable para facilitar el cálculo de la ganancia del observador.

Una de las principales diferencias y aportaciones de este trabajo respecto a estudios previos es la es la aplicación del control lineal en el espacio de estados a un robot redundante, frente a los robots no-redundantes abordados hasta ahora en los trabajos previos mencionados en la sección 1.1.Esto supone un reto adicional tanto a nivel de modelado como de diseño del regulador, debido a la complejidad inherente a los sistemas MIMO con acoplamientos dinámicos y grados de libertad excedentes.

Todo el desarrollo se valida mediante simulaciones en MATLAB y Simulink, tanto sobre modelos linealizados como sobre el modelo dinámico no lineal completo, con el fin de comprobar la eficacia del controlador en condiciones realistas.

1.3 Estructura de esta memoria

La presente memoria, seguirá los siguientes párrafos:

En primer lugar, se dedicará un capítulo al estudio de la dinámica no lineal del robot redundante que se analiza en este trabajo. En este apartado se parte del modelo matemático conocido que describe el comportamiento del sistema, con el objetivo de disponer de una base sólida para su posterior linealización y control.

A continuación, se procederá a linealizar el sistema dinámico no lineal en torno a un punto de equilibrio. Esto nos permitirá obtener un modelo en incrementos que

facilita enormemente el diseño del controlador. En este mismo apartado se comparan las respuestas del sistema linealizado y del no lineal ante una misma entrada, con el fin de comprobar si el modelo simplificado es suficientemente representativo en el entorno del punto de operación.

Tras esta parte, llega el núcleo del proyecto. Se diseñará un regulador por realimentación del estado, añadiendo integradores para garantizar que el sistema cumpla los requisitos tanto en régimen permanente como en transitorio.

En esta etapa se presta especial atención al diseño de la matriz $\tilde{\tilde{G}}_R$ que representa el sistema ampliado en su forma canónica controlable.

Una vez diseñado el controlador, se implementará en Simulink, aplicándolo primero sobre el sistema linealizado (tanto discreto como continuo) y, posteriormente, sobre el sistema no lineal original.

En un apartado adicional se desarrollará el diseño de un observador, en el caso en que no se pueda medir directamente el estado completo del sistema. Se trabajará sobre la forma canónica observable y se calculará la ganancia adecuada para estimar correctamente los estados a partir de las salidas.

Finalmente, se presentarán las conclusiones obtenidas a lo largo del estudio y se propondrán posibles líneas futuras de trabajo. Entre ellas, se destaca la implementación de un prototipo físico para validar el sistema en la práctica, el estudio de robots con más grados de libertad, o la linealización a lo largo de trayectorias completas para abordar sistemas variantes en el tiempo.

2 MODELADO DE LA CINEMÁTICA Y DINÁMICA DEL ROBOT

El primer objetivo para realizar será linealizar las ecuaciones no-lineales que describe la cinemática directa y la dinámica del robot. La cinemática directa, implica la relación cinemática entre las coordenadas articulares ‘‘q_i’’ y las salidas ‘‘y_i’’.

$$y_1 = q_1 + \cos q_3 \qquad \textit{Ecuación 1}$$

$$y_2 = q_2 + \sin q_3 \qquad \textit{Ecuación 2}$$

La dinámica del robot se describe mediante la siguiente ecuación matricial, que asume que existen masas puntuales ‘‘m_i’’ ubicadas en el extremo final de cada articulación:

$$\begin{bmatrix} m_1+m_2+m_3 & 0 & -m_3\sin q_3 \\ 0 & m_2+m_3 & m_3\cos q_3 \\ -m_3\sin q_3 & m_3\cos q_3 & m_3 \end{bmatrix}\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\\ \ddot{q}_3\end{bmatrix} = \begin{bmatrix} m_3\dot{q}_3^2\cos q_3 \\ m_3\dot{q}_3^2\sin q_3 - (m_2+m_3)g \\ -m_3\, g\cos q_3 \end{bmatrix} + \begin{bmatrix}u_1\\ u_2\\ u_3\end{bmatrix} \qquad \textit{Ecuación 3}$$

Para empezar, usaremos Derive para facilitar la linealización mediante el uso de la función JACOBIAN. Esta función, nos permite linealizar todas las ecuaciones en conjunto, ya que, como sabemos, la JACOBIANA es la derivada de muchas funciones con respecto a múltiples variables.

A continuación, se procederá a detallar todos los pasos realizados para la linealización.

Partimos de la Ecuación 3. Debido a que derive no entiende las segundas derivadas como doble punto, procederemos a cambiar la notación de la siguiente manera.

$$\begin{bmatrix} m_1+m_2+m_3 & 0 & -m_3\sin q_3 \\ 0 & m_2+m_3 & m_3\cos q_3 \\ -m_3\sin q_3 & m_3\cos q_3 & m_3 \end{bmatrix}\begin{bmatrix}ddq_1\\ ddq_2\\ ddq_3\end{bmatrix} = \begin{bmatrix} m_3 dq_3^2\cos q_3 \\ m_3 dq_3^2\sin q_3 - (m_2+m_3)g \\ -m_3\, g\cos q_3 \end{bmatrix} + \begin{bmatrix}u_1\\ u_2\\ u_3\end{bmatrix}$$

Pasaremos todo a un solo lado para facilitar las operaciones matemáticas.

$$\begin{bmatrix} m_1+m_2+m_3 & 0 & -m_3\sin q_3 \\ 0 & m_2+m_3 & m_3\cos q_3 \\ -m_3\sin q_3 & m_3\cos q_3 & m_3 \end{bmatrix}\begin{bmatrix} ddq_1 \\ ddq_2 \\ ddq_3 \end{bmatrix} - \begin{bmatrix} m_3 dq_3^2\cos q_3 \\ m_3 dq_3^2\sin q_3 - (m_2+m_3)g \\ -m_3\, g\cos q_3 \end{bmatrix} - \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Ecuación 4 Código DERIVE: Pasando todo a un solo miembro de la ecuación.

Luego se procederá a operarlo para dejarlo todo en función de una sola matriz.

$$\begin{bmatrix} -dq3^2.m3.COS(q3) - ddq3.m3.SIN(q3) + ddq1.(m1+m2+m3) - u1 \\ ddq3.m3.COS(q3) - dq3^2.m3.SIN(q3) + ddq2.(m2+m3) + g.(m2+m3) - u2 \\ (ddq2.m3 + g.m3).COS(q3) - ddq1.m3.SIN(q3) + ddq3.m3 - u3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Ecuación 5 Código DERIVE: Resultado de la operación.

Como observamos, nos damos cuenta de que las tres funciones a linealizar dependen de las siguientes variables.
[ddq1, ddq2, ddq3, dq3, u1, u2, u3, q3]

Este paso es importante ya que de esta forma podemos hacer la linealización de todas las funciones usando JACOBIAN.

$$JACOBIAN([-dq3^2 \cdot m3 \cdot COS(q3) - ddq3 \cdot m3 \cdot SIN(q3) + ddq1 \cdot (m1 + m2 + m3) - u1, ddq3 \cdot m3 \cdot COS(q3) - dq3^2 \cdot m3 \cdot SIN(q3) + ddq2 \cdot (m2 + m3) + g \cdot (m2 + m3) - u2, (ddq2 \cdot m3 + g \cdot m3) \cdot COS(q3) - ddq1 \cdot m3 \cdot SIN(q3) + ddq3 \cdot m3 - u3], [ddq1, ddq2, ddq3, dq3, u1, u2, u3, q3])$$

Ecuación 6 Código DERIVE: definición de la Jacobiana.

Ahora simplificamos los resultados.

$$\begin{bmatrix} m1+m2+3 & 0 & -m3.SIN(q3) & -2.dq3.m3.COS(q3) & -1 & 0 & 0 & dq3^2.m3.SIN(q3) - ddq3.m3.COS(q3) \\ 0 & m2+m3 & m3.COS(q3) & -2.dq3.m3.SIN(q3) & 0 & -1 & 0 & -dq3^2.m3.COS(q3) - ddq3.m3.SIN(q3) \\ -m3.SIN(q3) & m3.COS(q3) & m3 & 0 & 0 & 0 & -1 & -dq1.m3.COS(q3) - m3.(ddq2+g).SIN(q3) \end{bmatrix}$$

Ecuación 7 Código DERIVE: simplificación de resultados.

Por último, multiplicamos por las variables incrementales puestas en columna, ya que cuando linealizamos todas las variables pasan a ser incrementos.

$$\begin{bmatrix} m1+m2+3 & 0 & -m3.SIN(q3) & -2.dq3.m3.COS(q3) & -1 & 0 & 0 & dq3^2.m3.SIN(q3) - ddq3.m3.COS(q3) \\ 0 & m2+m3 & m3.COS(q3) & -2.dq3.m3.SIN(q3) & 0 & -1 & 0 & -dq3^2.m3.COS(q3) - ddq3.m3.SIN(q3) \\ -m3.SIN(q3) & m3.COS(q3) & m3 & 0 & 0 & 0 & -1 & -dq1.m3.COS(q3) - m3.(ddq2+g).SIN(q3) \end{bmatrix} \cdot \begin{bmatrix} \Delta ddq1 \\ \Delta ddq2 \\ \Delta ddq3 \\ \Delta dq3 \\ \Delta u1 \\ \Delta u2 \\ \Delta u3 \\ \Delta q3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Ecuación 8 Código DERIVE: Multiplicando por sus incrementales.

Procedemos a realizar el producto.

$$\begin{bmatrix} -(ddq3.m3.\Delta q3 + 2.dq3.m3.\Delta dq3).COS(q3) + (dq3^2.m3.\Delta q3 - m3.\Delta ddq3).SIN(q3) + m1.\Delta ddq1 + m2.\Delta ddq1 + m3.\Delta ddq1 - \Delta u1 \\ (m3.\Delta ddq3 - dq3^2.m3.\Delta q3).COS(q3) - (ddq3.m3.\Delta q3 + 2.dq3.m3.\Delta dq3).SIN(q3) + m2.\Delta ddq2 + m3.\Delta ddq2 - \Delta u2 \\ (m3.\Delta ddq2 - ddq1.m3.\Delta q3).COS(q3) - m3.(ddq2.\Delta q3 + g.\Delta q3 + \Delta ddq1).SIN(q3) + m3.\Delta ddq3 - \Delta u3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Ecuación 9 Código DERIVE: Resultado de la multiplicación

.

Expandimos lo de adentro del paréntesis, para poder identificar todos los términos. De este modo obtendremos el modelo linealizado.

$$\begin{bmatrix} -ddq3.m3.\Delta q3\,.COS(q3) - 2.dq3.m3.\Delta dq3.COS(q3) + dq3^2.m3.\Delta q3.SIN(q3) - m3.\Delta ddq3.SIN(q3) + m1.\Delta ddq1 + m2.\Delta ddq1 + m3.\Delta ddq1 - \Delta u1 \\ -dq3^2.m3.\Delta q3.COS(q3) + m3.\Delta ddq3.COS(q3) - ddq3.m3.\Delta q3.SIN(q3) - 2.dq3.m3.\Delta dq3.SIN(q3) + m2.\Delta ddq2 + m3.\Delta ddq2 - \Delta u2 \\ -ddq1.m3.\Delta q3.COS(q3) + m3.\Delta ddq2.COS(q3) - ddq2.m3.\Delta q3.SIN(q3) - g.m3.\Delta q3.SIN(q3) - m3.\Delta ddq1.SIN(q3) + m3.\Delta ddq3 - \Delta u3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Ecuación 10 Código DERIVE: Resultado de la expansión.

Cabe recordar que el vector de estados está compuesto por: [Δq1, Δq2, Δq3, Δdq1, Δdq2, Δdq3], correspondientes a las posiciones y velocidades.

Como siguiente paso, se utilizará la herramienta SOLVE de DERIVE para calcular el incremento de las aceleraciones ddq1, ddq2 y ddq3. Dado que se dispone de tres ecuaciones con tres incógnitas, el sistema es completamente determinado, por lo que la solución puede obtenerse sin mayores dificultades.

Este paso resulta fundamental para la obtención de las matrices, ya que nos permitirá calcular directamente las variables de estado derivadas. Como es bien sabido, la ecuación general del sistema en espacio de estados está dada por:

$$\Delta\dot{x} = \mathrm{A} * \Delta\mathrm{x} + \mathrm{B} * \Delta\mathrm{u}$$

Donde el miembro izquierdo de la igualdad corresponde a las derivadas de las variables de estado. Teniendo en cuenta que las variables de estado "x" y la entrada "u " son conocidas, es posible, a partir de estas expresiones, extraer las matrices A, B, C y D. A continuación, se procederá a mostrar los pasos detallados.

$$SOLVE\left(\begin{bmatrix} -ddq3.m3.\Delta q3.\mathrm{COS}(q3) - 2.\mathrm{dq3}.\mathrm{m3}.\Delta\mathrm{dq3}.\mathrm{COS}(q3) + dq3^2.\mathrm{m3}.\Delta q3.\mathrm{SIN}(q3) - \mathrm{m3}.\Delta\mathrm{ddq3}.\mathrm{SIN}(q3) + \mathrm{m1}.\Delta\mathrm{ddq1} + \mathrm{m2}.\Delta\mathrm{ddq1} + \mathrm{m3}.\Delta\mathrm{ddq1} - \Delta u1 \\ -dq3^2.\mathrm{m3}.\Delta q3.\mathrm{COS}(q3) + \mathrm{m3}.\Delta\mathrm{ddq3}.\mathrm{COS}(q3) - ddq3.m3.\Delta q3.\mathrm{SIN}(q3) - 2.\mathrm{dq3}.\mathrm{m3}.\Delta\mathrm{dq3}.\mathrm{SIN}(q3) + \mathrm{m2}.\Delta\mathrm{ddq2} + \mathrm{m3}.\Delta\mathrm{ddq2} - \Delta u2 \\ -\mathrm{ddq1}.\mathrm{m3}.\Delta q3.\mathrm{COS}(q3) + \mathrm{m3}.\Delta\mathrm{ddq2}.\mathrm{COS}(q3) - ddq2.m3.\Delta q3.\mathrm{SIN}(q3) - \mathrm{g}.\mathrm{m3}.\Delta q3.\mathrm{SIN}(q3) - \mathrm{m3}.\Delta\mathrm{ddq1}.\mathrm{SIN}(q3) + \mathrm{m3}.\Delta\mathrm{ddq3} - \Delta u3 \end{bmatrix}, [\Delta\mathrm{ddq1}, \Delta\mathrm{ddq2}, \Delta\mathrm{ddq3}]\right)$$

Ejecutando en DERIVE obtenemos lo siguiente:

Debido a que la solución de las aceleraciones linealizadas es muy grande, se mostrarán en la siguiente página.

$$\Delta ddq1 = (m1 \cdot m3\text{^}2 \cdot (ddq3 \cdot \Delta q3 + 2 \cdot dq3 \cdot \Delta dq3) \cdot COS(q3)\text{^}3 + m1 \cdot m3 \cdot \Delta u1 \cdot COS(q3)\text{^}2 + m3 \cdot COS(q3) \cdot (m1 \cdot m3 \cdot (ddq3 \cdot \Delta q3 + 2 \cdot dq3 \cdot \Delta dq3) \cdot SIN(q3)\text{^}2 + (m1 + m2 + m3) \cdot (\Delta u2 - ddq1 \cdot \Delta q3 \cdot (m2 + m3)) \cdot SIN(q3) - (m1 + m2) \cdot (m2 + m3) \cdot (ddq3 \cdot \Delta q3 + 2 \cdot dq3 \cdot \Delta dq3)) - (m2 + m3) \cdot (m3 \cdot (ddq2 \cdot \Delta q3 \cdot (m1 + m2 + m3) + g \cdot \Delta q3 \cdot (m1 + m2 + m3) + \Delta u1) \cdot SIN(q3)\text{^}2 + (m1 + m2 + m3) \cdot (\Delta u3 - dq3\text{^}2 \cdot m3 \cdot \Delta q3) \cdot SIN(q3) + \Delta u1 \cdot (m1 + m2))))/((m1 + m2 + m3) \cdot (m1 \cdot m3 \cdot COS(q3)\text{^}2 - (m1 + m2) \cdot (m2 + m3)))$$

Ecuación 11 Código DERIVE: Resultado de $\Delta ddq1$

$$\Delta ddq2 = (m1 \cdot m3 \cdot COS(q3)\text{^}2 \cdot (m3 \cdot (ddq3 \cdot \Delta q3 + 2 \cdot dq3 \cdot \Delta dq3) \cdot SIN(q3) + ddq1 \cdot \Delta q3 \cdot (m1 + m2 + m3)) + (m1 + m2 + m3) \cdot COS(q3) \cdot (m3 \cdot (ddq2 \cdot \Delta q3 \cdot (m1 + m2 + m3) + g \cdot \Delta q3 \cdot (m1 + m2 + m3) + \Delta u1) \cdot SIN(q3) - (m1 + m2 + m3) \cdot (dq3\text{^}2 \cdot m3 \cdot \Delta q3 - \Delta u3)) + m1 \cdot m3\text{^}2 \cdot (ddq3 \cdot \Delta q3 + 2 \cdot dq3 \cdot \Delta dq3) \cdot SIN(q3)\text{^}3 + m3 \cdot (m1 + m2 + m3) \cdot (\Delta u2 - ddq1 \cdot \Delta q3 \cdot (m2 + m3)) \cdot SIN(q3)\text{^}2 - m3 \cdot (m1\text{^}2 + 2 \cdot m1 \cdot (m2 + m3) + m2 \cdot (m2 + m3)) \cdot (ddq3 \cdot \Delta q3 + 2 \cdot dq3 \cdot \Delta dq3) \cdot SIN(q3) + (m1 + m2 + m3) \cdot (ddq1 \cdot m3 \cdot \Delta q3 \cdot (m2 + m3) - \Delta u2 \cdot (m1 + m2 + m3)))/((m1 + m2 + m3) \cdot (m1 \cdot m3 \cdot COS(q3)\text{^}2 - (m1 + m2) \cdot (m2 + m3)))$$

Ecuación 12 Código DERIVE: Resultado de $\Delta ddq2$

$$\Delta ddq3 = (dq3\text{^}2 \cdot m1 \cdot m3\text{^}2 \cdot \Delta q3 \cdot COS(q3)\text{^}2 + m3 \cdot COS(q3) \cdot (m1 \cdot m3 \cdot (ddq3 \cdot \Delta q3 + 2 \cdot dq3 \cdot \Delta dq3) \cdot SIN(q3) - (m1 + m2 + m3) \cdot (ddq1 \cdot \Delta q3 \cdot (m2 + m3) - \Delta u2)) - (m2 + m3) \cdot (m3 \cdot (ddq2 \cdot \Delta q3 \cdot (m1 + m2 + m3) + g \cdot \Delta q3 \cdot (m1 + m2 + m3) + \Delta u1) \cdot SIN(q3) - dq3\text{^}2 \cdot m3\text{^}2 \cdot \Delta q3 + \Delta u3 \cdot (m1 + m2 + m3)))/(m3 \cdot (m1 \cdot m3 \cdot COS(q3)\text{^}2 - (m1 + m2) \cdot (m2 + m3)))]$$

Ecuación 13 Código DERIVE: Resultado de $\Delta ddq3$

Como siguiente paso, se procederá a la obtención de las variables de estado derivadas($\dot{x}$). Para ello, es fundamental disponer de las expresiones correspondientes a la derivada de la posición y de la velocidad. Dado que las variables de estado consideradas en este sistema son la posición y velocidad, la obtención de su derivada implica de forma natural, el conocimiento de las aceleraciones linealizadas. Por ello, en el paso previo se ha llevado a cabo el cálculo de las derivadas de las velocidades(aceleraciones).

Como hemos dicho en el párrafo anterior, debemos obtener las variables de estado derivadas para poder obtener las matrices con la ayuda de DERIVE. A continuación, se mostrará las variables de estado derivadas.

$$\Delta\dot{x} = \begin{bmatrix} \Delta dq1 \\ \Delta dq2 \\ \Delta dq3 \\ \Delta ddq1 \\ \Delta ddq2 \\ \Delta ddq3 \end{bmatrix}$$

Ecuación 14 Variables de estado derivadas.

Finalmente, para la obtención de las matrices del sistema, se procederá a factorizar todos los términos que contengan las variables de estado no derivadas. De este modo, se logrará identificar la matriz A. Asimismo, seguiremos un proceso análogo para la obtención de las matrices B,C,D.

Lo descrito en el párrafo anterior se llevará a cabo mediante el uso del comando JACOBIAN, al cual se le proporciona como primer argumento el vector de estado derivado en formato columna y, como segundo argumento, el vector de estado (sin derivar). De este modo, al ejecutar el comando, se obtiene la matriz A.

$$A = JACOBIAN(\Delta\dot{x}\ COL\ 1, [\Delta q1, \Delta q2, \Delta q3, \Delta dq1, \Delta dq2, \Delta dq3])$$

Ecuación 15 Código DERIVE: obtención de la matriz A.

Siguiendo el mismo proceso, en lugar de poner el vector de estado, si queremos obtener la matriz B debemos poner el vector de las entradas de nuestro sistema.

$$B = JACOBIAN(\Delta\dot{x}\ COL\ 1, [\Delta u1, \Delta u2, \Delta u3])$$

Ecuación 16 Código DERIVE: obtención de la matriz B.

Dado que, para la obtención de las matrices C y D, es necesario disponer del vector de salida linealizado. Tal como indica la ecuación correspondiente:

$$\Delta y = C * \Delta x + D * \Delta u$$

A partir de esta expresión, se puede obtener la matriz **C** como la derivada parcial del vector de salida respecto al vector de estado, y la matriz **D** como la derivada parcial del vector de salida respecto al vector de entrada.

Primero debemos obtener la matriz de salida linealizada, y esto lo haremos con los siguientes comandos en DERIVE.

$$\begin{bmatrix} y1 \\ y2 \end{bmatrix} = \begin{bmatrix} q1 + COS(q3) \\ q2 + SIN(q3) \end{bmatrix}$$

Ecuación 17

A continuación, se pasará todo hacia el lado izquierdo de la igualdad para poder facilitar los cálculos.

$$\begin{bmatrix} y1 \\ y2 \end{bmatrix} - \begin{bmatrix} q1 \ + \ COS(q3) \\ q2 \ + \ SIN(q3) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Ecuación 18

Desarrollando la ecuación, obtendremos lo siguiente:

$$\begin{bmatrix} -COS(q3) - q1 + y1 \\ -SIN(q3) - q2 + y2 \end{bmatrix}$$

Ecuación 19

Nos damos cuenta de que nuestra matriz, depende de las siguientes variables: [q1, q2, q3, y1, y2].
Haremos uso de la JACOBIANA tal cual hicimos en la Ecuación 6.

$$JACOBIAN([-COS(q3) \ - \ q1 \ + \ y1, -SIN(q3) \ - \ q2 \ + \ y2], [q1, q2, q3, y1, y2])$$

Una vez ejecutado el comando anterior, tendremos como resultado lo siguiente:

$$\begin{bmatrix} -1 & 0 & SIN(q3) & 1 & 0 \\ 0 & -1 & COS(q3) & 0 & 1 \end{bmatrix}$$

Ecuación 20 Código DERIVE: Resultado de la JACOBIANA

Como paso siguiente, procederemos a multiplicar por las variables incrementales ya que de este modo nuestra linealización estará completa.

$$\begin{bmatrix} -1 & 0 & SIN(q3) & 1 & 0 \\ 0 & -1 & COS(q3) & 0 & 1 \end{bmatrix} . [\Delta q1, \Delta q2, \Delta q3, \Delta y1, \Delta y2]$$

Ecuación 21 Código DERIVE: Paso previo para la obtención de la matriz de salida linealizada.

Ejecutando y organizando los términos, obtendremos lo siguiente:

$$\begin{bmatrix} \Delta y1 \\ \Delta y2 \end{bmatrix} = \begin{bmatrix} -\Delta q3 . SIN(q3) + \Delta q1 \\ \Delta q3 . COS(q3) + \Delta q2 \end{bmatrix}$$

Ecuación 22 Código DERIVE: Matriz de salida linealizada.

Llegados hasta este punto, debemos recordar que nuestro objetivo principal es la obtención de las matrices C y D. Es por ello, que como siguiente paso lo que debemos hacer es seguir los pasos realizados en las Ecuación 15 y Ecuación 16.

$$C = JACOBIAN([\Delta y1, \Delta y2]\ COL\ 1, [\Delta q1, \Delta q2, \Delta q3, \Delta dq1, \Delta dq2, \Delta dq3])$$

Ecuación 23 Código DERIVE: obtención de la matriz C.

$$D = JACOBIAN([\Delta y1, \Delta y2]\ COL\ 1, [\Delta u1, \Delta u2, \Delta u3])$$

Ecuación 24 Código DERIVE: obtención de la matriz D.

Los resultados de las obtenciones de las matrices A,B,C,D se muestran en la siguiente tabla.

```
A = [0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0, 0, 0, 0, 1; 0, 0,
m3*(ddq3*m1*m3*cos(q3)^3 + cos(q3)*(ddq3*m1*m3*sin(q3)^2 - ddq1*(m1 +
m2 + m3)*(m2 + m3)*sin(q3) - ddq3*(m1 + m2)*(m2 + m3)) - (m1 + m2 +
m3)*(m2 + m3)*sin(q3)*((ddq2 + g)*sin(q3) - dq3^2))/((m1 + m2 +
m3)*(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3))), 0, 0,
2*dq3*m2*m3*cos(q3)/((m1 + m2)*(m2 + m3) - m1*m3*cos(q3)^2); 0, 0,
m3*(cos(q3)^2*(ddq3*m1*m3*sin(q3) + ddq1*(m1 + m2 + m3)^2) + (m1 + m2
+ m3)^2*cos(q3)*((ddq2 + g)*sin(q3) - dq3^2) +
ddq3*sin(q3)*(m1*m3*sin(q3)^2 - m1^2 - 2*m1*(m2 + m3) - m2*(m2 +
m3)))/((m1 + m2 + m3)*(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3))), 0, 0,
2*dq3*m3*(m1 + m2)*sin(q3)/((m1 + m2)*(m2 + m3) - m1*m3*cos(q3)^2); 0,
0, (dq3^2*m1*m3*cos(q3)^2 + cos(q3)*(ddq3*m1*m3*sin(q3) - ddq1*(m1 +
m2 + m3)*(m2 + m3)) - (m2 + m3)*((ddq2 + g)*(m1 + m2 + m3)*sin(q3) -
dq3^2*m3))/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)), 0, 0,
2*dq3*m1*m3*sin(q3)*cos(q3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3))];
B = [0, 0, 0; 0, 0, 0; 0, 0, 0; (m1*m3*cos(q3)^2 - (m2 +
m3)*(m3*sin(q3)^2 + m1 + m2))/((m1 + m2 + m3)*(m1*m3*cos(q3)^2 - (m1
+ m2)*(m2 + m3))), m3*sin(q3)*cos(q3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2
+ m3)), (m2 + m3)*sin(q3)/((m1 + m2)*(m2 + m3) - m1*m3*cos(q3)^2);
m3*sin(q3)*cos(q3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)),
(m3*sin(q3)^2 - m1 - m2 - m3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)),
(m1 + m2 + m3)*cos(q3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)); (m2 +
m3)*sin(q3)/((m1 + m2)*(m2 + m3) - m1*m3*cos(q3)^2), (m1 + m2 +
m3)*cos(q3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)), (m1 + m2 + m3)*(m2
+ m3)/(m3*((m1 + m2)*(m2 + m3) - m1*m3*cos(q3)^2))];
C = [ 1 0 -sin(q3) 0 0 0;0 1 cos(q3) 0 0 0];
D = [0 0 0;0 0 0];
```

Tabla 1 Resultado de las matrices A,B,C,D.

2.1 Obtención de la ecuación de estados

Para la obtención de las matrices A, B, C y D se ha creado una función llamada calculaABCD. Los parámetros de los que depende la función son los siguientes: (Q, dQ, ddQ). Estos son los valores en el punto de funcionamiento que nos será útil para poder obtener la linealización. Estas variables, en realidad, contienen cada una de las articulaciones; en este caso, son tres. Es decir, Q (posición) contiene q1, q2 y q3; de igual forma ocurre con dQ (velocidades): dq1, dq2 y dq3, y con ddQ (aceleraciones): ddq1, ddq2 y ddq3.

Los cálculos se han realizado con los siguientes datos:

```
%% Datos de partida del robot:

q1=1;

q2=2;

q3=0;

m1=1;

m2=1;

m3=1;

g=10;
```

Tabla 2 Código MATLAB: Datos de partida.

```
function [A,B,C,D] = calculaABCD(Q,dQ,ddQ)
q1=Q(1);
q2=Q(2);
q3=Q(3);
dq1=dQ(1);
dq2=dQ(2);
dq3=dQ(3);
ddq1=ddQ(1);
ddq2=ddQ(2);
ddq3=ddQ(3);
m1=1;
m2=1;
m3=1;
g=10;
A = [0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0, 0, 0, 0, 1; 0, 0,
m3*(ddq3*m1*m3*cos(q3)^3 + cos(q3)*(ddq3*m1*m3*sin(q3)^2 - ddq1*(m1 + m2 +
m3)*(m2 + m3)*sin(q3) - ddq3*(m1 + m2)*(m2 + m3)) - (m1 + m2 + m3)*(m2 +
m3)*sin(q3)*((ddq2 + g)*sin(q3) - dq3^2))/((m1 + m2 + m3)*(m1*m3*cos(q3)^2
- (m1 + m2)*(m2 + m3))), 0, 0, 2*dq3*m2*m3*cos(q3)/((m1 + m2)*(m2 + m3) -
m1*m3*cos(q3)^2); 0, 0, m3*(cos(q3)^2*(ddq3*m1*m3*sin(q3) + ddq1*(m1 + m2 +
m3)^2) + (m1 + m2 + m3)^2*cos(q3)*((ddq2 + g)*sin(q3) - dq3^2) +
ddq3*sin(q3)*(m1*m3*sin(q3)^2 - m1^2 - 2*m1*(m2 + m3) - m2*(m2 + m3)))/((m1
+ m2 + m3)*(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3))), 0, 0, 2*dq3*m3*(m1 +
m2)*sin(q3)/((m1 + m2)*(m2 + m3) - m1*m3*cos(q3)^2); 0, 0,
(dq3^2*m1*m3*cos(q3)^2 + cos(q3)*(ddq3*m1*m3*sin(q3) - ddq1*(m1 + m2 +
m3)*(m2 + m3)) - (m2 + m3)*((ddq2 + g)*(m1 + m2 + m3)*sin(q3) -
dq3^2*m3))/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)), 0, 0,
2*dq3*m1*m3*sin(q3)*cos(q3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3))];
B = [0, 0, 0; 0, 0, 0; 0, 0, 0; (m1*m3*cos(q3)^2 - (m2 + m3)*(m3*sin(q3)^2
+ m1 + m2))/((m1 + m2 + m3)*(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3))),
m3*sin(q3)*cos(q3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)), (m2 +
m3)*sin(q3)/((m1 + m2)*(m2 + m3) - m1*m3*cos(q3)^2);
m3*sin(q3)*cos(q3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)), (m3*sin(q3)^2 -
m1 - m2 - m3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)), (m1 + m2 +
m3)*cos(q3)/(m1*m3*cos(q3)^2 - (m1 + m2)*(m2 + m3)); (m2 + m3)*sin(q3)/((m1
+ m2)*(m2 + m3) - m1*m3*cos(q3)^2), (m1 + m2 + m3)*cos(q3)/(m1*m3*cos(q3)^2
- (m1 + m2)*(m2 + m3)), (m1 + m2 + m3)*(m2 + m3)/(m3*((m1 + m2)*(m2 + m3) -
m1*m3*cos(q3)^2))];
C = [ 1 0 -sin(q3) 0 0 0;0 1 cos(q3) 0 0 0];
D = [0 0 0;0 0 0];
```

Tabla 3 Código MATLAB: Definición de las matrices ABCD

3 Diseño de un controlador MIMO(Multiple Input Multiple Output)

En este capítulo se tiene por finalidad estabilizar el sistema mediante la implementación de un controlador por realimentación del estado, con el objetivo de lograr su estabilización en lazo cerrado. Asimismo, se busca que las variables de salida del sistema alcancen las posiciones deseadas con error nulo en régimen permanente, gracias a la acción integral (integradores). Además, se procura cumplir con determinadas especificaciones dinámicas, como el tiempo de establecimiento y la sobreoscilación.

Además, se asumirá que el control se realiza por computador, por lo que será necesario discretizar el sistema continuo.

3.1 Sistema discreto equivalente

Como sabemos, las únicas matrices que varían de un sistema continuo a discreto son A y B que pasarán a ser G y H. Estos últimos lo procederemos a obtener mediante un comando en MATLAB(c2d) indicándole como argumentos las matrices que deseamos transformar y el periodo de muestreo.

Este procedimiento sencillo nos permite obtener las matrices G y H, con las que estudiaremos la controlabilidad del sistema. Asimismo, las matrices G y C serán utilizadas para analizar su observabilidad.

```
%Discretizamos el sistema linealizado para un periodo T
%segundos
[G,H]=c2d(A,B,Ts);
```

Tabla 4 Código MATLAB: Discretización.

El resultado que se mostrará a continuación ha sido obtenido considerando que el robot se encuentra en reposo, lo cual, en nuestro caso, corresponde a las condiciones en las que las articulaciones toman los valores $q_1 = 1, q_2 = 2, q_3 = 0$ y tanto las velocidades como las aceleraciones correspondientes son nulas.

Ejecutando en Matlab el código de arriba, obtenemos las siguientes matrices:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 25

$$H = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.033 & 0 & 0 \\ 0 & 0.01 & -0.1 \\ 0 & -0.01 & 0.02 \end{bmatrix}$$

Ecuación 26

Las matrices obtenidas tendrán el mismo orden que las anteriores. Es decir, como se puede observar, la matriz G tiene la misma dimensión que A(6x6).

Para saber si es controlable utilizaremos las matrices G, H y para la observabilidad las matrices G, C.

3.2 Controlabilidad y observabilidad del sistema discreto equivalente

Antes de profundizar en este apartado, es útil entender el propósito del estudio de la controlabilidad y la observabilidad. La controlabilidad nos permite determinar si, mediante ciertas variaciones en las entradas, es posible alcanzar cualquier punto dentro del espacio de estados del sistema. Por su parte, la observabilidad analiza en qué medida los estados del sistema afectan o se reflejan en sus salidas. Con ello, el siguiente paso sería determinar si el sistema es controlable(Q) y observable(P), ya que, en caso contrario, lo que se debería hacer es dividir el sistema en su parte controlable y observable, ya que solo podemos diseñar un controlador para la parte controlable.

A continuación, se mostrará las formas de calcularlas tanto con fórmula genérica como usando los comandos de Matlab.

Fórmula genérica:

$$Q = [H \quad GH \quad G^2H \quad \dots \quad G^{n-1}H]$$

$$P = \begin{bmatrix} C \\ CG \\ CG^2 \\ \vdots \\ CG^{n-1} \end{bmatrix}$$

Siendo:

n: El orden del sistema

Particularizando las ecuaciones anteriormente mostradas. Quedarían de la siguiente manera.

$$Q = [H \quad GH \quad G^2H \; G^3H \quad G^4H \quad G^5H] \qquad \textit{Ecuación 27}$$

$$P = \begin{bmatrix} C \\ CG \\ CG^2 \\ CG^3 \\ CG^4 \\ CG^5 \end{bmatrix} \qquad \textit{Ecuación 28}$$

Usando los comandos de Matlab se vería de la siguiente manera:

```
% Estudiaremos su controlabilidad
Q = ctrb(A,B);
rQ = rank(Q);
% Estudiaremos su observabilidad
P = obsv(A,C);
rP = rank(P);
```

Tabla 5 Código MATLAB: Estudio de la controlabilidad y observabilidad.

Como hemos mencionado anteriormente el robot, tiene 3 articulaciones y posee tres grados de libertad, que son seis estados(posición y velocidad), ya que se trata de un sistema mecánico. En este caso, estudiaremos la controlabilidad y la observabilidad de forma más genérica. Es decir, implementamos un bucle en el que, dependiendo de la articulación q_3, que es rotativa, se generan las matrices correspondientes.

Tomamos esta decisión al darnos cuenta de que, en condiciones estáticas (velocidades y aceleraciones nulas), dichas matrices solo varían cuando se modifica precisamente esa articulación; de lo contrario, permanecen constantes. Además de todo ello, se calculó la condición con el comando "cond ()" y hemos ido apilando cada valor para posteriormente obtener las gráficas.

El análisis de la condición de las matrices de controlabilidad y observabilidad permite estimar la energía mínima necesaria para controlar el sistema, así como la sensibilidad de la salida ante los distintos estados, lo cual es clave para el diseño de controladores y observadores robustos.

A continuación, se procederá a explicar de forma un poco más detalla este concepto debido a que es fundamental entenderlo.

Condición de la matriz controlable (Q)	Condición de la matriz observable (P)
Una mayor condición de la matriz de controlabilidad indica que el sistema, aunque sea teóricamente controlable, no lo es de manera eficiente en la práctica, ya que requiere mayor energía de entrada para alcanzar ciertos estados.	Una alta condición de la matriz de observabilidad implica que, aunque el sistema sea observable en teoría, algunos estados son difíciles de detectar porque producen muy poca energía en la salida, lo que los hace sensibles al ruido y difíciles de estimar con precisión

En las siguientes líneas, se mostrará el código.

```
n=6; % N será el tamaño de la matriz A.
contador_no_observable=0;
contador_no_controlable=0;
condsQ = [];
condsP= [];
valores_q3 = [0 : 0.005 : pi , pi , pi : 0.005 : 2*pi , 2*pi ];
% Esta línea nos está indicando el paso de 0 a pi, así como también de pi a
% 2pi.
for q3_value = valores_q3
    fprintf ('El valor de q3 que se está usando es: %d\n',q3_value);
    [A,B,C,D]=calculaABCD([0 0 q3_value], [0 0 0], [0 0 0]);
    fprintf ('Analizaremos su controlabilidad:\n');
    Q=ctrb(A,B);
    condsQ= [condsQ; cond(Q)]; % Apilo valores
    jd=rank(Q);
    fprintf ('El valor de rank de la matriz Q es: %d\n', jd);
    if jd == n
       fprintf ('La matriz es controlable\n');
    else
       fprintf ('La matriz no es controlable\n');
       contador_no_controlable=contador_no_controlable+1;
    end
    fprintf ('Analizaremos su observabilidad:\n');
    P=obsv(A,C);
    condsP = [ condsP; cond(P)];
    dn=rank(P);
    fprintf ('El valor de rank de la matriz P es: %d\n',dn);
    if dn == n
       fprintf ('La matriz es observable\n');
    else
       fprintf ('La matriz no es observable\n');
       contador_no_observable=contador_no_observable+1;
    end
end
plot (valores_q3, condsQ) % Gráfica para ver cada valor de q3 su variación.
figure
plot (valores_q3, condsP)
ylim ([0,1000]) % Representamos el eje vertical de 0 a 1000
```

Tabla 6 Código MATLAB: Cálculo de la observabilidad y controlabilidad

Los resultados nos muestran que nuestro sistema es tanto controlable como observable, lo cual facilita mucho los cálculos al no tener que separarlo.
Cabe indicar que, al contar con una articulación que puede girar libremente(q_3) el sistema puede presentar diferentes matrices A, B, C, D. Por ello, las condiciones de controlabilidad y observabilidad pueden cambiar.

Es por ello que, a continuación, mostraremos las gráficas obtenidas con el código adjunto, en donde se observarán con mayor detalle todos los valores que puede ir tomando q_3, así como también las gráficas de controlabilidad y observabilidad, con el objetivo de analizar su comportamiento.

- Condición de la controlabilidad

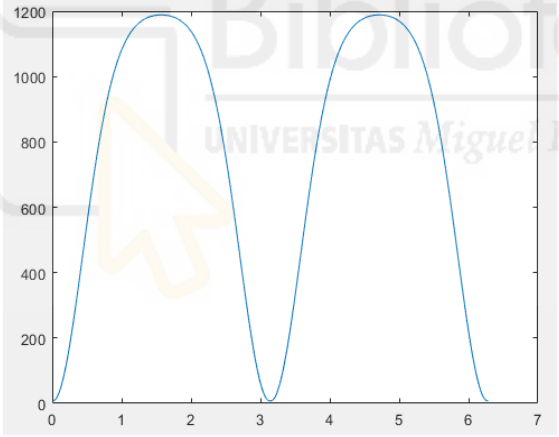


Ilustración 2 Controlabilidad.

Como se observa, el sistema presenta una buena condición de controlabilidad para q_3 cuando vale π y 2π, donde el control puede ejercerse de forma eficiente. Sin embargo, a medida que q_3 se aproxima a $\pi/2$, la condición de controlabilidad empeora, indicando que se requiere un mayor aporte de energía para lograr el control deseado, lo cual puede comprometer la eficiencia del sistema.

- Condición de la observabilidad

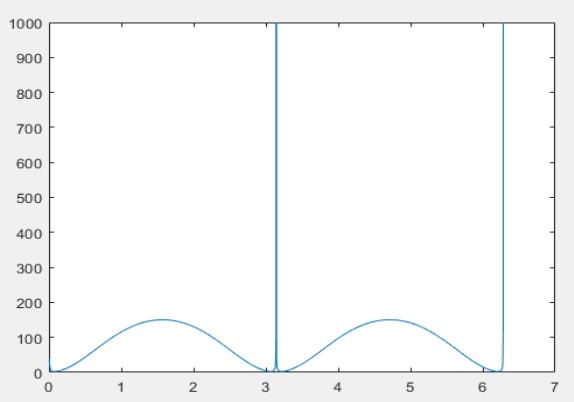


Ilustración 3 Observabilidad.

En esta gráfica ocurre algo similar, pero con un matiz un poco más profundo. Es decir, como se puede ver, cuando q_3 vale aproximadamente π o 2π, nuestro sistema presenta un pico prominente, lo que se traduce en que justo en esa combinación nuestro sistema no es observable.
Esto lo veremos más adelante con mayor detalle.

Ahora , procederemos a dar un ejemplo numérico considerando los siguientes valores para las articulaciones: q_1=1, q_2=2, q_3=0.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0.33 & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & 0 & 0 & 0 & -1 & 2 & 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0.33 & 0 & 0 & 0 & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & -1 & 2 & 0 & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix}$$

Ecuación 29

$$cond\_Q = 7.8541$$

Este valor nos indica que en esa configuración el robot fácilmente podemos alcanzar cualquier valor de los estados con poca energía a las entradas.

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0.01 & 0.01 \\ 1 & 0 & 0 & 0.02 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0.02 & 0.02 \\ 1 & 0 & 0 & 0.03 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0.03 & 0.03 \\ 1 & 0 & 0 & 0.04 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0.04 & 0.04 \\ 1 & 0 & 0 & 0.05 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0.05 & 0.05 \end{bmatrix}$$

Ecuación 30

$$cond\_P = \ 6.2475e + 17.$$

Esto significa que, en nuestro caso, el sistema no es observable para esa configuración, debido a que su número de condición es muy grande. Es por ello que, primero trabajaremos con el diseño del controlador.

A continuación, procederemos ya al diseño del controlador de nuestro sistema discreto.

3.3 Diseño del controlador por retroalimentación e integral

3.3.1 Introducción

En este inciso queremos reflejar el interés por diseñar el controlador. Este Trabajo de Fin de Grado se desarrollará más ampliamente en este apartado, ya que es aquí donde explicaremos detalladamente la conveniencia de posicionar los polos en el plano complejo de acuerdo con nuestras preferencias, con el fin de alcanzar el objetivo principal: que la dinámica del sistema se ajuste a unas especificaciones previamente establecidas.

Una vez finalizado este apartado, habremos sido capaz de diseñar la ganancia del regulador integral (K_i) y la matriz que retroalimentara (K_c). Antes de empezar ya con el cálculo de estas, es necesario entender por qué son necesarias. Son importantes ya que estas harán que la salida de nuestro sistema pueda cumplir con lo que uno requiera.

A continuación, se explicará de forma más clara la función de K_i y K_c:

- K_c: Ajusta el comportamiento dinámico del sistema para que su transitorio cumpla con determinada suavidad (tiempo de establecimiento, sobreoscilación).

- K_i: Elimina el error en régimen permanente.

Normalmente, además de diseñar el controlador, también se suele diseñar un observador (Ko). Esto se hace cuando los estados del sistema no son directamente medible**s**. Dado que esta es una situación habitual en aplicaciones prácticas, se recurre a estimadores para poder estimarlos a partir de las salidas medidas.

No obstante, como se ha observado previamente, cuando el valor de q3 es igual 0, el sistema resulta no observable, lo que imposibilita la estimación de los estados mediante un observador.
Por esta razón, se procederá al diseño del observador utilizando un ángulo de giro distinto de 0° para q_3, considerando también casos afines como 180° y 360°.

No obstante, es fundamental asegurarse de que los valores asignados a q_3 no comprometan la controlabilidad del sistema. Para ello, se debe analizar cuidadosamente la Ilustración 2 y la Ilustración 3, verificando que las configuraciones seleccionadas garanticen tanto la controlabilidad como la observabilidad del sistema.

En la imagen que se adjunta a continuación, se pretende dar una idea de la forma que tendrá nuestro esquema. Ahora debemos tener en cuenta qué implicación tendrá o qué cambio se hará en el esquema de Simulink.

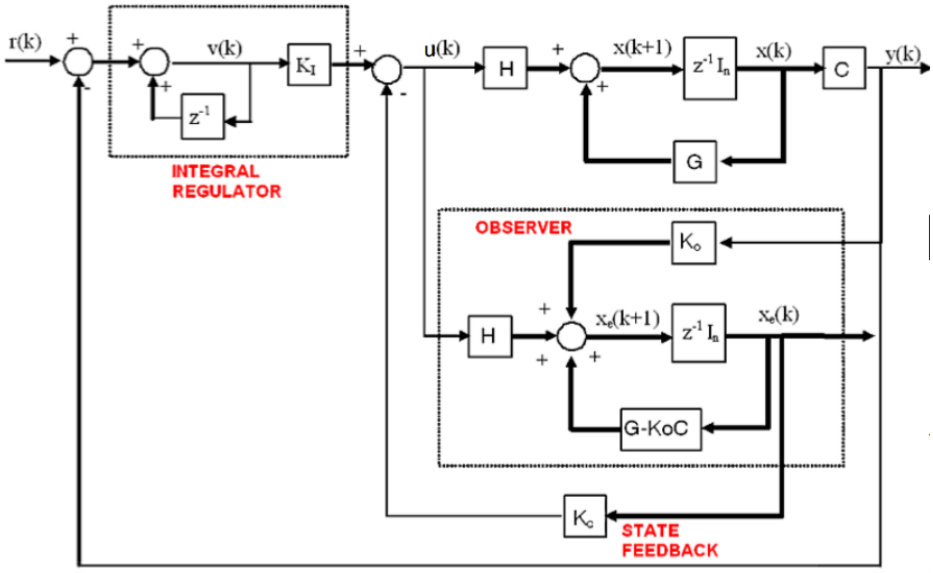


Ilustración 4 Esquema completo: Controlador más observador.

3.3.2 Construcción del sistema extendido

Antes de abordar en profundidad este apartado, es importante comprender la relevancia de este. Nuestro objetivo principal es eliminar el error en régimen permanente y ajustar el comportamiento transitorio. Para lograrlo, se plantea la construcción de un sistema extendido que permita incorporar integradores adicionales o realimentaciones adecuadas que mejoren el desempeño del sistema frente a perturbaciones y cambios en la referencia.

A continuación, se explicará de forma algebraica como se llega a la conclusión de que se tiene que aumentar las matrices para controlar el error.

Control integral:

$$v_k = v_{k-1} + e_k \qquad \textit{Ecuación 31}$$

- Donde, sabemos que por definición el error es:

$$e_k = r_k - y_k \qquad \textit{Ecuación 32}$$

- En la mayoría de los casos y en este no será la excepción la salida se define como:

$$y_k = C * x_k \qquad \textit{Ecuación 33}$$

ya que, como hemos visto, nuestra matriz D está vacía.
Luego, podemos reemplazar tanto la Ecuación 32 y Ecuación 33 en la Ecuación 31 y quedaría de la siguiente manera.

$$v_k = v_{k-1} + r_k - C * x_k \qquad \textit{Ecuación 34}$$

Podemos observar claramente la presencia de una nueva variable de estado, representada por v_k .Esta variable actúa como un acumulador del error entre la referencia r_k y la salida del sistema y_k, es decir, realiza la función de un integrador discreto.

Esto significa que, si inicialmente nuestro sistema tiene n variables de estado, al incorporar control integral, el número total de estados se incrementará en función de la cantidad de salidas en las que se desee eliminar el error en régimen permanente. Específicamente, el sistema pasará a tener:

$$n_{aumentado} = n + p \qquad \textit{Ecuación 35}$$

Donde:

- n: es el número original de estados del sistema,
- p: es el número de salidas que se desean integrar (es decir, aquellas en las que queremos eliminar el error en régimen permanente).

Una vez habiendo entendido todo ello, pasamos a mostrar la forma de la nueva ecuación de estados.

$$\begin{bmatrix} \dot{x}_{k+1} \\ \dot{v}_k \end{bmatrix} = \begin{bmatrix} G & 0_{n*p} \\ -C & I_{p*p} \end{bmatrix} * \begin{bmatrix} x_k \\ v_{k-1} \end{bmatrix} + \begin{bmatrix} H \\ 0_{p*m} \end{bmatrix} * [u_k] + \begin{bmatrix} 0 \\ I \end{bmatrix} r_k \qquad \textit{Ecuación 36}$$

Donde:

- p =Numero de salidas del sistema.
- m = Número de entradas.

Para entrar más en detalle, debemos introducir los nuevos términos que se utilizan en estos tipos de estudio.
Para ello debemos tener en cuenta lo siguiente:

$$\hat{G} = \begin{bmatrix} G & 0_{n*p} \\ -C & I_{p*p} \end{bmatrix} \; y \; \hat{H} = \begin{bmatrix} H \\ 0_{p*m} \end{bmatrix} \qquad \textit{Ecuación 37}$$

Donde:

- I : Matriz identidad.
- 0 : Matriz de ceros.

Con todo ello, la ecuación de estados queda, de la siguiente manera

$$\begin{bmatrix} \dot{x}_{k+1} \\ \dot{v}_k \end{bmatrix} = \hat{G} * \begin{bmatrix} x_k \\ v_{k-1} \end{bmatrix} + \hat{H} * [u_k] + \begin{bmatrix} 0 \\ I \end{bmatrix} r_k \qquad \textit{Ecuación 38}$$

Donde:

- $\hat{G}$: Matriz G aumentada después de añadir los integradores.
- $\hat{H}$: Matriz H aumentada después de añadir los integradores.

Las nuevas matrices aumentadas en nuestro caso de estudio tendrán los siguientes valores para nuestro caso:

$$\hat{G} = \begin{bmatrix} 1 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 39

$$\hat{H} = \begin{bmatrix} 0.0017 & 0 & 0 \\ 0 & 0.005 & -0.005 \\ 0 & -0.005 & 0.01 \\ 0.033 & 0 & 0 \\ 0 & 0.1 & -0.1 \\ 0 & -0.1 & 0.2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Ecuación 40

Las matrices presentadas anteriormente se han obtenido manteniendo el mismo punto de trabajo previamente definido. Es decir, se han considerado los mismos valores para las articulaciones, las velocidades y las aceleraciones, tal como se indicó anteriormente. Recordemos que dicho punto de operación corresponde a q1=1, q2=2 , q3=0, mientras que las velocidades y aceleraciones asociadas a estas articulaciones se asumen nulas.

Estas matrices se han obtenido mediante los siguientes comandos en Matlab.

```
%Aumentamos G y H
G_aug = [G, zeros(6,2); -C, eye(2)];
H_aug = [H ; zeros(2,3)];
```

Tabla 7 Código MATLAB: Obtención de las matrices G y H aumentadas.

3.3.3 Controlabilidad y la matriz de controlabilidad del sistema extendido

Llegado a este punto, nuestro objetivo será calcular la controlabilidad de nuestro nuevo sistema ampliado.

Como sabemos, por definición la matriz de controlabilidad(Q) se define de la siguiente manera.

$$\hat{Q} = [\hat{H} \quad \hat{G}\hat{H} \quad \hat{G}^2\hat{H} \quad \hat{G}^3\hat{H} \quad \hat{G}^4\hat{H} \quad \hat{G}^5\hat{H} \quad \hat{G}^6\hat{H} \quad \hat{G}^7\hat{H}\ ]$$

Ecuación 41

La matriz obtenida en nuestro caso es el siguiente:

$$\hat{Q} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0.0001 & 0 & 0 & 0.0001 & 0 & 0 & 0.0001 & 0 & 0 & 0.0002 & 0 & 0 & 0.0002 & 0 & 0 & 0.0002 & 0 & 0 \\
0 & 0.0001 & -0.0001 & 0 & 0.0002 & -0.0002 & 0 & 0.0003 & -0.0003 & 0 & 0.0003 & -0.0003 & 0 & 0.0004 & -0.0004 & 0 & 0.0006 & -0.0006 & 0 & 0.0007 & -0.0007 & 0 & 0.001 & -0.001 \\
0 & -0.0001 & 0.0001 & 0 & -0.0002 & 0.0003 & 0 & -0.0003 & 0.0005 & 0 & -0.0003 & 0.0007 & 0 & -0.0004 & 0.0009 & 0 & -0.0006 & 0.0011 & 0 & -0.007 & 0.0013 & 0 & -0.001 & 0.0015 \\
0.0033 & 0 & 0 & 0.0033 & 0 & 0 & 0.0033 & 0 & 0 & 0.0033 & 0 & 0 & 0.0033 & 0 & 0 & 0.0033 & 0 & 0 & 0.0033 & 0 & 0 & 0.0033 & 0 & 0 \\
0 & 0.01 & -0.01 & 0 & 0.01 & -0.01 & 0 & 0.01 & -0.01 & 0 & 0.01 & -0.01 & 0 & 0.01 & -0.01 & 0 & 0.01 & -0.01 & 0 & 0.01 & -0.01 & 0 & 0.01 & -0.01 \\
0 & -0.01 & 0.02 & 0 & -0.01 & 0.02 & 0 & -0.01 & 0.02 & 0 & -0.01 & 0.02 & 0 & -0.01 & 0.02 & 0 & -0.01 & 0.02 & 0 & -0.01 & 0.02 & 0 & -0.01 & 0.02 \\
0 & 0 & 0 & 0 & 0 & 0 & -0.0001 & 0 & 0 & -0.0001 & 0 & 0 & -0.0003 & 0 & 0 & -0.0004 & 0 & 0 & -0.0006 & 0 & 0 & -0.0008 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -0.0001 & 0 & 0 & -0.0002 & 0 & 0 & -0.0004 & 0 & 0 & -0.0008 & 0 & 0 & -0.0012 & 0 & 0 & -0.0018 & 0 & 0 & -0.0024
\end{bmatrix}$$

Ecuación 42

Luego, sería verificar si nuestra nueva matriz es controlable. Al calcularlo con el comando "CTRB", que previamente hemos visto, podemos confirmar que nuestro sistema sigue siendo controlable. Este paso de comprobación es muy importante, ya que si no fuera controlable tendríamos que volver a realizar el proceso, pues en algo nos habríamos equivocado.

Esto se debe a que, si inicialmente nuestro sistema es controlable, el nuevo sistema también debería serlo.

El siguiente paso sería obtener la matriz de transformación que nos permitirá pasar a la forma canónica controlable.

3.3.4. Cálculo de la matriz de transformación a forma canónica controlable

Como nos hemos podido dar cuenta, la matriz aumentada ($\hat{Q}$) tiene una dimensión que no es cuadrada; de hecho, su tamaño es de 8x24. Esto no nos permitiría calcular su inversa y, por lo tanto, no podríamos obtener la matriz de transformación de forma sencilla, como es el caso de los sistemas SISO (Single Input Single Output).

En nuestro caso, al tratarse de un sistema MIMO (Multiple Input Multiple Output), debemos obtener una matriz a la que llamaremos L, que deberá tener el mismo orden que nuestro sistema ampliado y cuadrado, en la cual escogeremos columnas que sean LTI. A su vez, tendremos que determinar cuántos estados serán controlados por las entradas.

La pregunta que podría surgir es:

¿Por qué se hace esto en un sistema MIMO y no en uno SISO?

La respuesta radica en que, cuando solo tenemos una entrada y una salida, no es necesario dividir la matriz Q, ya que una única entrada controla todos los estados. Además, esta matriz normalmente es cuadrada en ese tipo de configuraciones.

La decisión de tener que determinar cuántos estados serán controlados por cada una de las entradas radica en que, al ser el sistema controlable, significa que puedo controlar cualquier estado a partir de la entrada. Y, claro, al tener un nuevo sistema aumentado, como es en nuestro caso, lo conveniente sería poder dividir equitativamente los estados entre las entradas.

Sin embargo, en nuestro caso tenemos ocho estados, ya que inicialmente eran seis, pero al añadir dos integradores, el sistema aumentado queda con ocho estados.

La distribución que se escogió no fue arbitraria, ya que desarrollamos un algoritmo que calcula todos los casos posibles para formar la matriz L. Esta matriz contendrá las columnas de la matriz ($\hat{Q}$), eligiendo cuántos estados serán controlados por las entradas u_1, u_2 y u_3.

El algoritmo que se realizó se muestra en la página siguiente.

```
combinaciones = nchoosek([1:24],8);%nchoosek nos
%da todas las combinaciones de 8 en 8 apiladas en

%fila en fila es decir todas las formas de poder
%combinar los números de 1 al 24 pero en 8 en 8
%es.

%Después de ello, creamos un vector vacio que
%contendrá todas las posibles combinaciones.

condiciones_L_desordenada = [];
```

```
for  i  =  1  :  size(combinaciones,1)%  Size(combinaciones,1)
%nos dice %cuántas filas tiene combinaciones
     i
% Lo que hacemos a continuación es seleccionar de la matriz
%Q_aug
%  todas  sus  filas,  pero  solo  las  columnas  indicadas  por
%combinaciones(i,:).
% Sabemos que combinaciones es una matriz de tamaño 734571x8.
% combinaciones(i,:) nos indica que, según la fila en la que
%nos
% encontremos, se tomarán todas sus columnas para esa fila i.
% Una vez tengamos eso, con Q_aug(:, [...]) obtendremos todas
%las filas
% de la matriz, pero únicamente las columnas que estén dentro
%de esos corchetes.
      L_desordenada = Q_aug( : , combinaciones(i,:) );
     condiciones_L_desordenada(i) = cond(L_desordenada);
 end
[valor_minimo,fila_minimo] = min(condiciones_L_desordenada);

combinaciones(fila_minimo,:);

columnas_u1 = 1 + 3*[0:1:10];
columnas_u2 = 2 + 3*[0:1:10];
columnas_u3 = 3 + 3*[0:1:10];
```

Tabla 8 Código MATLAB: Obtención de la matriz L.

En el código se define la variable L_desordenada, la cual contiene un subconjunto de columnas de la matriz $\widehat{Q}$ seleccionadas según las combinaciones generadas, sin mantener un orden específico. La matriz combinaciones proporciona las distintas selecciones de columnas posibles, pero no garantiza que estas estén ordenadas.

Se inicializa un vector vacío para almacenar los números de condición de cada submatriz L_desordenada generada. Posteriormente, mediante el uso de la función min, se obtiene el valor mínimo del número de condición junto con el índice (fila) correspondiente, que indica la combinación óptima de columnas.
Dado que la matriz L deseada debe estar ordenada y que esta se construye tomando columnas específicas de $\widehat{Q}$ indicadas por "combinaciones (i,:)", se utiliza el índice obtenido con min para acceder a la combinación de columnas que genera la submatriz mejor condicionada.
Finalmente, se crean tres vectores auxiliares que permiten identificar y clasificar los estados correspondientes a cada entrada, facilitando la organización y ordenamiento de las columnas en la matriz L.

Sin embargo, al aplicar este procedimiento nos encontramos con un problema importante: al comparar la matriz $\widetilde{\widetilde{H}}$ con la diferencia $\widetilde{\widetilde{G}}$ -$\widetilde{\widetilde{G}}_R$ observamos que no se anulan los elementos que deberían ser cero. Esto imposibilita el cálculo correcto de las matrices de realimentación K_cy K_i.

Este error se debe a que, en el algoritmo diseñado para seleccionar la matriz L con la mejor condición numérica, no se ha respetado el orden adecuado de selección de columnas. Específicamente, el algoritmo se ha centrado únicamente en minimizar el número de condición, sin considerar que la estructura de L debe mantener un orden coherente de izquierda a derecha, agrupando correctamente los estados asociados a cada entrada.

Esta falta de orden afecta directamente la construcción de la matriz de transformación y, en consecuencia, la transformación del sistema a su forma canónica controlable (CCF), haciendo que las matrices transformadas no presenten la estructura esperada. Por ello, se tomó la decisión de escoger de izquierda a derechas cuya única problemática es que la condición es mucho mayor que la mínima.

```
%L=[Q_aug(:,[1,13,22,2,11,3,9,15])] No elegimos esta %opción,
a pesar de que proporciona la menor condición %para
%L, porque la matriz H_aug_ccf que se obtiene al usar %T_inv,
calculada a partir de H, no tiene la forma %adecuada
%para aplicar las transformaciones necesarias
L=[Q_aug(:,[1,4,7,2,5,3,6,9])];  % Elección de Izquierda a
%Derecha. Condición muy grande.
%Invertimos la matriz L
L_cond=cond(L);% ¿Qué implicación tiene? Significa que va %ser
un poco más difícil, en términos de energía, %controlable pero
aún lo es.
L_inv=inv(L);
```

Tabla 9 Código MATLAB: Explicación del uso de otra combinación.

Como podemos observar la matriz L comentada es la que nos ha dado con el algoritmo y la matriz L sin comentar es la que vamos a usar ya que cumple con el criterio previamente explicado.

A continuación, procederemos a mostrar lo que nos ha dado por pantalla para ver el resultado.

$$L = \begin{bmatrix} 0.0000 & 0.0000 & 0.0001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0001 & 0.0002 & -0.0001 & -0.0002 & -0.0003 \\ 0 & 0 & 0 & -0.0001 & -0.0002 & 0.0001 & 0.0003 & 0.0005 \\ 0.0033 & 0.0033 & 0.0033 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0100 & 0.0100 & -0.0100 & -0.0100 & 0.0100 \\ 0 & 0 & 0 & -0.0100 & -0.0100 & 0.0200 & 0.0200 & 0.0200 \\ 0 & -0.0000 & -0.0001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.0001 & -0.0002 \end{bmatrix}$$

Ecuación 43

Antes de pasar a calcular la inversa deberíamos ver que efectivamente nuestra condición L no sea muy grande.

$L_{cond} = 4539{,}5$

Esta condición, a simple vista, parece incorrecta, ya que al ser muy grande podemos pensar que no es controlable; sin embargo, el sistema aún es controlable. No obstante, la implicación de que el valor de la condición sea muy grande es que será un poco más costoso controlarlo, en el sentido de que se requerirá más energía o esfuerzo para lograr el control deseado.

Una vez hecha la comprobación, procederemos a calcular su inversa, cuya forma deberá ser de la siguiente manera:

$$L^{-1} = \begin{bmatrix} l1 \\ l2 \\ l3 \\ l4 \\ l5 \\ l6 \\ l7 \\ l8 \end{bmatrix}$$

Ecuación 44

A continuación, se mostrará el resultado.

$$L^{-1} = 1x10^{4}.\begin{bmatrix} -4.5 & 0 & 0 & 0.053 & 0 & 0 & -3 & 0 \\ 6 & 0 & 0 & -0.03 & 0 & 0 & 6 & 0 \\ -1.5 & 0 & 0 & 0.01 & 0 & 0 & -3 & 0 \\ 0 & -2 & -1 & 0 & 0.03 & 0.02 & 0 & 0 \\ 0 & 2 & 1 & 0 & -0.01 & 0 & 0 & 0 \\ 0 & -1.5 & -1.5 & 0 & 0.02 & 0.02 & 0 & -1 \\ 0 & 2 & 2 & 0 & -0.01 & -0.01 & 0 & 2 \\ 0 & -0.5 & -0.5 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Ecuación 45

Para obtener la matriz de transformación inversa hacia la forma canónica controlable, se seguirá un procedimiento muy similar al aplicado en sistemas SISO. En esos casos, bastaba con tomar la última fila de la matriz inversa de controlabilidad, ya que solo existía una entrada y, por tanto, una única cadena de control. Sin embargo, en nuestro sistema MIMO, al haber múltiples entradas, es necesario considerar **varias agrupaciones** correspondientes a cada entrada.

Por ello, el procedimiento consiste en seleccionar la última fila de cada una de las agrupaciones elegidas dentro de la matriz de controlabilidad aumentada.

A partir de esas filas, se construyen vectores que se van multiplicando progresivamente por la matriz $\hat{G}$ (la matriz aumentada del sistema).
Esta secuencia de productos permite capturar el comportamiento dinámico de cada subsistema y, en conjunto, nos permite formar la matriz de transformación inversa ${T_c}^{-1}$, que refleja el grado de controlabilidad del sistema completo.

$$ {T_C}^{-1} = \begin{bmatrix} l3 \\ l3 * \hat{G} \\ l3 * \hat{G}^2 \\ l5 \\ l5 * \hat{G} \\ l8 \\ l8 * \hat{G} \\ l8 * \hat{G}^2 \end{bmatrix} $$ *Ecuación 46*

$$ {T_c}^{-1} = 1x10^4. \begin{bmatrix} -1.5 & 0 & 0 & 0.01 & 0 & 0 & -3 & 0 \\ 1.5 & 0 & 0 & -0.01 & 0 & 0 & -3 & 0 \\ 4.5 & 0 & 0 & 0.01 & 0 & 0 & -3 & 0 \\ 0 & 2 & 1 & 0 & -0.01 & -0.01 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0.01 & 0.01 & 0 & 0 \\ 0 & -0.5 & -0.5 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1.5 & 1.5 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} $$ *Ecuación 47*

Una vez llegados a este punto, debemos recordar que con esta matriz inversa podemos representar cualquier estado en la representación del estado correspondiente a la forma canónica controlable.
A continuación, se realizará una agrupación de términos para dejar todo más claro de aquí en adelante.

Partiremos de nuestro sistema aumentado.

$$ \hat{x}_{k+1} = \hat{G}\hat{x}_k + \hat{H}u_k + \begin{bmatrix} 0 \\ I \end{bmatrix} r_k $$

- r_k: $Valor\ de\ referencia (Lo\ quiero\ seguir)$.
- u_k: $Valor\ de\ entrada\ del\ sistema$.

La ecuación anteriormente mostrada está en lazo abierto, y eso como sabemos no nos es muy útil ya que no habría forma de poder corregir los errores que se presenten.

Si tenemos en cuenta que la señal de control tiene la siguiente forma.

$$u_k = -\hat{K}_C \hat{x}_k + K_i r_k \qquad \textit{Ecuación 48}$$

Siendo $\hat{K}_C$ lo siguiente:

$\hat{K}_C = [K_C + K_i C, -K_i]$

Una vez definido u_k ,procederemos a cerrar el lazo, obteniendo una expresión que permitirá corregir los errores.

$$\hat{x}_{k+1} = \hat{G}\hat{x}_k + \hat{H}(-\hat{K}_C \hat{x}_k + K_i r_k) + \begin{bmatrix} 0 \\ I \end{bmatrix} r_k$$

Agrupando las ecuaciones obtenemos lo siguiente:

$$\hat{x}_{k+1} = (\hat{G} - \hat{H}\hat{K}_C)\hat{x}_k + \left(\begin{bmatrix} 0 \\ I \end{bmatrix} + \hat{H}K_i\right) r_k$$

Acto seguido expresaremos la ecuación en su forma canónica controlable.

$$\hat{x}_{k+1} = T_c{}^{-1}(\hat{G} - \hat{H}\hat{K}_C)T_c \hat{x}_k + T_c{}^{-1}\left(\begin{bmatrix} 0 \\ I \end{bmatrix} + \hat{H}K_i\right) r_k \qquad \textit{Ecuación 49}$$

Podemos expresar esta ecuación en su forma estándar, empleando la nomenclatura definitiva, con el fin de estudiarla en los apartados siguientes.

$\tilde{\hat{x}}_{k+1} = \tilde{\hat{G}}_R \tilde{\hat{x}}_k + \tilde{\hat{H}}_R r_k$

Donde:

$$\tilde{\hat{H}}_R = T_c{}^{-1}\begin{bmatrix} H * K_i \\ I \end{bmatrix} \qquad \textit{Ecuación 50}$$

$$\tilde{\hat{G}}_R = T_c{}^{-1}(\hat{G} - \hat{H}\hat{K}_C)T_c \qquad \textit{Ecuación 51}$$

La matriz $\tilde{\hat{G}}_R$, obtenida mediante los pasos anteriormente descritos, será clave para que el regular mantenga las salidas del sistema dentro de los requerimientos del diseño.

En las siguientes líneas, procederemos a mostrar las expresiones de las matrices en su forma canónica controlable, al igual que el cálculo de estas para nuestro caso.

$$\tilde{\hat{G}} = {T_C}^{-1} * \hat{G} * T_C$$

$$\tilde{G} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -0.0003 & 0.0003 & 0 & -0.0010 & 1 & -3.0012 & 3.0012 \end{bmatrix}$$

Ecuación 52

$$\tilde{\hat{H}} = {T_C}^{-1} * \hat{H}$$

$$\hat{H} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 53

A continuación, se mostrar los pasos realizados en Matlab para la obtención de esas matrices.

```
% Obtenemos la Tc_inv
Tc_inv=[L_inv(3,:);L_inv(3,:)*G_aug;L_inv(3,:)*G_aug*G_aug;L
_inv(5,:);L_inv(5,:)*G_aug;L_inv(8,:);L_inv(8,:)*G_aug;L_inv
(8,:)*G_aug*G_aug];
% Pasamos a la FORMA CANÓNICA CONTROLABLE
G_aug_ccf=Tc_inv*G_aug*inv(Tc_inv);
H_aug_ccf=Tc_inv*H_aug;
```

Tabla 10 Código MATLAB: Pasando a la Forma Canónica Controlable.

Con esto hemos conseguido obtener nuestro sistema extendido en la forma canónica controlable. Nuestro siguiente paso será calcular el polinomio

característico necesario para que nuestro sistema responda a las especificaciones que proporcionaremos en la salida.

3.3.5 Polinomio característico requerido del sistema extendido

En esta sección se procederá al cálculo del polinomio deseado, cuyo objetivo principal es permitir el control de un robot redundante con tres grados de libertad (G.D.L.), garantizando además que la señal de salida cumpla con un conjunto de especificaciones previamente definidas.

Los criterios que deseamos cumplir se verán reflejados en el polinomio característico deseado, el cual estará determinado por las características que detallaremos a continuación:

- Sobre oscilación de las señales (M_p). Para el caso que nos ocupa, su valor será de un 20%.
- Tiempo de establecimiento de las señales (t_s). En nuestro caso, el tiempo será de 2 segundos.

Para poder satisfacer esos requerimientos, lo que se debe hacer es utilizar las siguientes formulas:

$$t_s = \frac{\pi}{\sigma}$$ *Ecuación 54*

$$M_p = e^{-\frac{\pi}{\tan\theta}}$$ *Ecuación 55*

Estas fórmulas las usaremos para poder obtener los polos que nos dará el polinomio característico que estamos buscando.

Los polos en el caso de los sistemas sub-amortiguados se obtienen de la siguiente manera.

$Pd = -\sigma \pm \sigma * \tan\theta\, i$

A continuación, se mostrarán los comandos utilizados en MATLAB, así como los valores numéricos empleados en nuestro caso de estudio.

```
%% Sub-amortiguado
%Asumiré un Mp=20% and ts=2
% Deseado G
sigma=pi/ts;
Mp=0.2;
tang_teta=-pi/log(Mp);
s1_b=-sigma + sigma*tang_teta*1i;
s2_b=-sigma - sigma*tang_teta*1i;
s1_b_dis=exp(s1_b*Ts);
s2_b_dis=exp(s2_b*Ts);
```

Tabla 11 Código MATLAB: Definición de los polos deseados.

- $\sigma = 1.5708\ rad.$
- $\tan\theta = 1.9520.$

Una vez alcanzado este punto, se puede utilizar la expresión previamente definida para representar los polos en su formato estándar.

$$Pd_{continuos} = -1.5708 \pm 1.5708 * 1.9520i$$ *Ecuación 56*

La siguiente problemática que se presenta es que estamos trabajando en un sistema discreto, mientras que los polos obtenidos se encuentran en el dominio continuo. Por lo tanto, el siguiente paso consiste en discretizarlos, lo cual se realizará mediante la siguiente ecuación:

$$Pd_{discretos} = e^{Ts*Pd}$$ *Ecuación 57*

En Matlab hemos obtenido los polos discretos de la siguiente manera.

```
s1_b_dis=exp(s1_b*Ts);
s2_b_dis=exp(s2_b*Ts);
```

Tabla 12 Código MATLAB: Obtención de los polos discretos.

Su forma final de polinomio realimentado ya discretizado será de la forma siguiente:

$$PR(z) = \alpha_2 z^2 + \alpha_1 z + \alpha_0$$ *Ecuación 58*

Los valores de los coeficientes son:

$\alpha_0 = 0.9691$

$\alpha_1 = -1.96979$

$\alpha_2 = 1.0000$

A continuación, se mostrará el script que se ha ejecutado para la obtención de los coeficientes.

```
%Calcularemos los coeficientes del polinomio deseado ya
%discretizado.
coef = poly([s1_b_dis,conj(s1_b_dis)]);
alfa2 = coef(1);
alfa1 = coef(2);
alfa0 = coef(3);
```

Tabla 13 Código MATLAB: Obtenemos los valores de los coeficientes del polinomio deseado.

Los coeficientes del polinomio característico discretizado son importantes, ya que se utilizarán posteriormente para obtener una matriz cuya función es calcular las constantes de control Kc y Ki.

3.3.6 Diseño de las matrices $\tilde{\tilde{G}}_R$ y $\tilde{\tilde{H}}_R$

En las secciones 3.3.4 y 3.3.5 hemos logrado obtener el sistema extendido expresado en Forma Canónica Controlable (C.C.F.), así como el polinomio característico deseado.

Pese a que nuestro sistema aún no cumple del todo con nuestras expectativas en cuanto al sobreoscilamiento, debido a que todavía no hemos aplicado el polinomio característico discretizado, ya contamos con los elementos necesarios para calcular los parámetros del regulador (Kc y Ki).

Teniendo en cuenta que disponemos de la matriz $\tilde{\tilde{G}}_R$, la cual, como hemos visto anteriormente, es fundamental para los cálculos de Kc y Ki, y que también contamos con la matriz $\tilde{\tilde{H}}_R$. En realidad, si analizamos nuestro sistema, solo nos haría falta obtener $\tilde{\tilde{G}}_R$ para poder obtener los coeficientes tanto de $\tilde{K}_C \; como \; de \; \tilde{K}_i$.Todo esto lo podremos conseguir mediante el polinomio característico previamente diseñado.

El primer paso será determinar qué forma le daremos a nuestra matriz $\tilde{\tilde{G}}_R$, ya que, como vimos anteriormente, el sistema cuenta con tres entradas. También debemos tener en cuenta que disponemos de ocho estados, por lo que la matriz $\tilde{\tilde{G}}_R$ tendrá una dimensión de 8×8.

La siguiente pregunta que puede surgir es: ¿cómo dividimos esta matriz? La respuesta es que no existe una única forma de hacerlo. Sin embargo, lo más lógico es respetar el orden previamente establecido, ya que lo que puede pasar es que tengamos una mala respuesta en la salida o que no podamos obtener los parámetros Kc y Ki. En nuestro caso, habíamos acordado una división de los estados en tres bloques: 3, 2 y 3. Por tanto, organizaremos la matriz siguiendo esa estructura.

Para hacernos una idea, a continuación, se mostrará una plantilla con la división elegida.

$$\tilde{\tilde{G}}_R = \begin{bmatrix} \begin{matrix} 1 & \\ & 1 \\ xxxx & xxxx\ xxxx \end{matrix} & & \\ & \begin{matrix} 1 \\ xxxx\ xxxx \end{matrix} & \\ & & \begin{matrix} 1 & \\ & 1 \\ xxxx & xxxx\ xxxx \end{matrix} \end{bmatrix}$$

Ecuación 59

Donde las "XXXX" representan los coeficientes del polinomio característico.
Llegados a este punto, podemos preguntarnos: ¿Con qué valor rellenamos el tercer polo de cada división si solo disponemos de dos polos en dicho polinomio?
La respuesta es sencilla: podemos completarlo con un Z=0, con el objetivo de que este no afecte a nuestro sistema ya que se extinguirá lo más rápido posible.
Además, es importante tener en cuenta que todos los elementos que no se encuentran dentro de los recuadros establecidos serán igualados a cero, manteniendo así la estructura deseada para la matriz.

A continuación, se mostrará la matriz con los datos de nuestro sistema

$$\tilde{\tilde{G}}_R = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.97 & 1.97 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.97 & 1.97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.97 & 1.97 \end{bmatrix}$$

Ecuación 60

Como podemos observar hemos replicado los polos en cada división con el objetivo de que el comportamiento se acerque al deseado y como hemos dicho antes hemos puesto un polo en z=0 para que este tenga el menor efecto posible.

Una vez obtenido la matriz $\tilde{\tilde{G}}_R$ pasaremos a hacer los despejes necesarios en la Ecuación 51 para poder obtener la expresión de $\tilde{\tilde{K}}_c$ para posteriormente obtener los parámetros K_C y K_i.

3.3.7 Cálculo de K_C y K_i

Llegados a este punto pasaremos a indicar las dimensiones de las matrices con las cuales estaremos trabajando, ya que es importante a la hora de solucionar esta ecuación.

$$\tilde{\tilde{G}}_{R_{8x8}} = \tilde{\tilde{G}}_{8x8} - \tilde{\tilde{H}}_{8x3}\tilde{\tilde{K}}_{C_{3x8}}$$

Pasamos al lado izquierdo el producto $\tilde{\tilde{H}}_{8x3}\tilde{\tilde{K}}_{C_{3x8}}$ y al lado derecho la matriz $\tilde{\tilde{G}}_{R_{8x8}}$ quedando la ecuación de la siguiente manera.

$$\tilde{\tilde{H}}_{8x3}\tilde{\tilde{K}}_{C_{3x8}} = \tilde{\tilde{G}}_{8x8} - \tilde{\tilde{G}}_{R_{8x8}}$$

Como podemos ver a simple vista tenemos un gran problema. Si queremos obtener $\tilde{\tilde{K}}_c$ necesitamos calcular la inversa de $\tilde{\tilde{H}}_{8x3}$,y como podemos observar, no es cuadrada, lo que imposibilita obtener su inversa. Sin embargo, si recordamos que la matriz $\tilde{\tilde{H}}_{8x3}$ está en la forma canónica controlable, esto significa que los ceros presentes en sus filas y/o columnas no afectan directamente a la dinámica del sistema. Por consiguiente, este hecho elimina el problema anteriormente planteado.

Pese a lo hecho anteriormente, seguimos encontrándonos con otra problemática, ya que la diferencia de la matriz $\tilde{\tilde{G}} - \tilde{\tilde{G}}_R$ nos dará una matriz cuadra de dimensión ocho por ocho. Esto imposibilita que sea multiplicado por la inversa de $\widetilde{\widetilde{H}}$.

$$\tilde{\tilde{G}} - \tilde{\tilde{G}}_R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2.03 & 1.03 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.031 & 0.032 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2.031 & 1.032 \end{bmatrix}$$

Sin embargo, pasa lo mismo que en el caso anterior. Si prestamos atención, la diferencia está en su forma canónica controlable. Por lo tanto, podemos eliminar los ceros de las filas y/o columnas con el objetivo de volverla cuadrada. Esto que estamos haciendo ha sido posible debido a que la composición de $\tilde{\tilde{G}}$ es similar a la que hemos impuesto a la matriz $\tilde{\tilde{G}}_R$.

A continuación, se mostrará la ecuación con las dimensiones después de haber suprimido las filas de ceros, tanto en $\widetilde{\widetilde{H}}$ como en la resta $\tilde{\tilde{G}} - \tilde{\tilde{G}}_R$.

$$\widetilde{\widetilde{H}}_{3x3}\widetilde{\widetilde{K}}_{C_{3x8}} = (\tilde{\tilde{G}} - \tilde{\tilde{G}}_R)_{3x8}$$

Pasamos al lado derecho la matriz $\widetilde{\widetilde{H}}_{3x3}$ y obtenemos $\widetilde{\widetilde{K}}_{C_{3x8}}$.

$$\widetilde{\widetilde{K}}_{C_{3x8}} = (\widetilde{\widetilde{H}}_{3x3})^{-1} * (\tilde{\tilde{G}} - \tilde{\tilde{G}}_R)_{3x8} \qquad \textit{Ecuación 61}$$

$$\tilde{\tilde{K}}_C = \begin{bmatrix} 1.0000 & -2.0309 & 1.0321 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0309 & 0.0321 & -0.0000 & 0.0000 & 0 \\ 0 & 0 & 0 & -0.0000 & 0.0000 & 1.0000 & -2.0309 & 1.0321 \end{bmatrix} \qquad \textit{Ecuación 62}$$

Ahora que tenemos la matriz $\widetilde{\widehat{K}}_c$, debemos multiplicarla por la inversa de la matriz de transformación (${T_C}^{-1}$) que hemos ido usando, con el fin de revertir la C.C.F.

$$\widehat{K}_{C_{3x8}} = \widetilde{\widehat{K}}_{C_{3x8}} \, {T_{C_{8X8}}}^{-1}$$

Dándonos como resultado:

$$\widehat{K}_C = \begin{bmatrix} 980.4012 & 0 & 0 & 304.7268 & 0 & 0 & -35.0493 & 0 \\ 0 & 23.3662 & 11.6831 & 0 & 6.3023 & 3.1512 & 0 & -0.0000 \\ 0 & 326.8004 & 326.8004 & 0 & 101.5756 & 101.5756 & 0 & 11.6831 \end{bmatrix} \quad \textit{Ecuación 63}$$

Como parte de los pasos finales de este apartado, al retomar la ecuación que define nuestra nueva entrada del sistema discreto (Ecuación 48) y sustituirle en ella el valor de $\widehat{K}_{C_{3x8}}$, podemos obtener el valor de K_C y K_i.

$$\widehat{K}_C = [K_i C \; + \; K_C, -K_i] \quad \textit{Ecuación 64}$$

Es importante comprender qué representa la matriz $\widehat{K}_C$. Cada una de sus filas corresponde a una de las entradas del sistema, las cuales han sido previamente linealizadas (Δu_1 , Δu_2 y Δu_3), dado que el sistema original es no lineal.
Por otro lado, las columnas representan el total de estados del sistema: las primeras seis están asociadas a las variables de estado originales, mientras que las dos últimas corresponden a los integradores introducidos durante el diseño.

Por este motivo, los valores que se obtienen son los siguientes:

$$K_i = \begin{bmatrix} 35.0493 & 0 \\ 0 & 0.0000 \\ 0 & 11.6831 \end{bmatrix} \quad \textit{Ecuación 65}$$

$$K_C = \begin{bmatrix} 945.3519 & 0 & 0 & 304.7268 & 0 & 0 \\ 0 & 23.3662 & 11.6831 & 0 & 6.3023 & 3.1512 \\ 0 & 315.1173 & 315.1173 & 0 & 101.5756 & 101.5756 \end{bmatrix}$$

Ecuación 66

Con esto, hemos logrado que la salida de nuestro sistema se comporte conforme a una serie de especificaciones establecidas, gracias a la acción combinada de los reguladores, tanto el integral como la retroalimentación de estados que hemos implementado.

Como último paso, procederemos a mostrar los scripts completos que hemos utilizado para obtener todos los cálculos de los valores que hemos presentado.

```
p1 = poly([s1_b_dis, s2_b_dis,0]);  % El polo s3  puede ser % 
0 o lo que elijas. En esta ocasión elegiremos cero, debido % 
a que este no tendrá efecto en la salida.
p2 = poly([s1_b_dis, s2_b_dis]);
primero=[0 1 0 zeros(1,5);0 0 1 zeros(1,5);-p1(4) -p1(3) -
p1(2) zeros(1,5)];
segundo=[zeros(1,3) 0 1 zeros(1,3);zeros(1,3) -p2(3) -p2(2) 
zeros(1,3)];
tercero=[zeros(1,5) 0 1 0;zeros(1,5) 0 0 1;zeros(1,5) -p1(4) 
-p1(3) -p1(2)];
G_r_aug_ccf_def=[primero;segundo;tercero];
% Ahora haremos la diferencia
jd=G_aug_ccf-G_r_aug_ccf_def;
H_aug_ccf=Tc_inv*H_aug;
%% Obtener las K
% producto=H_aug_ccf * k
k_aug_cff=jd([3,5,8],:);
k_aug=k_aug_cff*Tc_inv;
ki=-k_aug(:,end-1:end);% Con esto el error de posición es nulo
aux=k_aug(:,1:end-2);
kc=aux-ki*C;
```

Tabla 14 Código MATLAB: Obtención de los controladores K_c y K_i.

A continuación, procederemos a implementar el esquema en Simulink para poder ver si nuestro diseño cumple en simulación lo que significaría que sí satisfacemos los criterios establecidos.

3.3.8 Implementación en el esquema y simulación

Una vez calculadas las dos matrices que componen los reguladores, procederemos a implementarlas en el esquema de Simulink.

Los datos obtenidos se han realizado teniendo en cuenta la Tabla 2.

3.3.8.1 DISEÑO DEL CONTROLADOR DISCRETO LINEALIZADO

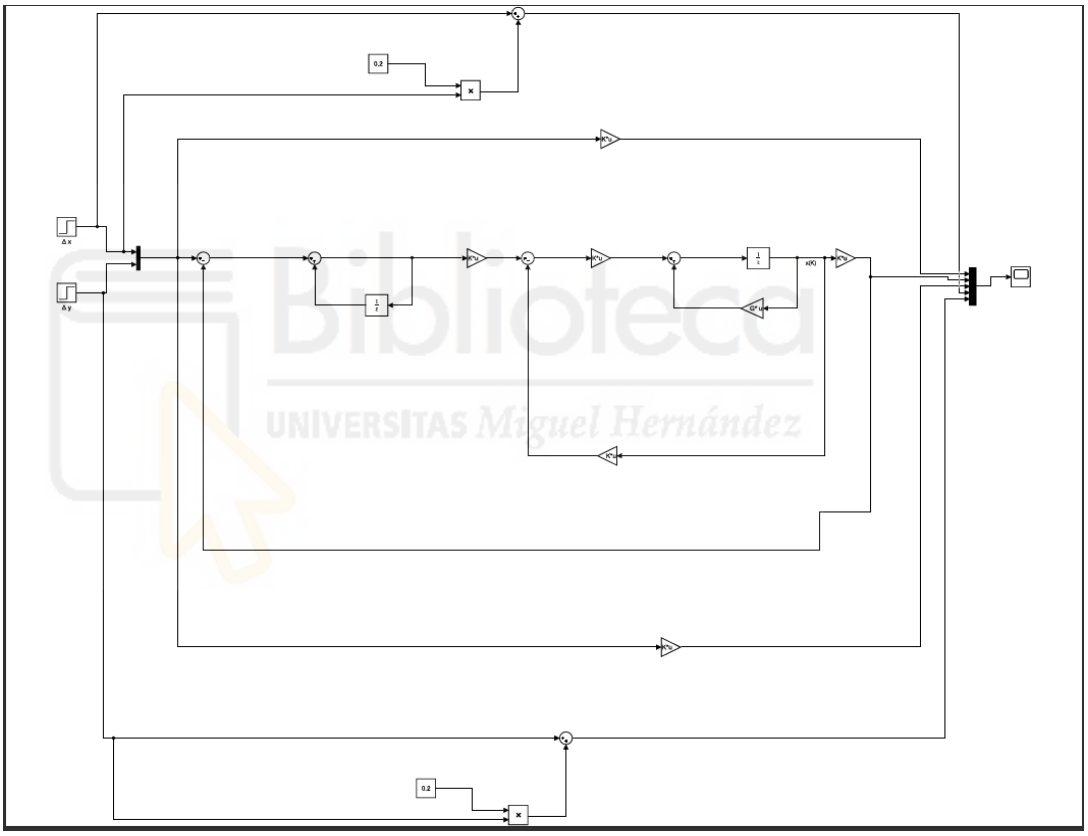


Ilustración 5 Diagrama de bloques de la implementación del controlador linealizado discretizado.

Una vez presentada la gráfica, a continuación, se detallan los parámetros empleados en la simulación.

- Incremento deseado respecto a la posición de equilibrio inicial en la coordenada "x" (m): 0.5
- Incremento deseado respecto a la posición de equilibrio inicial en la coordenada "y" (m): 1
- Coordenada "x" inicial del efector x0 (m): $q_1 + \cos q_3$
- Coordenada "y" inicial del efector y0 (m): $q_2 + \sin q_3$
- Aceleración gravitatoria g (m/s^2): 10
- Masa de las barras (Kg): 1
- Sobre oscilación de las salidas Mp (%): 20
- Tiempo de establecimiento de las señales ts (s): 2
- Periodo de muestreo para la discretización del sistema Ts (s): 0.01

Dado que es necesario verificar el cumplimiento del tiempo de establecimiento y de la sobre oscilación, se añadirán marcadores en las gráficas con el único propósito de facilitar su análisis. Para ello, se debe añadir lo siguiente:

- Para medir fácilmente la sobre oscilación debemos hacer lo siguiente: $\Delta x + \Delta x * 0.2$ e $\Delta y + \Delta y * 0.2$.
- Para calcular la franja del tiempo de establecimiento: $(\Delta x) * 0.95\ y\ (\Delta x) * 1.05$.

Debemos tener en cuenta que tanto Δx como Δy representan las referencias a seguir para x e y.

A continuación, se procederá a mostrar la salida del DISEÑO DEL CONTROLADOR DISCRETO LINEALIZADO teniendo en cuenta que la articulación $q_3 = \frac{\pi}{4}$.

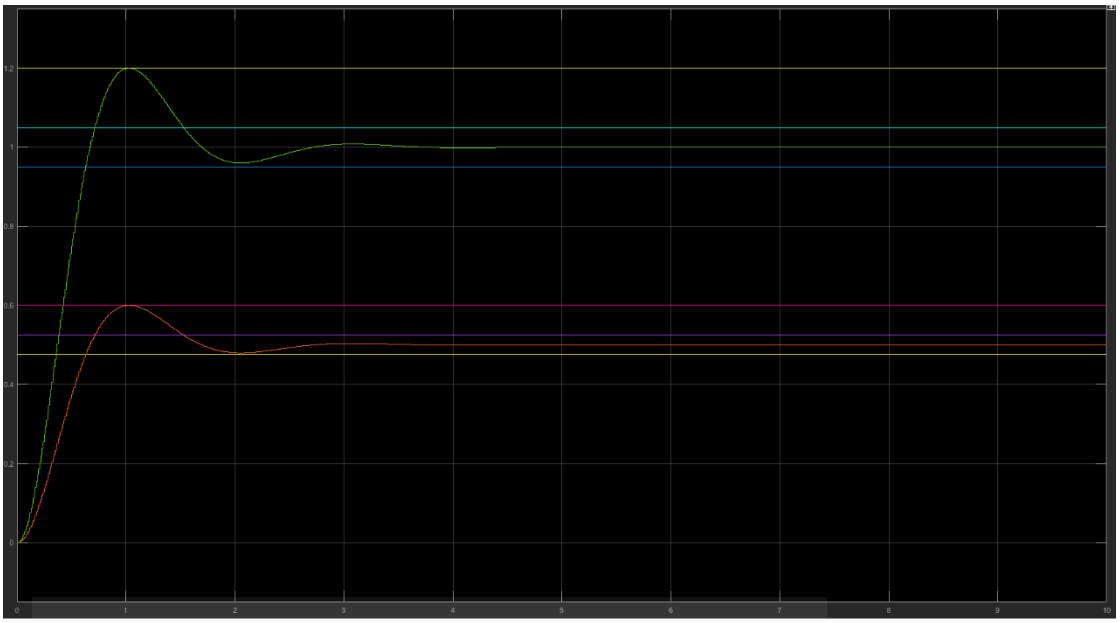


Ilustración 6 Señal de salida de nuestro sistema tras implementar nuestro regulador.

Los resultados obtenidos indican que el esquema propuesto satisface los criterios de diseño especificados. La magnitud del sobre oscilamiento de cada señal de entrada no excede el 20% establecido, y el sistema alcanza el régimen permanente antes de ts=2 , manteniéndose dentro del umbral de ±5% respecto al valor final. En consecuencia, se puede afirmar que el diseño linealizado y discretizado del sistema cumple adecuadamente con los requisitos de desempeño dinámico.

El siguiente objetivo planteado consiste en analizar el sistema continúo utilizando los mismos valores calculados en el sistema discretizado y observar si tiene el mismo comportamiento.

3.3.8.2 PRUEBA DEL CONTROL DISCRETO SOBRE EL CONTINUO LINEALIZADO

En este caso como queremos usar los mismos datos obtenidos en el sistema discreto, primero tenemos que pasar esos valores del sistema discreto al continuo mediante:

- Muestreador: Su función principal es discretizar la señal, para lo cual se les debe especificar el tiempo de muestreo Ts. Como MATLAB no dispone de un bloque específico de muestreador, se implementa utilizando un bloque de función de transferencia, asignando el valor 1 tanto al numerador como al denominador.
- Bloqueadores de orden 0: Función principal es la de pasar de un sistema discreto a continuo.

A continuación, se procederá a mostrar el esquema en Simulink con lo comentado anteriormente.

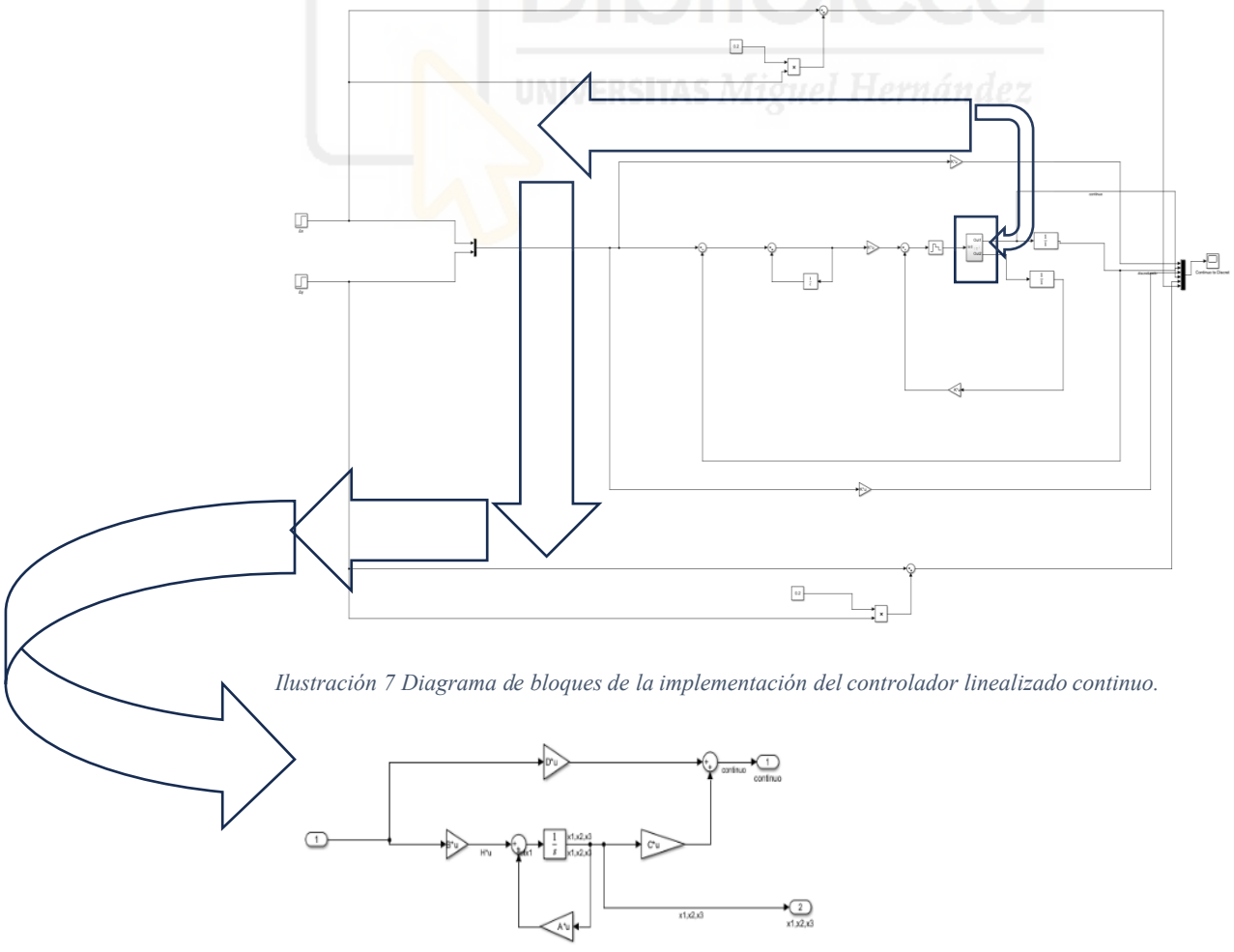


Ilustración 7 Diagrama de bloques de la implementación del controlador linealizado continuo.

Al observar Ilustración 7, podemos notar que, tal como se había anticipado, se ha colocado un bloqueador de orden cero antes de las matrices continua. Esto se debe a que, dado que todo nuestro sistema está diseñado para operar en el sistema discreto, al realizar operaciones con elementos en continuo es necesario convertirlos mediante el bloqueador de orden cero. Asimismo, se han colocado dos muestreadores, uno en la salida de la ecuación de estado y otro en las variables de estado, con el fin de no alterar los resultados previamente calculados y asegurar el correcto funcionamiento del sistema.

A continuación, se mostrará la salida del esquema en Simulink visto anteriormente con un tiempo de simulación de 10 segundos.

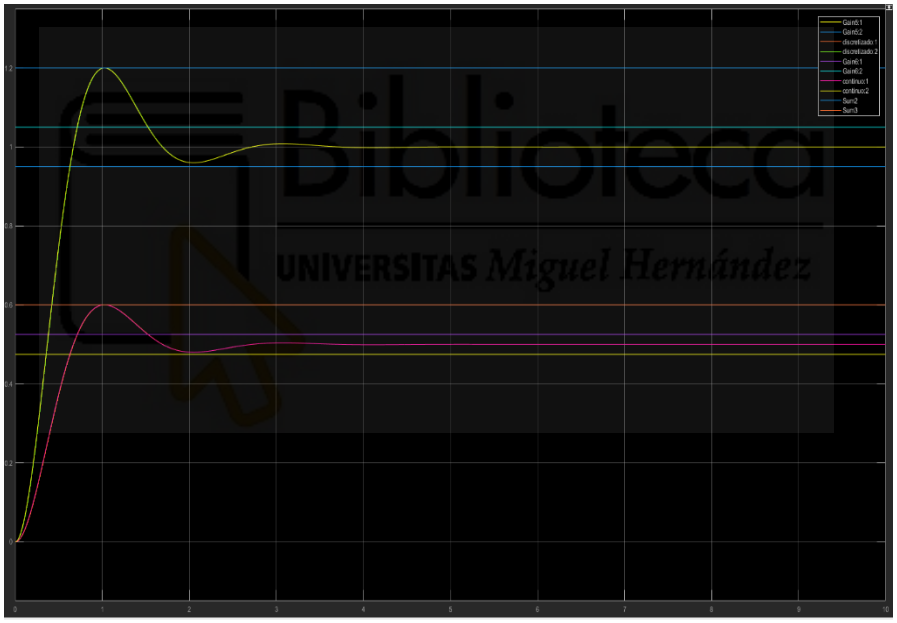
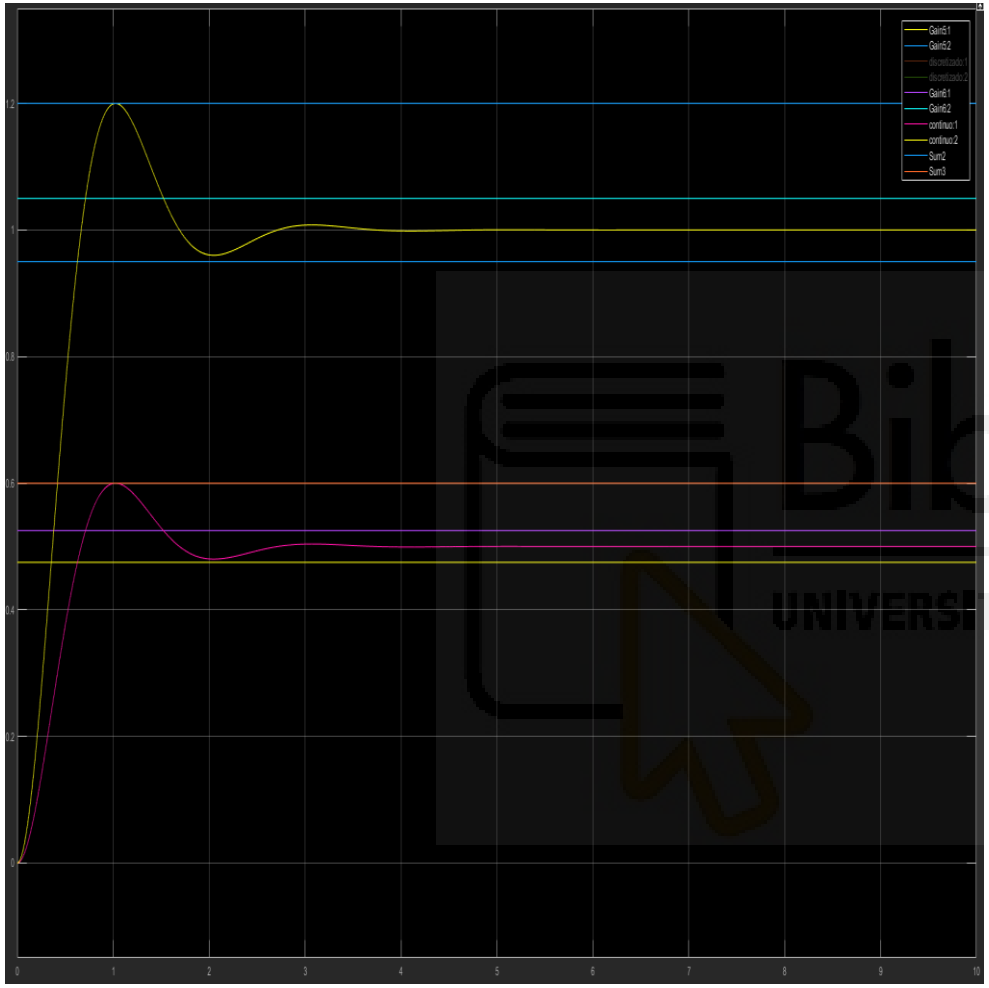


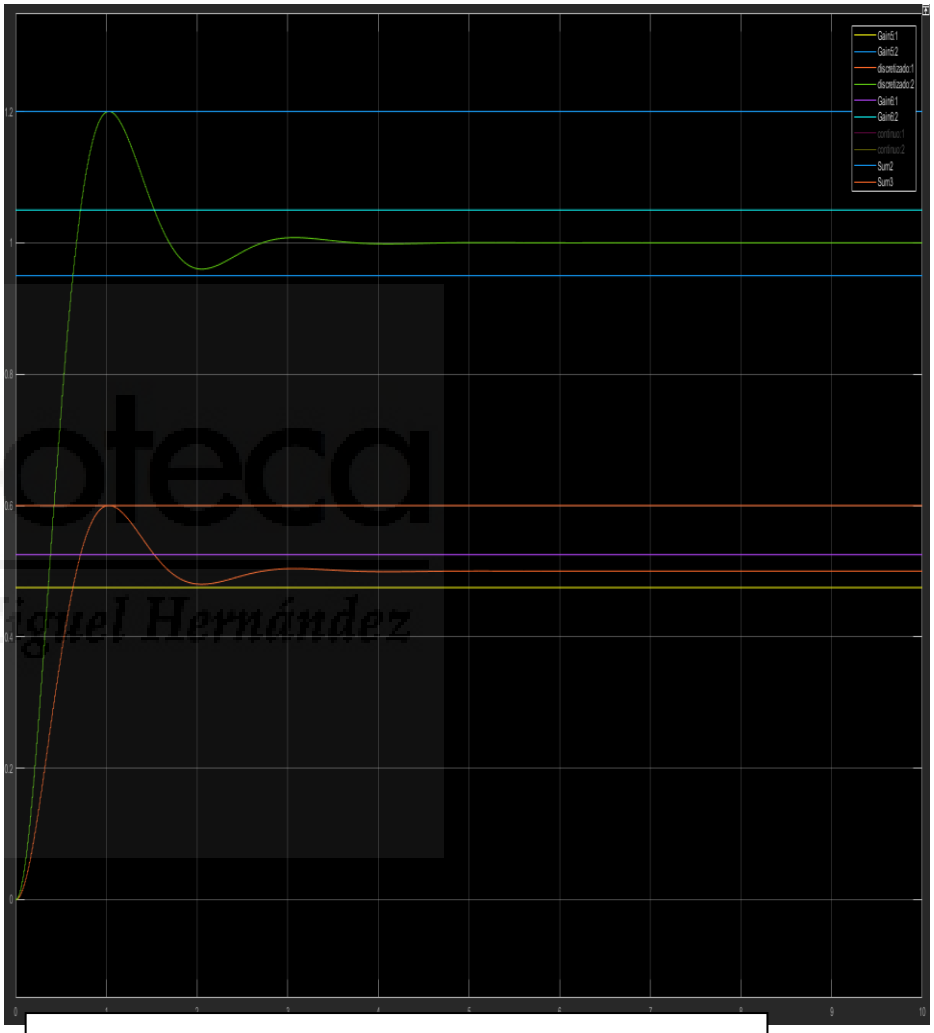
Ilustración 8 Comparación entre el sistema continuo/discreto linealizado.

En la imagen adjunta se puede observar cómo el sistema continúo linealizado se comporta de manera prácticamente idéntica al sistema discreto. En la esquina superior derecha se encuentra una leyenda que indica claramente qué color corresponde a cada uno de los sistemas. Como se puede apreciar, las diferencias entre ambas respuestas son prácticamente imperceptibles.

También se mostrará las gráficas por separado en la siguiente página con el fin de observarlas mejor.



SISTEMA CONTINUO LINEALIZADO



SISTEMA DISRETIZADO LINEALIZADO

Una vez analizado el comportamiento de los sistemas linealizados, tanto en su versión continua como discreta, procederemos a estudiar el comportamiento del sistema original, es decir, el no lineal. Para ello, se ha implementado una función en Simulink que permite simular su dinámica.

3.3.8.3 SISTEMA NO LINEAL DISCRETO

En este apartado, como ya se ha mencionado, debemos crear una función en **Simulink** mediante el bloque *Function*, al que en nuestro caso llamaremos **Sistema no lineal**, donde se introducirán todas las ecuaciones no lineales y se despejará la variable de aceleración. Esta variable será necesaria posteriormente para, a partir de ella, obtener tanto la velocidad como la posición.

El bloque en cuestión tiene esta forma:

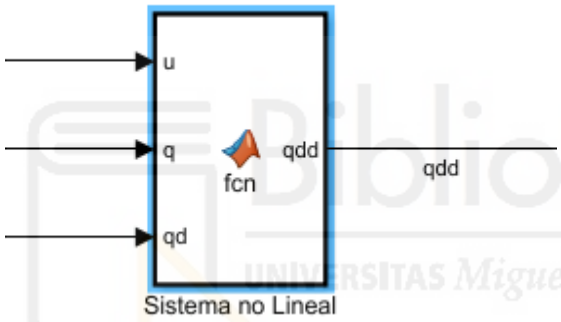


Ilustración 9 Bloque FUNCTION en Simulink.

Seguidamente, se mostrará el código empleado en dicha función.

```
function qdd = fcn(u,q,qd)
m1=1;
m2=1;
m3=1;
g=10;
q1=q(1);
q2=q(2);
q3=q(3);
dq1=qd(1);
dq2=qd(2);
dq3=qd(3);
u1=u(1);
u2=u(2);
u3=u(3);
```

```
% Matriz de inercia M(q3)
M = [m1 + m2 + m3,      0,              -m3*sin(q3);
      0,                m2 + m3,        m3*cos(q3);
     -m3*sin(q3),       m3*cos(q3),     m3];

% Vector b(q3, dq3)
b = [ m3*dq3^2*cos(q3) + u1;
       m3*dq3^2*sin(q3) - (m2 + m3)*g + u2;
      -m3*g*cos(q3) + u3];



% Cálculo de aceleraciones
qdd = inv(M)*b; % tener cuidado con el orden.
```

Tabla 15 Código MATLAB: Definición de las ecuaciones No-Lineales.

De igual forma, con las salidas ya que estas también son no lineales, con la pequeña diferencia que le entra las articulaciones, es decir, depende directamente de estas.

Es por ello, que tiene la siguiente forma:

Ilustración 10 Bloque Salidas No-Lineal

Y su código en MATLAB es el siguiente:

```
function y = fcn(q)
q1 = q(1);
q2 = q(2);
q3 = q(3);
y1=q1+cos(q3);
y2=q2+sin(q3);
y = [y1; y2];
```

Tabla 16 Código MATLAB: Definición de las ecuaciones No-Lineales de las salidas.

Como se puede apreciar en la Tabla 15 adjunta, a la función se le pasan como parámetros u, q y qd, que corresponden a las entradas, la posición y la velocidad, respectivamente. Es importante recordar que cada uno de estos parámetros es un vector de tres filas y una columna.

Mediante las ecuaciones del sistema, es necesario despejar la aceleración. De esta manera, podremos retroalimentar progresivamente las variables q y qd durante la simulación.

Además, como estamos trabajando con un sistema no lineal, mientras que los cálculos previos se realizaron sobre un modelo linealizado, debemos realizar ajustes adecuados. En particular:

- Para las **entradas**, dado que queremos trabajar con valores absolutos, se debe **sumar la entrada en el punto de equilibrio**.
- Para los **estados** (posición y velocidad), se debe **restar el punto de equilibrio** para que en la salida nos dé el incremento y de ese modo poder realimentar el sistema.
- Y para la **salida**, se aplica el mismo criterio que para los estados: se debe **restar el punto de equilibrio** respecto al punto de operación para obtener el incremento correctamente y así poder ver si nuestro sistema No-Lineal se comporta igual que nuestro sistema lineal al menos cerca del punto de funcionamiento.

Para calcular el valor en el punto de funcionamiento, utilizaremos nuevamente la herramienta **DERIVE**. En esta ocasión, aplicaremos la Ecuación 3 y **accederemos al apartado** de sustituir(sub). Luego, asignaremos el valor de **cero a todas las derivadas**, ya que nuestro objetivo es obtener las variables en su punto de equilibrio.

A continuación, se mostrarán ilustraciones que respaldan y aclaran lo explicado anteriormente.

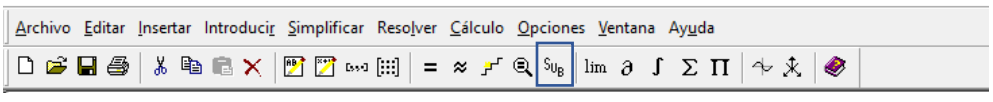


Ilustración 11 DERIVE: Interfaz.

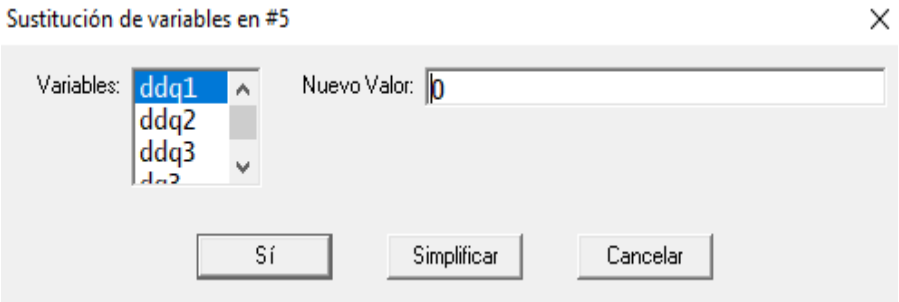


Ilustración 12 DERIVE: Sub.

Como se puede apreciar se sustituye los valores en el punto de funcionamiento con el fin de calcular las **entradas correspondientes a ese punto**.

Una vez hecho esto, le damos al botón de simplificar y obtenemos los valores de las entradas en el punto de funcionamiento.

- $u_1 = 0.$
- $u_2 = g(m_2 + m_3).$
- $u_3 = g.m_3.\cos q_3$

De igual forma tenemos que calcular el valor en punto de funcionamiento de los estados y salidas.

Tal como se indicó previamente, el sistema cuenta con tres eslabones, y a cada uno le corresponden dos variables: posición articular y su derivada temporal (velocidad articular).

$$x_0 = \left[q_{1_0}, q_{2_0}, q_{3_0}, dq_{1_0}, dq_{2_0}, dq_{3_0}\right]^T$$

En este estudio se ha considerado que el robot opera en condiciones de velocidad nula, lo que implica que todas las velocidades lineales iniciales son cero. Esta suposición permite simplificar notablemente el análisis del sistema, ya que se elimina el término dinámico asociado al movimiento.

De este modo las variables de estado, en el punto de funcionamiento, quedan de la siguiente forma:

$$x_0 = \left[q_{1_0}, q_{2_0}, q_{3_0}, 0,0,0\right]^T$$

Ahora lo que nos falta obtener son los valores de las articulaciones en el punto de funcionamiento, es decir, en condiciones iniciales.

Estos valores los podemos obtener , mediante las Ecuación 1 y Ecuación 2:

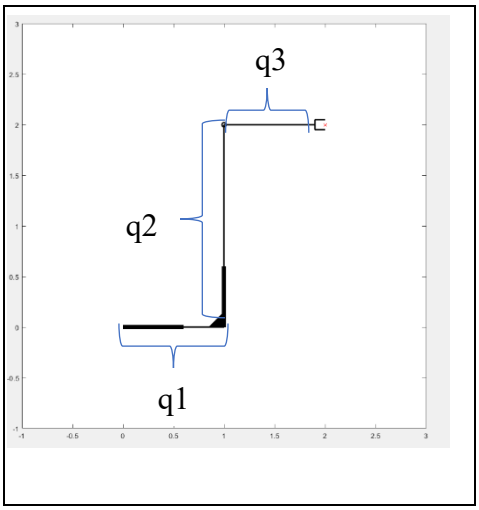


Ilustración 13 Gráfica del robot.

Observando la gráfica adjunta podemos observar en qué posición se encuentra la pinza y de este modo encontraríamos el punto de funcionamiento de la salida que para nuestro caso sería de:

$$y_0 = [2,2]^T$$

Una vez obtenidos todos estos datos, procederemos a completar el esquema en Simulink de la parte no lineal, añadiendo bloques de suma o diferencia según corresponda para obtener los incrementos, quedándonos del siguiente modo:

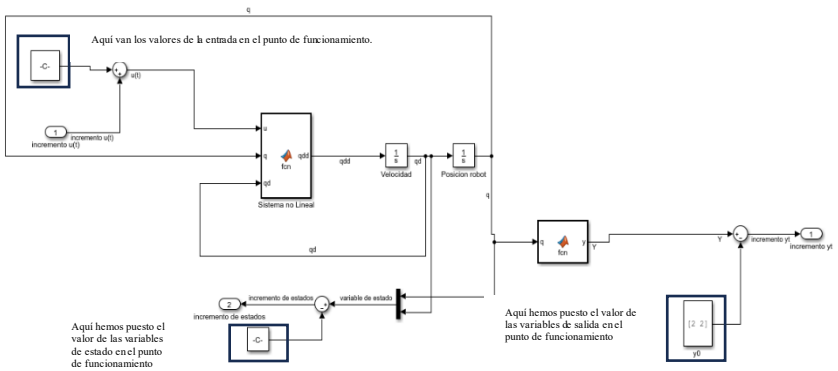


Ilustración 14 Esquema No Lineal.

Y compactando la ilustración anterior obtendríamos lo siguiente:

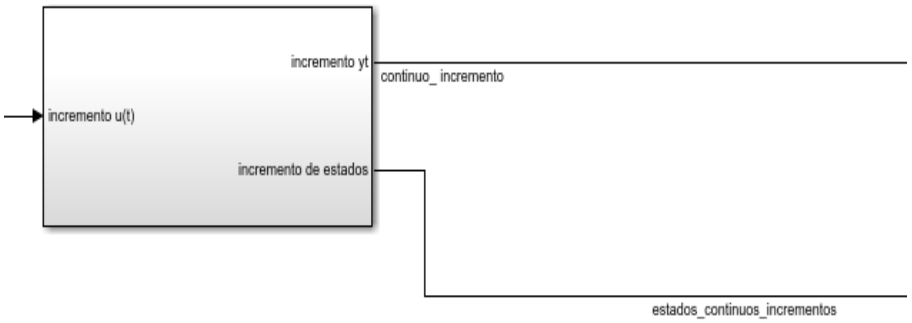


Ilustración 15 Esquema No Lineal compactado.

Como se puede observar, el sistema no lineal que estamos analizando es **continuo en el tiempo**, por lo tanto, es necesario llevar a cabo un proceso de **discretización** para poder analizar la salida ya que los cálculos hechos están en el dominio discreto. Lo haremos siguiendo el mismo proceso que hicimos anteriormente, es decir, con bloqueador de orden cero y muestreador.

En la siguiente ilustración, se compara el controlador aplicado sobre el sistema no-lineal, con el mismo controlador, pero aplicado al sistema linealizado, para verificar la similitud o diferencia entre ambos.

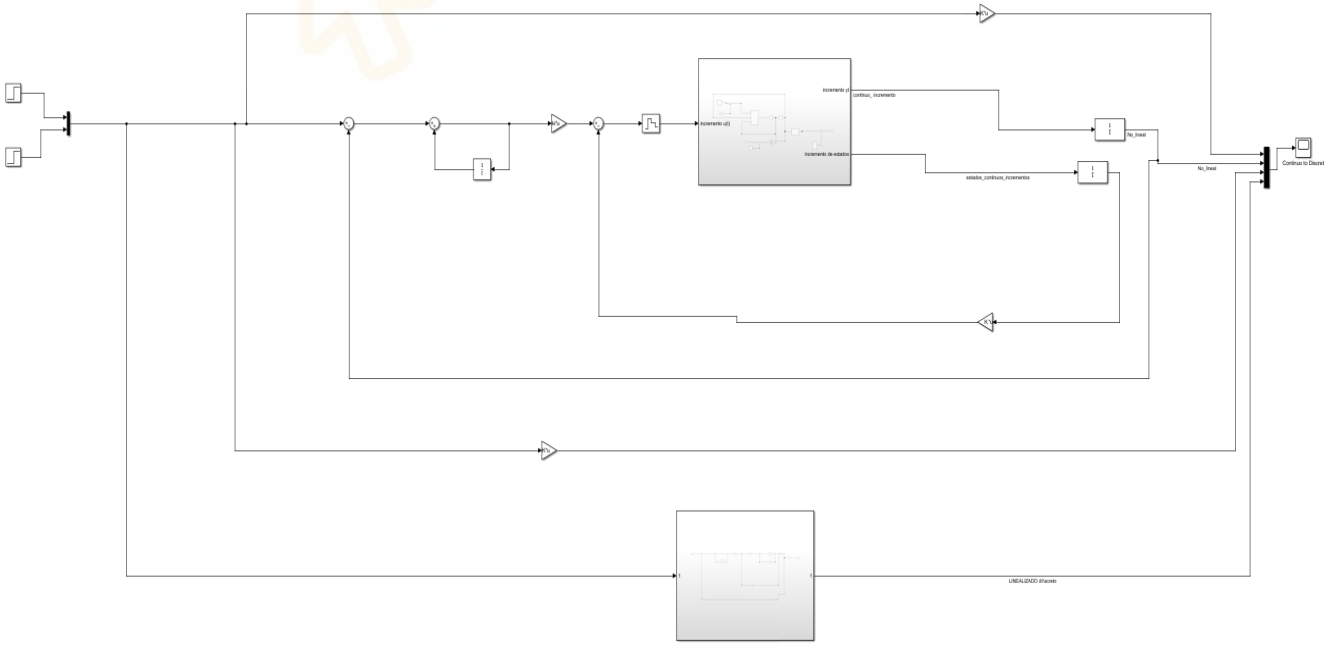


Ilustración 16 Simulink No-Lineal completo.

A continuación, mostraremos la simulación de la salida:

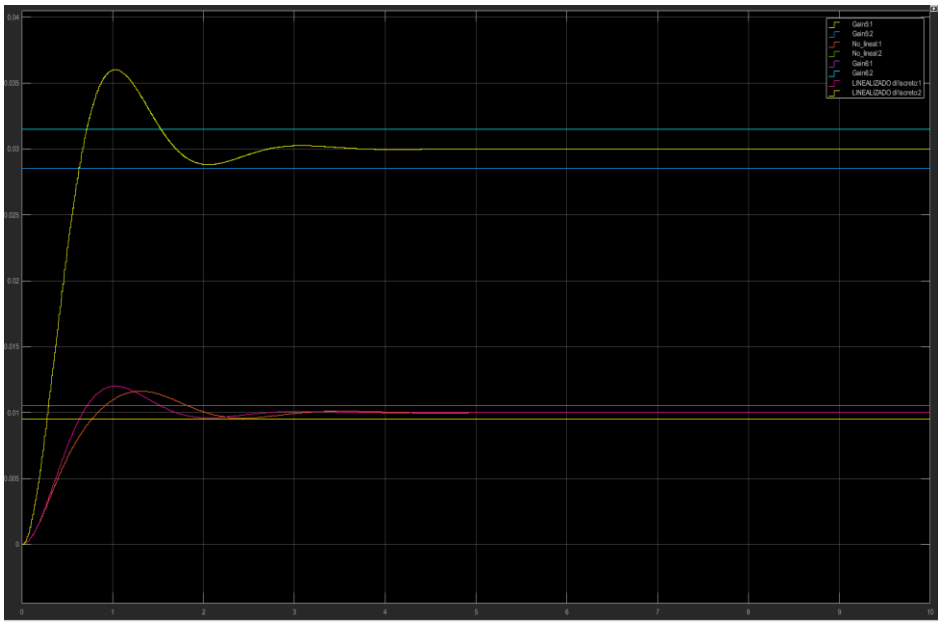


Ilustración 17 Simulación de la salida: Sistema No-Lineal completo.

Como se puede apreciar, el **sistema no lineal** presenta un comportamiento prácticamente equivalente al de su modelo linealizado **en las proximidades del punto de funcionamiento**. Esto indica que, en una región cercana a dicho punto, la aproximación lineal es válida y la simulación obtenida es representativa del comportamiento real del sistema.

Sin embargo, si el sistema se aleja significativamente de este punto de operación, la validez del modelo lineal se pierde, y la simulación deja de ser precisa. En ese caso, el comportamiento no lineal del sistema comenzaría a dominar, por lo que sería necesario recurrir a un modelo más general o relinearizar en torno a un nuevo punto de equilibrio.

3.3.8.4 DISEÑO DEL OBSERVADOR DISCRETO LINEALIZADO

A continuación, procederemos a mostrar el diseño de nuestro observador para poder estimar las variables de estado, suponiendo el caso en el que no pueden medirse directamente, como suele ser habitual.

La gráfica que mostraremos a continuación del observador sigue la plantilla de Ilustración 4, pero de forma aislada.

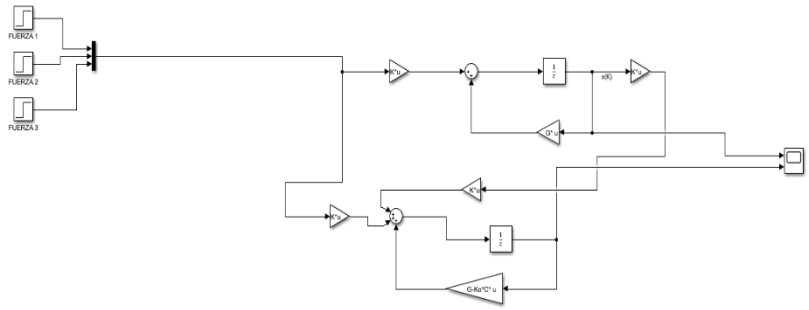


Ilustración 18 Diseño del Observador discreto Linealizado.

Cabe destacar que, en este caso, al igual que en el diseño del controlador, es necesario disponer de un **parámetro de transformación** que nos permita llevar el sistema a su **forma canónica observable**.

Para llevar a cabo el diseño del observador, se seguirán los mismos procedimientos aplicados previamente en el diseño del controlador para la obtención del parámetro k_o.

En primer lugar, se calcula una matriz auxiliar denominada Pabs, compuesta por una selección de filas de la matriz P original. Esta selección es necesaria debido a que esta no es cuadrada, por lo que no puede ser invertida directamente.

A partir de Pabs, se obtiene la matriz de transformación T_o, siguiendo un procedimiento análogo al utilizado anteriormente para calcular T_c en el controlador.

Mediante esta matriz T_o, se transforman las matrices del sistema sin aumentar, ya que solo queremos observar 6 estados lo que tenemos en realidad, y discretizado a su **forma canónica observable**. A partir de dicha

representación, se obtiene el parámetro $\hat{k}_o$, y finalmente se calcula el valor de la **ganancia del observador** k_o en las coordenadas originales.

A continuación, se mostrará el código en MATLAB para obtención de esos datos descritos.

```
%% Paso 1: Verificamos observabilidad del sistema sin aumentar debido a que nosotros
%queremos estimar solo los estados reales.
% Hay que recordar que para este apartado hemos decidido trabajar con un ángulo de
%pi/4 ya que con 0º y 180º el sistema no es observable cuando lo %discretizamos y más
%adelante procederemos a estudiar el porqué de eso.
P = obsv(G,C);
rP = rank(P);
cond(P)
fprintf('Rango de P_aug: %d\n', rP);
if rP < size(G,1)
fprintf('Sistema NO completamente observable.\n');
fprintf('Dividiendo en parte observable y no observable...\n');
else
   fprintf('Sistema completamente observable.\n');
end
%% Como hemos visto antes al tener un sistema MIMO nuestras matrices Q y P no son
cuadradas.
% Utilizaremos el mismo procedimiento para volver la matriz P cuadrada.
combinacionesP = nchoosek([1:12],6);%nchoosek nos da todas las combinaciones de 8
%en 8 apiladas en fila en fila es decir todas las formas de poder combinar los números de
%1 al 12.
condiciones_P_desordenada = [];
 for i = 1 : size(combinacionesP,2)%Size(combinaciones,1) nos dice %cuántas filas tiene
%combinaciones.
i
P_desordenada = P(combinacionesP(i,:),:  );
condiciones_P_desordenada(i) = cond(P_desordenada);
 end
  %%
[valor_minimo,columna_min] =min(condiciones_P_desordenada)
combinacionesP(columna_min,:)
%%
```

```
P_abs=[P([1,3,5,7,2,4],:)] % Pasa lo mismo que en la matriz
%Q ya que tenemos que coger de arriba a bajo si no la
%selección no será correcta.
P_abs_cond=cond(P_abs)
P_abs_inv=inv(P_abs)
%%
% Obtenemos la To
To=[P_abs_inv(:,4)    G*P_abs_inv(:,4)    G*G*P_abs_inv(:,4)
G*G*G*P_abs_inv(:,4) P_abs_inv(:,6) G*P_abs_inv(:,6)]
%Con esta elección, estamos utilizando la primera salida para
%estimar 4 estados, y la segunda para estimar los 2 estados
%restantes.
G_hat = inv(To)*G*To
H_hat = inv(To)*H
C_hat = C*To

G_hat_desired = [zeros(1,5),0;eye(5),zeros(5,1)]
Resta = G_hat - G_hat_desired
C_hat
% Resta = Ko*C
Ko_hat = Resta(:,[4,6])*inv(C_hat(:,[4,6]))
Ko = To*Ko_hat
```

Realizando nuestra simulación con los parámetros obtenidos:

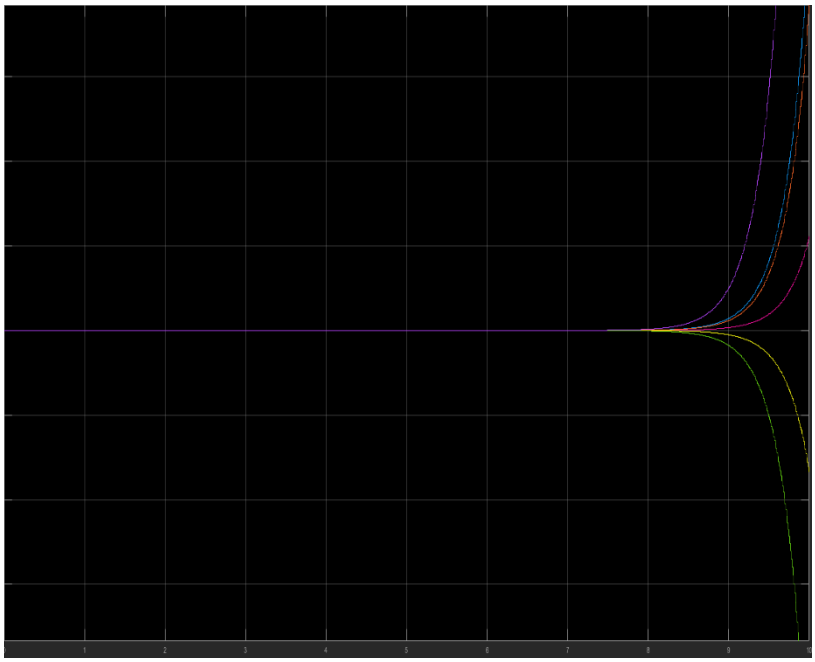


Ilustración 19 Estimación de las variables de estado.

Como se puede observar en la gráfica adjunta vemos que los seis estados son observables tal como habíamos dicho anteriormente.

3.3.8.5 UNIÓN DEL OBSERVADOR Y CONTROLADOR

Llegados a este punto, resulta interesante observar tanto el observador como el controlador integrado en un mismo esquema. A continuación, se presentarán el modelo en Simulink, su correspondiente gráfica de salida, así como la estimación de los estados mediante el observador.

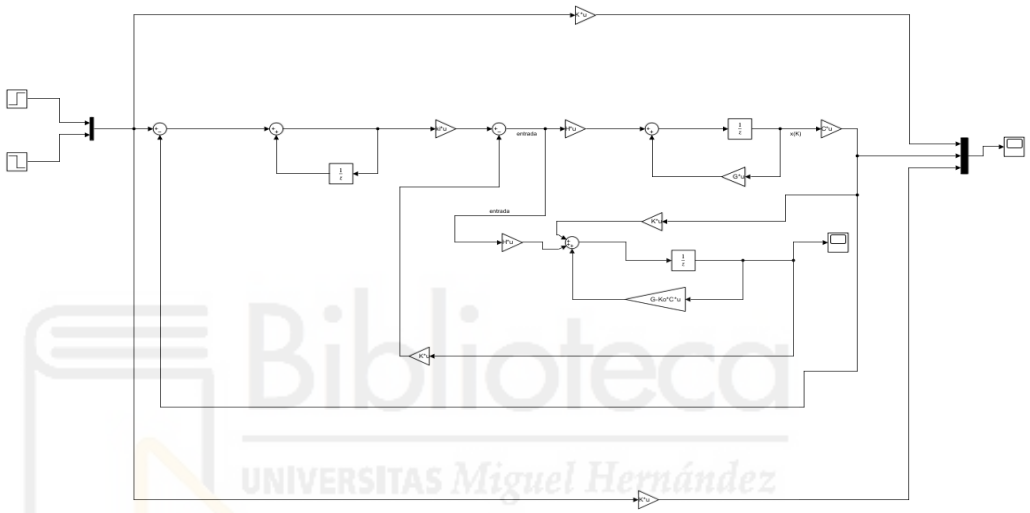


Ilustración 20 Esquema completo: Observador y controlador.

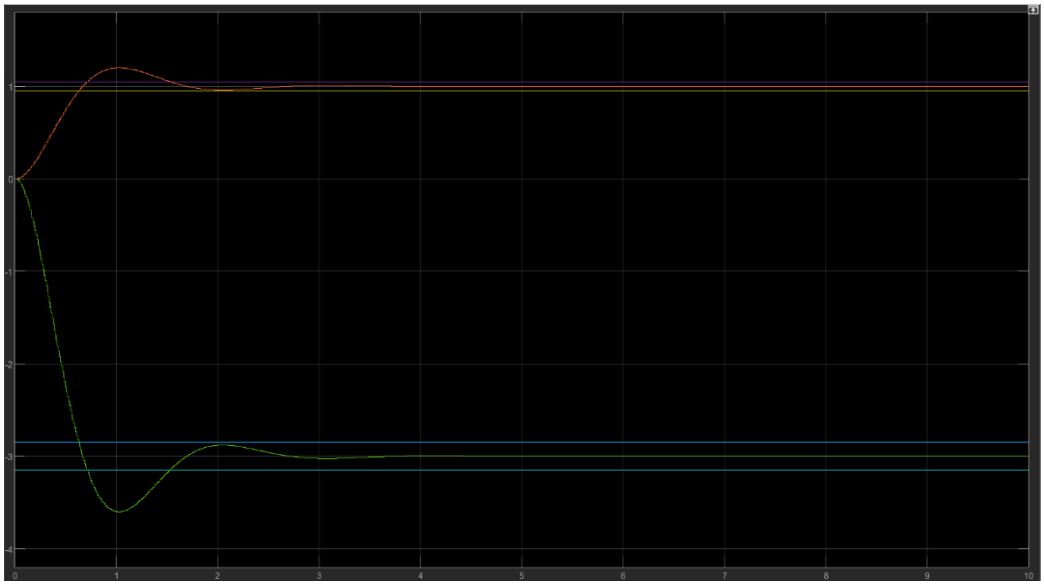


Ilustración 21 Salida de nuestro sistema completo.

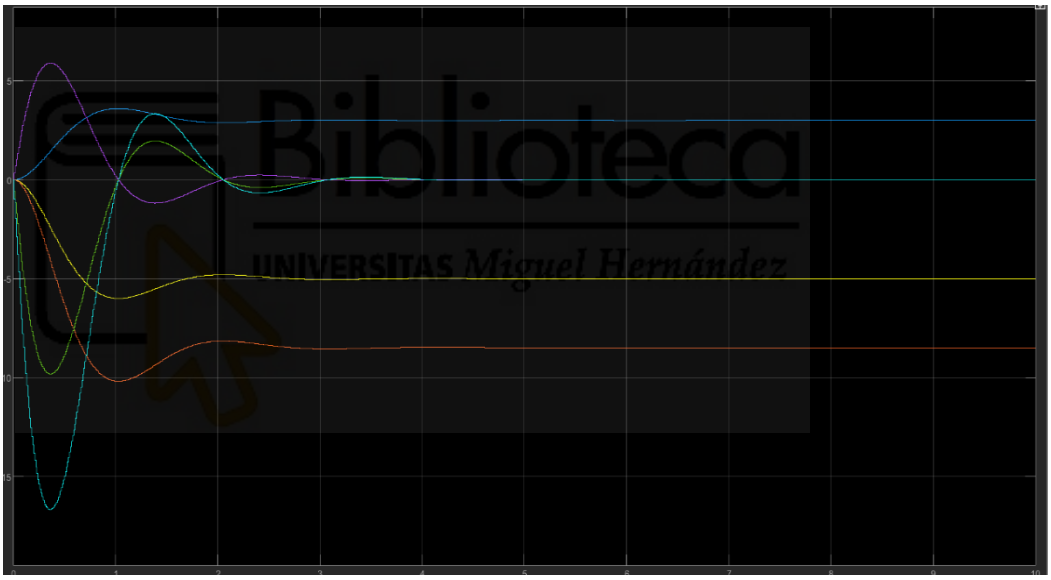


Ilustración 22 Salida del observador.

Como se puede observar en este caso, cuando q3 toma el valor de $\frac{\pi}{4}$, es posible observar los seis estados del sistema.

A continuación, estudiaremos por qué, cuando q3 toma el valor de 0 o π, el sistema se vuelve no observable, es decir cuando nuestro brazo robótico está totalmente horizontal, ya sea hacia la derecha (q3=0) o hacia la izquierda (q3= π). Asimismo, compararemos los resultados de la matriz P cuando q3=$\frac{\pi}{4}$(caso observable) y analizaremos los resultados.

4 ANALISIS DE LA OBSERVABILIDAD CUANDO q3= 0 y $\frac{\pi}{4}$

Como se mencionó anteriormente, nuestro robot se vuelve no observable cuando la articulación 3 toma los valores de 0 o π. Esto se debe a que, al tratarse de un robot redundante, existen situaciones en las que, desde el punto de vista cinemático, no es posible determinar la observabilidad del sistema. Al ser redundante, es posible modificar algunos grados de libertad sin afectar la posición del efector final.

Esto implica que, si el robot se mueve, pero la posición del efector final permanece inalterada, la información puramente cinemática no permite determinar si los actuadores del robot han cambiado su estado. En consecuencia, no es posible estimar con precisión las variables de estado únicamente a partir de la salida del sistema, lo que conlleva a una pérdida de observabilidad.

Dicho lo anterior, procederemos a comparar la matriz de observabilidad en dos casos, con el fin de comprender mejor el fenómeno. El primer caso corresponde a cuando q3 toma el valor de 0, y el segundo, cuando q3 toma el valor de π/4(observable).

4.1 Primer caso: q3=0

En este caso, procederemos a analizar la matriz de controlabilidad (P) en el modo discreto y también podremos hacer un símil para el caso continuo.

- Matriz P tanto continuo como en discreto

$$Pcont = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Tabla 17 Matriz observable continuo.

$$Pdis = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0000 & 1.0000 & 0 & 0 & 0 \\ 1.0000 & 0 & 0 & 0.0100 & 0 & 0 \\ 0 & 1.0000 & 1.0000 & 0 & 0.0100 & 0.0100 \\ 1.0000 & 0 & 0 & 0.0200 & 0 & 0 \\ 0 & 1.0000 & 1.0000 & 0 & 0.0200 & 0.0200 \\ 1.0000 & 0 & 0 & 0.0300 & 0 & 0 \\ 0 & 1.0000 & 1.0000 & 0 & 0.0300 & 0.0300 \\ 1.0000 & 0 & 0 & 0.0400 & 0 & 0 \\ 0 & 1.0000 & 1.0000 & 0 & 0.0400 & 0.0400 \\ 1.0000 & 0 & 0 & 0.0500 & 0 & 0 \\ 0 & 1.0000 & 1.0000 & 0 & 0.0500 & 0.0500 \end{bmatrix}$$

Ilustración 23 Matriz observable discreta

Las dos matrices mostradas anteriormente presentan una condición numérica demasiado grande para ser consideradas observables. En particular, la matriz P en el dominio continuo presenta un número de condición de 6.3050e+17, mientras que en el dominio discreto su número de condición es infinito. Podemos observar a su vez, que en el domino continuo, tenemos muchas filas de ceros, lo que nos imposibilita poder estimas los estados de nuestro sistema.

A continuación, se procederá analizar del mismo modo para el caso siguiente:

4.2 Segundo caso: q3=$\pi/4$

$$P_{\text{cont}} = \begin{bmatrix} 1.0000 & 0 & -0.7071 & 0 & 0 & 0 \\ 0 & 1.0000 & 0.7071 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & -0.7071 \\ 0 & 0 & 0 & 0 & 1.0000 & 0.7071 \\ 0 & 0 & -5.7143 & 0 & 0 & 0 \\ 0 & 0 & 4.2857 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -5.7143 \\ 0 & 0 & 0 & 0 & 0 & 4.2857 \\ 0 & 0 & -69.2676 & 0 & 0 & 0 \\ 0 & 0 & 51.9507 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -69.2676 \\ 0 & 0 & 0 & 0 & 0 & 51.9507 \end{bmatrix}$$

Ilustración 24 Matriz observable continuo.

$$Pdis = \begin{bmatrix} 1.0000 & 0 & -0.7071 & 0 & 0 & 0 \\ 0 & 1.0000 & 0.7071 & 0 & 0 & 0 \\ 1.0000 & 0 & -0.7074 & 0.0100 & 0 & -0.0071 \\ 0 & 1.0000 & 0.7073 & 0 & 0.0100 & 0.0071 \\ 1.0000 & 0 & -0.7083 & 0.0200 & 0 & -0.0141 \\ 0 & 1.0000 & 0.7080 & 0 & 0.0200 & 0.0141 \\ 1.0000 & 0 & -0.7097 & 0.0300 & 0 & -0.0212 \\ 0 & 1.0000 & 0.7090 & 0 & 0.0300 & 0.0212 \\ 1.0000 & 0 & -0.7117 & 0.0400 & 0 & -0.0283 \\ 0 & 1.0000 & 0.7105 & 0 & 0.0400 & 0.0283 \\ 1.0000 & 0 & -0.7143 & 0.0500 & 0 & -0.0355 \\ 0 & 1.0000 & 0.7125 & 0 & 0.0500 & 0.0354 \end{bmatrix}$$

Ilustración 25 Matriz observable discreta.

En este caso, se puede observar claramente que disponemos de información suficiente para determinar los 6 estados del sistema. Esta información adicional nos permite diseñar un **observador** adecuado, ya que contamos con los datos necesarios para estimar correctamente todos los estados del sistema, lo cual es crucial para la implementación de un control adecuado y efectivo.

Por último, para concluir con este apartado, podemos llegar a la conclusión que en nuestro robot, necesariamente necesitamos de la parte dinámica para poder determinar y diseñar un observador, debido a que como tenemos mas grados de libertad que la tarea a realizar pues hay más maneras de mover el robot sin necesidad de haber movido el efector final.

5 CONCLUSIONES

En este TFG se ha abordado el diseño y control de un robot manipulador redundante con tres grados de libertad (dos prismáticos y uno rotativo), partiendo desde su modelado cinemático y dinámico hasta la implementación y validación de un controlador avanzado. Inicialmente, se realizó una linealización del sistema mediante el uso del software Derive, obteniendo un modelo en espacio de estados que permitió calcular las matrices A, B, C y D necesarias para su posterior análisis y control.

Posteriormente, se evaluaron las propiedades de controlabilidad y observabilidad del sistema, tanto continua como discretizada, utilizando MATLAB. Este análisis permitió identificar configuraciones articulares en las que el sistema presenta buena condición numérica para el control y la estimación de estados. Con base en estos resultados, se procedió al diseño de un controlador MIMO basado en realimentación del estado e integradores, capaz de garantizar error nulo en régimen permanente y cumplir con especificaciones dinámicas como tiempo de establecimiento y sobreoscilación.

Para mejorar la robustez del sistema, se implementó una representación en forma canónica controlable, lo que facilitó el diseño de las ganancias del controlador (Kc y Ki) mediante técnicas algebraicas apoyadas en la discretización del sistema. Posteriormente, se integró este controlador en un esquema de Simulink y se probó tanto sobre el sistema linealizado como sobre el sistema no lineal original, verificando que el comportamiento del sistema se mantenía dentro de las especificaciones deseadas, al menos en las proximidades del punto de operación.

Finalmente, se diseñó e implementó un observador para estimar las variables de estado, considerando escenarios en los que no es posible medirlas directamente. También se estudiaron configuraciones articulares que afectaban la observabilidad del sistema, seleccionando aquellas que permitían una correcta estimación de estados.

6 TRABAJOS FUTUROS

A partir del trabajo realizado, se proponen varias líneas de desarrollo que podrían explorarse en el futuro. En primer lugar, sería de gran interés construir un prototipo físico del robot con el objetivo de validar experimentalmente el controlador diseñado y comprobar su funcionamiento en un entorno real.

Asimismo, se podría ampliar el estudio a robots con un mayor número de grados de libertad. En este contexto, sería especialmente relevante investigar cómo aprovechar de manera óptima los grados de libertad adicionales, no solo para el seguimiento de trayectorias, sino también para la evasión de obstáculos, la optimización energética o la mejora del comportamiento dinámico del sistema.

Otra línea prometedora consistiría en extender el proceso de linealización del sistema a lo largo de trayectorias completas, en lugar de hacerlo únicamente en un punto de operación fijo. Esto permitiría trabajar con modelos lineales variantes en el tiempo (LTV).

7 Bibliografía y referencias

- Sempere Ruiz, Jaime.,(2022) CONTROL DE ROBOT PARALELO EN EL ESPACIO DE ESTADO.Trabajo de Fin de Grado, Universidad Miguel Hernández de Elche.
- Peidro Adrian, Payá Luis, Ballesta Mónica,Gil Arturo,Reinoso Óscar.,(2024) Identificación y control de robots paralelos en el espacio de estados con un laboratorio remoto.Revista Iberoamericana de Automática e Informática Industrial, vol 21(2), pp. 180–191. doi: 10.4995/riai.2024.20065.
- Domínguez Sergio, Campoy Pacual,Sebastián José María, Jiménez Agustín(2006)Control en el espacio de estados. 2a edición, Pearson Educación S.A.