Trabajo Fin de Grado
en Ingeniería Electrónica, Robótica y Mecatrónica

Modelado y Control de una Grúa Torre Subactuada

Autor: Alejandro Noci Morales

Tutor: Francisco Rodríguez Rubio

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2026

Trabajo Fin de Grado
en Ingeniería Electrónica, Robótica y Mecatrónica

Modelado y Control de una Grúa Torre Subactuada

Autor:

Alejandro Noci Morales

Tutor:

Francisco Rodríguez Rubio

Catedrático de Universidad

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2026

Trabajo Fin de Grado: Modelado y Control de una Grúa Torre Subactuada

Autor: Alejandro Noci Morales
Tutor: Francisco Rodríguez Rubio

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Un trabajo de extensa trayectoria y que pasa por tantas etapas acaba involucrando a muchas personas a lo largo de su desarrollo; sin embargo, algunas dejan huella ya sea por su influencia directa o por su apoyo moral indirecto. Primero, agradecer a mi tutor Francisco Rodríguez Rubio que siempre me guió cuando lo necesité y estuvo ahí para corregir y aconsejar, sin su ayuda esto no sería posible.
A mi familia por ofrecer apoyo a un tema desconocido, a lo que no podían comprender del todo, pero que a la hora de preguntar siempre eran los primeros.
También, gracias a todos mis amigos del grado, que se interesaron por el proyecto en algún momento y han formado parte de este trayecto de cuatro años que concluye con este trabajo.
Y por último, gracias a María, que ha animado cada paso que avanzaba, cada bajón que sufría y que atendía lo que le explicaba como si el proyecto fuese suyo también.

Gracias a todos.

Alejandro Noci Morales
Sevilla, 2026

Resumen

Este proyecto consiste en el modelado desde cero de una grúa torre de cinco grados de libertad subactuada, siendo lo más cercano posible a la realidad, tanto desde el punto de vista cinemático como dinámico. Posteriormente, se han utilizado dichos modelos para aplicar diferentes estrategias de control. Además, se ha realizado un generador de trayectorias parametrizable en función del tipo de trayectoria que se quiera, el tiempo de movimiento o la precisión en el mismo.

Para el modelo dinámico se ha empleado la formulación de Euler-Lagrange, que posteriormente se ha empleado como si fuese el sistema real en las simulaciones. Dicho modelo se ha simplificado con dos métodos distintos orientados al control. La primera simplificación se basa en una linealización física centrada en el desacoplamiento, orientada a la extracción de las funciones de transferencia de las variables actuadas, mientras que en la segunda se ha aplicado una linealización por Jacobianos, enfocada en la extracción del espacio de estados del sistema.

De la primera técnica de linealización se han sintonizado controladores de tipo PD y PID mientras que de la segunda se han aplicado estrategias de tipo LQR y MPC.

Por último, se han realizado pruebas de los controladores en el seguimiento de trayectorias, de rechazo de perturbaciones y de robustez.

Abstract

This project consists of modelling from scratch a tower crane with five degrees of freedom, which is underactuated, making it the most similar to reality from both a kinematic and dynamic perspective. Afterwards, different control techniques have been used in these models. In addition, a parametric trajectory generator has been developed depending on the type of trajectory wished, the duration of the movement or the precision of it.

The dynamic model has been obtained from the formulation of Euler-Lagrange that later has been used as the real system in the simulations. The model has been simplified with two methods in order to make it feasible to the control strategies. The first simplification is based on making the model lineal through the decoupling of the joints, focused on obtaining the transfer functions of the actuated joints, while in the second simplification the model was made lineal by Jacobians, directed to the extraction of the state space of the system.

From the first simplifications, the controls that have been tuned are PD and PID; meanwhile, from the second one, the control strategies were LQR and MPC types.

Finally, these controls have been tested in trajectory pursuit, disturbance rejection and robustness analysis.

Índice

1 Introducción

El concepto de automatizar se define, según el Diccionario de la Lengua Española, como "*Convertir ciertos movimientos en movimientos automáticos o indeliberados*". Por lo general, se aplica a la industria, ya que estas actuaciones que se realizan de forma automática suelen ser ejecutadas por máquinas sin interacción directa del ser humano y suponen grandes beneficios tanto económicos como productivos.
En el artículo *Historia de la automatización* de la Universidad Ecci [1], se desarrolla toda la historia de esta técnica de mejora de la actividad humana desde la Prehistoria hasta la primera mitad del siglo XXI. Inicialmente, no era automatización como tal, sino un mecanizado del trabajo; simplemente se pasó de máquinas actuadas por humanos y animales a energías renovables, como la fuerza del agua o el viento. Poco a poco, esto fue evolucionando hacia máquinas más complejas, que seguían siendo mecánicas y, por tanto, necesitaban una fuerza motora externa que les diera movimiento.

Gracias al desarrollo de la tecnología energética, cuyo descubrimiento más trascendental fue la electricidad, se permitió que, simplemente disponiendo de dicha energía, se pudieran accionar máquinas sin ningún tipo de acción humana. La evolución de la transferencia de energía, la división en módulos y los sensores, que proporcionaban información del entorno, hizo que la producción y la eficiencia del proceso aumentaran considerablemente.

La inteligencia artificial, la microelectrónica y el desarrollo de nuevas técnicas de fusión sensorial, control, percepción y tratamiento de datos han mejorado las prestaciones y la coordinación de diferentes elementos con los principales sistemas de automatización; los Autómatas Lógicos Programables (PLCs), los microcontroladores y las FPGAs.
Además, muchos de los entornos en los que se opera de forma conjunta con diversos elementos utilizados en la automatización permiten un control por computador remoto. En aquellos en los que no es posible, suelen disponer de sistemas HMI (Human-Machine Interface) o SCADA (Supervisory Control and Data Acquisition) para evitar la interacción directa.

Sin embargo, cabe destacar que no todos los sectores avanzan de la misma forma; no todos tienen la misma capacidad de automatización ni se invierte de la misma manera. Esto depende del trabajo que se realice y las habilidades que involucre. En el documento de la OECD (Organización para la Cooperación y el Desarrollo Económicos) *What skills and abilities can automation technologies replicate and what does it mean for workers?* [2], se muestran en los primeros puntos, a través de estudios realizados, las habilidades que son más propensas a automatizar. Sin embargo, lo verdaderamente relevante son los apartados cuatro y cinco.
En ellos se desglosa, por sectores, el grado de automatización que se podría alcanzar; colocando en lo alto de la tabla la construcción, algunas actividades del sector primario, la producción industrial y el transporte de material. Además, sitúa a España por encima de la media de los países pertenecientes a la organización, donde es más probable que sus actividades productivas puedan ser automatizadas. También relaciona estos sectores con otros aspectos, como la edad, el género o el nivel educativo de los habitantes. La razón de colocar estos sectores en lo alto de la tabla radica en las actividades que se les ha asociado; muchas de ellas se centran en lo técnico, la gestión de datos, la precisión de las máquinas, la atención selectiva, la fuerza estática o la información a través de la visión digital.

El dato más relevante del artículo anterior es que posiciona como el principal sector potencialmente factible para implementar la automatización en la construcción. Esto choca considerablemente con la opinión general, ya que, cuando se habla de automatización, robótica o inteligencia artificial, se suele pensar en innovaciones en defensa, en la industria de la producción o en transporte, con elementos como los UAVs, los gemelos digitales o robots dentro de líneas de producción.
En el artículo *Un futuro que funciona: Automatización, Empleo y Productividad* [3] de la empresa McKinsey & Company, entre otros temas, se expone que la construcción se encuentra en un punto intermedio de automatización en comparación con otras industrias. No es el sector con mayor potencial, lo que contradice en parte el artículo anterior, aunque sí le otorga bastante capacidad de automatización. En este caso, las premisas escogidas se centran en temas o habilidades más cualitativas, como pueden ser la gestión, la pericia, la actividad física no predecible o la predecible.

En esencia, se llega a la conclusión de que tiene potencial para ser un sector automatizado; sin embargo, no es algo que se aprecie en el día a día; es necesario evaluar su situación real. Según el artículo de RICS (Royal Institution of Chartered Surveyors) [4], la principal organización internacional que regula y representa a los profesionales del sector inmobiliario, la construcción y la valoración, reconocida por establecer altos estándares éticos y de calidad a nivel global; el índice de sentimiento de la construcción (CSI) bajó ligeramente, indicando una moderación en el impulso del mercado, aunque sigue en terreno positivo. Más allá, las proyecciones de inflación de costos se mantienen moderadas en comparación con años anteriores. Sin embargo, las restricciones financieras y la escasez de mano de obra cualificada siguen siendo los principales frenos para la actividad. Esto es a nivel global, y en concreto, Europa sigue enfrentando condiciones desafiantes, con una lectura negativa en el índice de actividad actual.

Parece ser que esta es la razón de la falta de automatización en el sector. De hecho, así lo corrobora el artículo *Construction Robotics: From Automation to Collaboration* de la revista Annual Review of Control, Robotics, and Autonomous Systems [5].
Se señala que la industria de la construcción es una de las mayores productoras de CO_2 y es conocida por ser ineficiente, poco estructurada y lenta en adoptar la digitalización en comparación con otras industrias. La adopción de robots se ve dificultada por la gran escala de los proyectos, la complejidad de las obras *in situ* (a diferencia de las fábricas controladas), la imprevisibilidad de los entornos de construcción y la necesidad de que humanos y robots interactúen de forma segura.

La investigación sobre los robots actuales, que son elementos clave en este sector, se centra en la mejora de sus sensores para que puedan adaptarse a medios dinámicos con cambios inesperados, en lugar de seguir instrucciones preprogramadas. Por lo que aún está en desarrollo cómo afrontar esta característica tan común en entornos de construcción; sin embargo, este artículo también menciona que hay tecnologías ya en desarrollo y funcionamiento, así como la impresión 3D de paredes y robots que trepan cualquier superficie.

A la hora de ahondar en las soluciones utilizadas para la automatización de dicho gremio, hay muchos ámbitos en los que se busca hacer uso de la tecnología.

- Impresora de muros [6]: La empresa ICON ha logrado desarrollar dos robots diferentes (VULCAN y TITAN) que imprimen paredes a partir de un material parecido al hormigón. Todos los diseños se crean previamente en el ordenador, por lo que, a la hora de la construcción, el robot los interpreta y realiza el recorrido poco a poco.
  De esta forma, se ahorra mucho tiempo gracias a que la máquina puede trabajar sin interrupciones; se ahorra dinero, ya que se minimiza el desperdicio de material e incluso las paredes se diseñan de forma que disponen de las canalizaciones internas para cables o tuberías, por lo que no es necesario hacer huecos o nuevos caminos para estos servicios.

Figura 1.1 Tipos de impresoras de muros de ICON (Imagen extraída de la referencia[6]).

- Robot albañil [7]: Este robot, a diferencia de los anteriores, construye paredes, pero a base de colocar ladrillos. Se encuentra colocado en un camión, por lo que es fácil de desplazar y para su funcionamiento sólo hay que proporcionarle el mapa del edificio y todos los ladrillos que sean necesarios. Los ladrillos viajarán a lo largo del brazo y la muñeca los colocará de forma automática.
  Las ventajas se repiten de nuevo; supone un ahorro de tiempo, principalmente, además de un mejor acabado, así como un ahorro de material. En este caso, no es tan crítico el ahorro de material, ya que se trata de ladrillos y no de un material como el hormigón, que es mucho más fácil de desperdiciar.

Figura 1.2 Robot albañil de FBR Limited (Imagen extraída de la referencia[7]).

- Robot para acabado de paredes [8]: El acabado de las paredes, tanto la pintura como alguna capa adicional que se quiera dar, siempre es una labor costosa y cuyos fallos son muy visibles. El robot desarrollado por OKIBO dispone de una precisión milimétrica para conseguir el mejor acabado posible y, al igual que otros robots, también optimiza la utilización del material a emplear.
  Se trata de un dispositivo móvil, por lo que puede acabar una habitación en mucho menos tiempo

que una persona y no es excesivamente grande, como sí lo eran los anteriores robots, así que no hay problema en poder usarlo en un espacio reducido.

Figura 1.3 Robot para acabado de paredes de OKIBO (Imagen extraída de la referencia[8]).

- Empresa de construcción modular [9]: Esta empresa se ha desprendido del uso de robots para una labor en concreto dentro de la construcción. Dorce ha conseguido construir módulos de una habitación que disponen de toda la fontanería e instalación eléctrica. A la hora de construir cualquier tipo de edificación, simplemente es necesario colocar los distintos módulos y conectarlos entre sí, lo que permite una construcción muy rápida.
  Estos módulos están diseñados para soportar muchas condiciones climáticas adversas, lo que los hace ideales en cualquier lugar del mundo. En situaciones de emergencia sanitaria, han servido para construir hospitales de forma rápida y también se ha usado este tipo de construcción en instalaciones militares por su rapidez y versatilidad.

Figura 1.4 Construcción modular de Dorce (Imagen extraída de la referencia[9]).

Otro de los elementos más utilizados en la construcción son las grúas, que sirven para la carga y el traslado de material pesado dentro de la zona de trabajo.
De hecho, en el ámbito de la ingeniería civil, la construcción naval y la logística portuaria, el transporte eficiente de cargas pesadas es una necesidad crítica. Por lo que no es un elemento específico de la construcción de edificios.
Más allá, las grúas torre se han consolidado como el estándar industrial para estas tareas debido a su gran capacidad de carga y su amplio espacio de trabajo. Sin embargo, la operación manual de estos sistemas

requiere operarios altamente cualificados y suele estar limitada por la fatiga humana y las condiciones de seguridad. En la automatización industrial exige sistemas robóticos capaces de realizar estas tareas de transporte ("*pick-and-place*") con mayor velocidad, precisión y seguridad.
Aún así, por normativa, aunque no se diga explícitamente, no se tiene aún la seguridad de dejar que este tipo de grúas operen solas sin ningún tipo de control. Siempre se requiere de algún operario que supervise de forma continuada su actividad y que tenga la capacidad de tomar el control en todo momento. Esto se debe a la magnitud de las cargas con las que trabajan y que suelen trabajar en entornos no controlados, es decir, la calle, donde hay muchos agentes externos que pueden afectar a la operabilidad del sistema.
Además, se añade el peligro de que puede haber viandantes que pasan cerca de estos entornos de construcción y, en caso de fallo, pueden sufrir consecuencias fatales.

Tanta precaución se debe al principal desafío dinámico que presentan las grúas torre, su naturaleza subactuada. Un sistema se considera subactuado cuando posee menos actuadores de control que grados de libertad. En el caso de una grúa, los motores controlan la posición de la estructura, pero no actúan directamente sobre la carga suspendida ni su orientación. Esto provoca que la carga se comporte como un péndulo, generando oscilaciones indeseadas inducidas por la inercia del movimiento o perturbaciones externas. Estas oscilaciones no solo reducen la eficiencia operativa, al requerir tiempos de espera para que la carga se estabilice, sino que también representan un riesgo significativo para la seguridad del entorno y la integridad de la carga.

Por tanto, aunque no se utilizan actualmente grúas torre automáticas, la mayoría dispone de sistemas de ayuda, como el *anti-sway*, para minimizar el balanceo o frenos automáticos en caso de un aumento de las velocidades. También se ha conseguido que estas se controlen de forma remota y que no sea necesario acceder a la cabina que hay en altura en la propia grúa por la seguridad del operario.

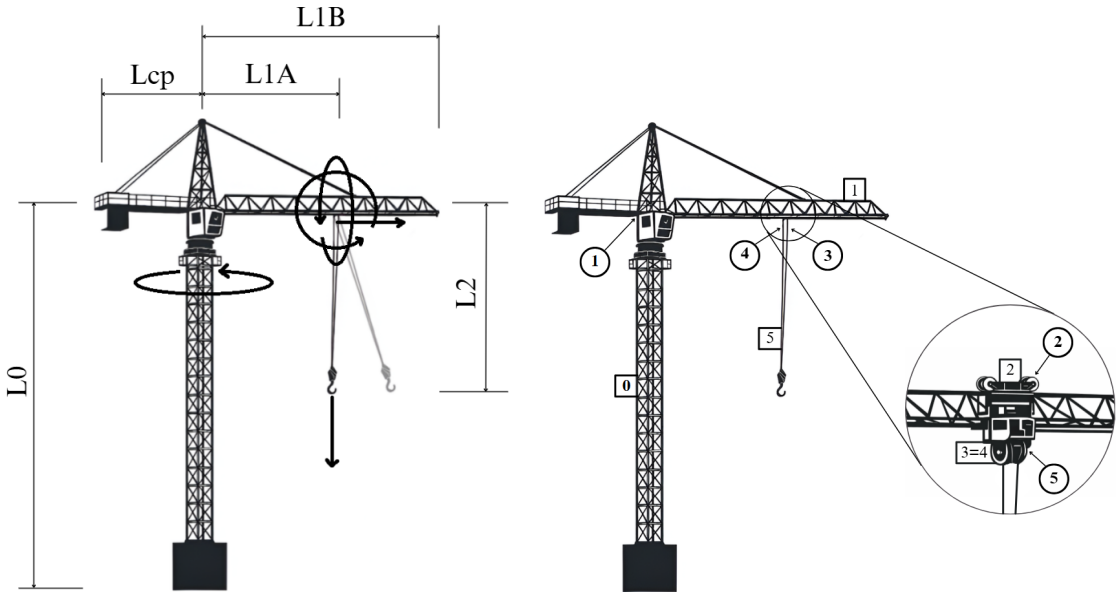
La complejidad dinámica de dicho sistema hace que el control automático del mismo sea todo un reto y haya que ahondar en técnicas de control fuera de lo común o lograr combinarlas para conseguir que funcione sin fallos ni oscilaciones significativas del cable.

En resumen, el objeto de estudio de este proyecto es una grúa torre modelada como un sistema de 5 grados de libertad. La configuración mecánica consta de una primera articulación rotacional que permite el giro de la torre, seguida de una articulación prismática que desplaza el carro a lo largo de la pluma. Desde el carro, un mecanismo de elevación controla la longitud del cable. La complejidad dinámica se introduce a través del cable que sujeta la carga; este se modela como un péndulo esférico. Este subsistema añade dos grados de libertad rotacionales que no poseen actuación directa. Por tanto, el control de la posición de la carga y la supresión de sus oscilaciones debe lograrse indirectamente a través de las variables manipulables.

En definitiva, este trabajo busca aportar soluciones al problema clásico del control de oscilaciones en sistemas subactuados, validando mediante simulación que es posible obtener un desempeño robusto y preciso a través del uso de modelos dinámicos avanzados y estrategias de control adecuadas.
El flujo de trabajo seguirá una ruta similar a la del libro *Fundamentos de Robótica* de Antonio Barrientos [10], donde se explica todo lo necesario para el modelado y control básico de robots manipuladores.

2 Modelo Cinemático del sistema

La cinemática de un brazo robótico es el estudio de su movimiento sin considerar las fuerzas involucradas, con respecto a un sistema de referencia fijo. Dicha relación se extrae a partir del modelo dinámico del sistema. El modelo de este proyecto se basará en los siguientes esquemas, donde se observan sus eslabones, sus articulaciones y el movimiento de cada articulación:



(a) Esquema de grúa torre con longitudes simbólicas y movimientos.

(b) Esquema de grúa torre indicando eslabones y articulaciones[1].

Figura 2.1 Representación detallada del modelo cinemático de la grúa torre..

El modelo cinemático se divide en dos partes; la cinemática directa, la cual expresa cuáles serán los valores de la posición y de la orientación, es decir, {x, y, z, θ, ϕ, ψ}, en función de los valores de las variables articulares. Por otro lado, se encuentra la cinemática inversa, que deriva de la cinemática directa, debido a que esta expresa justamente la relación inversa; el valor de las variables articulares en función de los valores de la posición y la orientación. Existen diferentes métodos para buscar estos modelos:

- Métodos geométricos
- Métodos basados en cambio de base
  - Denavit-Hartenberg
  - Cuaternios

En este proyecto se utilizará el método de Denavit-Hartenberg, ya que, geométricamente, no es posible obtener el modelo a simple vista de forma sencilla. Para dicho método, se necesitará tanto establecer unos ejes

[1] Los recuadros rectangulares pertenecen a los eslabones y los circulares a las articulaciones

de acuerdo con el algoritmo correspondiente como obtener una tabla necesaria para los cálculos a realizar. En el artículo de la Universidad de Sevilla de José Cortés Parejo, *La representación Denavit-Hartenberg*, se explica detalladamente todo el algoritmo en el primer apartado [11] . El primer eje será fijo; se encuentra en el centro del eslabón 0, la base, y el resto de los ejes se disponen de la siguiente manera:

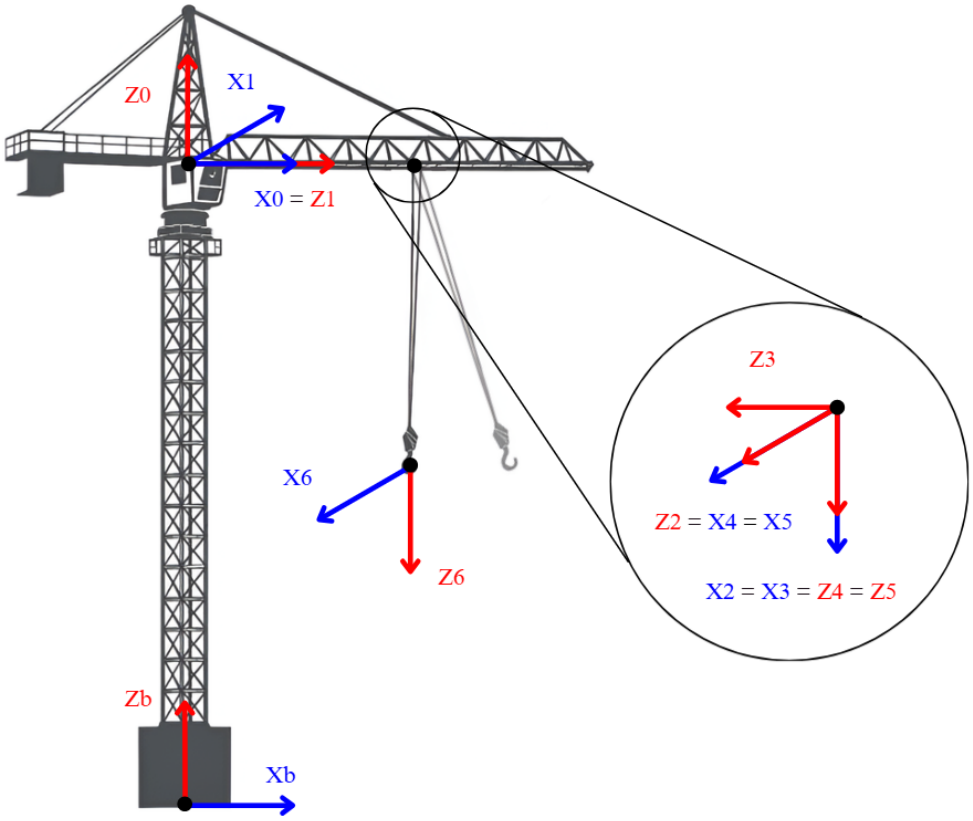


Figura 2.2 Disposición de ejes según Denavit-Hartenberg[2].

De dichos ejes se interpreta que la tabla de Denavit-Hartenberg correspondiente es la siguiente:

Tabla 2.1 Tabla de Denavit-Hartenberg.

Eslabón	θ	a	d	α
0	0	$L0$	0	0
1	$q_1 + \frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
2	$-\frac{\pi}{2}$	$L1A + q_2$	0	$\frac{\pi}{2}$
3	q_3	0	0	$\frac{\pi}{2}$
4	$q_4 + \frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
5	0	q_5	0	0
6	0	$L2$	0	0

[2] Los ejes Y siguen la regla de la mano derecha. Los orígenes de 3, 4 y 5 son coincidentes

2.1 Cinemática directa

Para la cinemática directa, a partir de funciones creadas en Matlab que representan el giro o desplazamiento de un sistema de referencia, se construirá la función *MDH* (Código A.4) que realiza la transformación completa de un eje a otro según la filosofía de Denavit-Hartenberg. Esto consiste en un giro en el eje Z (Código A.1), una traslación (Código A.2) en Z, otra traslación en X y un giro adicional en X (Código A.3). Todo ello en la función irá postmultiplicado, lo que quiere decir que las operaciones se realizarán sobre los ejes móviles, o lo que es lo mismo, sobre el eje transformado debido al movimiento previo. El resultado es una matriz de dimensión 4x4, ya que abarca también la posibilidad de cambio de escala o de perspectiva.

Por tanto, al introducir los valores de la tabla de Denavit-Hartenberg en esta función *MDH*, se obtienen todas las transformaciones de una referencia a otra, y al multiplicar las matrices entre ellas en orden, se obtendrá la transformación completa. Por ejemplo, la matriz que transforma del sistema de referencia 1 al 2 se multiplicará por la matriz que transforma del 2 al 3, obteniendo como resultado el cambio desde el sistema de coordenadas 1 al 3 directamente. De forma homóloga, se realizará la transformación directa del sistema de referencia base al del extensor final; en este caso concreto, el número seis. Las matrices utilizadas en la transformación y la multiplicación completa de las mismas se muestran a continuación:

$$T_{B0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{01} = \begin{bmatrix} \cos(q_1+\frac{\pi}{2}) & 0 & \sin(q_1+\frac{\pi}{2}) & 0 \\ \sin(q_1+\frac{\pi}{2}) & 0 & -\cos(q_1+\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{12} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & L1A+q_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{23} = \begin{bmatrix} \cos(q_3) & 0 & \sin(q_3) & 0 \\ \sin(q_3) & 0 & -\cos(q_3) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{34} = \begin{bmatrix} \cos(q_4+\frac{\pi}{2}) & 0 & \sin(q_4+\frac{\pi}{2}) & 0 \\ \sin(q_4+\frac{\pi}{2}) & 0 & -\cos(q_4+\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{45} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{56} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{B6} = T_{B0} \cdot T_{01} \cdot T_{12} \cdot T_{23} \cdot T_{34} \cdot T_{45} \cdot T_{56} =$$

$$\begin{bmatrix} \cos(q_4)\sin(q_1) - \cos(q_1)\sin(q_3)\sin(q_4) & -\cos(q_1)\cos(q_3) & \sin(q_1)\sin(q_4) + \cos(q_1)\cos(q_4)\sin(q_3) \\ -\cos(q_1)\cos(q_4) - \sin(q_1)\sin(q_3)\sin(q_4) & -\cos(q_3)\sin(q_1) & \cos(q_4)\sin(q_1)\sin(q_3) - \cos(q_1)\sin(q_4) \\ \cos(q_3)\sin(q_4) & -\sin(q_3) & -\cos(q_3)\cos(q_4) \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \cos(q_1)(L1A+q_2)+(L2+q_5)(\sin(q_1)\sin(q_4)+\cos(q_1)\cos(q_4)\sin(q_3)) \\ \sin(q_1)(L1A+q_2)-(L2+q_5)(\cos(q_1)\sin(q_4)-\cos(q_4)\sin(q_1)\sin(q_3)) \\ L0-\cos(q_3)\cos(q_4)(L2+q_5) \\ 1 \end{bmatrix}$$

Dada esta matriz correspondiente a la transformación completa del sistema (T_{B6}), se compara con una matriz (T_{noap}) de la que se extrae directamente la posición de la cinemática directa; p_x, p_y y p_z.

$$T_{noap} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$p_x = \cos(q_1)(L1A+q_2)+(L2+q_5)(\sin(q_1)\sin(q_4)+\cos(q_1)\cos(q_4)\sin(q_3))$$

$$p_y = \sin(q_1)(L1A+q_2)-(L2+q_5)(\cos(q_1)\sin(q_4)-\cos(q_4)\sin(q_1)\sin(q_3))$$

$$p_z = L0-\cos(q_3)\cos(q_4)(L2+q_5)$$

Para la orientación, se optará por la representación en ángulos de Euler RPY (roll, pitch, yaw). El proceso comienza con una rotación ψ sobre el eje X, seguida de una inclinación θ respecto al eje Y original (Código A.5) y finaliza con otro giro ϕ sobre el eje Z fijo inicial. Las funciones de los giros se van a premultiplicar para obtener la matriz de rotación, y esto se debe a que todos los giros se realizan sobre los ejes originales fijos.

$$M_{RPY} = rotz(\phi)\cdot roty(\theta)\cdot rotx(\psi) =$$

$$\begin{bmatrix} \cos(\phi)\cos(\theta) & \cos(\phi)\sin(\psi)\sin(\theta)-\cos(\psi)\sin(\phi) & \sin(\phi)\sin(\psi)+\cos(\phi)\cos(\psi)\sin(\theta) & 0 \\ \cos(\theta)\sin(\phi) & \cos(\phi)\cos(\psi)+\sin(\phi)\sin(\psi)\sin(\theta) & \cos(\psi)\sin(\phi)\sin(\theta)-\cos(\phi)\sin(\psi) & 0 \\ -\sin(\theta) & \cos(\theta)\sin(\psi) & \cos(\psi)\cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Esta matriz, al igual que T_{noap} previamente, se comparará con T_{B6}, obteniendo así los ángulos de Euler despejándolos de la submatriz de rotación 3x3. La resolución se realizará de forma parecida a la publicación de Gregory G. Slabaugh, profesor y director del Instituto de Investigación en Entornos Digitales (DERI), en la Universidad Queen Mary de Londres [12].
Se busca despejar una incógnita con los términos de la matriz en la que aparece y una vez se obtenga la combinación, se sustituyen por los términos equivalentes de la matriz a comparar. La resolución se muestra en el código de la cinemática directa en el apartado de los ángulos de Euler. La solución final se expone a continuación, definiendo previamente algunas componentes para simplificar el resultado:

$$r_{11} = T_{B6}(1,1) = \cos(q_4)\sin(q_1)-\cos(q_1)\sin(q_3)\sin(q_4)$$

$$r_{21} = T_{B6}(2,1) = -\cos(q_1)\cos(q_4)-\sin(q_1)\sin(q_3)\sin(q_4)$$

$$r_{31} = T_{B6}(3,1) = \cos(q_3)\sin(q_4)$$

$$r_{32} = T_{B6}(3,2) = -\sin(q_3)$$

$$r_{33} = T_{B6}(3,3) = -\cos(q_3)\cos(q_4)$$

$$\phi = \arctan(r_{21}, r_{11})$$

$$\theta = \arctan(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2})$$

$$\psi = \arctan(r_{32}, r_{33})$$

Finalmente, para cualquier duda acerca del despeje o la resolución de alguna ecuación, se encuentra todo recogido en el Código A.7 en el anexo de códigos. Se muestran todos los resultados de forma ordenada a continuación:

$$x = \cos(q_1)(L1A + q_2) + (L2 + q_5)(\sin(q_1)\sin(q_4) + \cos(q_1)\cos(q_4)\sin(q_3)) \tag{2.1}$$

$$y = \sin(q_1)(L1A + q_2) - (L2 + q_5)(\cos(q_1)\sin(q_4) - \cos(q_4)\sin(q_1)\sin(q_3)) \tag{2.2}$$

$$z = L0 - \cos(q_3)\cos(q_4)(L2 + q_5) \tag{2.3}$$

$$\phi = \arctan(r_{21}, r_{11}) \tag{2.4}$$

$$\theta = \arctan(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}) \tag{2.5}$$

$$\psi = \arctan(r_{32}, r_{33}) \tag{2.6}$$

2.2 Cinemática inversa

Una vez que se ha obtenido la cinemática directa, se buscará despejar las ecuaciones obtenidas para poder expresar el valor de las variables articulares en función de la posición y orientación del efector final. No se ha comentado antes, pero es más que observable que el sistema está prácticamente desacoplado, es decir, la orientación y la posición no son dependientes entre sí, ya que se definen por variables articulares diferentes. Esto se debe a que los últimos tres ejes del sistema intersectan en el mismo punto, como se puede observar en la Figura 2.2. La posición realmente depende de todas las variables articulares; sin embargo, la orientación únicamente depende de q_1, q_3 y q_4, por lo que a partir de los ángulos de Euler se pueden despejar y posteriormente obtener el resto de las variables con las ecuaciones de posición.

En este caso, las componentes conocidas van a ser x, y, z, θ, ψ y ϕ. Se muestra la resolución de las variables q_3, q_4 y q_1 definiendo las componentes de la matriz M_{RPY} como variables, al igual que se hizo con T_{B6} anteriormente. La demostración, al igual que en la cinemática directa, se encontrará en el anexo de códigos en el Código A.8.

$$r_{12} = M_{RPY}(1,2) = \cos(\phi)\sin(\psi)\sin(\theta) - \cos(\psi)\sin(\phi)$$

$$r_{13} = M_{RPY}(1,3) = \sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta)$$

$$r_{22} = M_{RPY}(2,2) = \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\psi)\sin(\theta)$$

$$r_{23} = M_{RPY}(2,3) = \cos(\psi)\sin(\phi)\sin(\theta) - \cos(\phi)\sin(\psi)$$

$$r_{31} = M_{RPY}(3,1) = -\sin(\theta)$$

$$r_{32} = M_{RPY}(2,1) = \cos(\theta)\sin(\psi)$$

$$r_{33} = M_{RPY}(3,3) = \cos(\psi)\cos(\theta)$$

$$q_3 = \arctan(-r_{32}, \sqrt{r_{31}^2 + r_{33}^2})$$

$$q_4 = \arctan(r_{31}, -r_{33})$$

$$q_1 = \arctan(-r_{22}, -r_{12})$$

Con estas variables, si se observa la expresión (2.3), se puede obtener la variable q_5 directamente:

$$q_5 = \frac{L0 - z}{\cos(q_3)\cos(q_4)} - L2$$

Finalmente, la variable q_2 se definirá en base a las ecuaciones (2.1) y (2.2). Matemáticamente, se desarrolla en el código; sin embargo, de forma lógica, inicialmente se obtiene la posición del *trolley* o carrito en función de la pose del extremo final. Esta se proyecta sobre el plano de giro del eslabón uno, obteniendo así la distancia exacta al carrito. Al restarle el *offset* o desplazamiento inicial (L1A), obtenemos q_2.

$$q_2 = \sin(q_1)\,(y - r_{23}\,(L_2 + q_5)) - \mathrm{L1A} + \cos(q_1)\,(x - r_{13}\,(L_2 + q_5))$$

Finalmente, se muestran todos los resultados de forma ordenada:

$$q_1 = \arctan(-r_{22}, -r_{12}) \tag{2.7}$$

$$q_2 = \sin(q_1)\,(y - r_{23}\,(L_2 + q_5)) - \mathrm{L1A} + \cos(q_1)\,(x - r_{13}\,(L_2 + q_5)) \tag{2.8}$$

$$q_3 = \arctan(-r_{32}, \pm\sqrt{r_{31}^2 + r_{33}^2}) \tag{2.9}$$

$$q_4 = \arctan(r_{31}, -r_{33}) \tag{2.10}$$

$$q_5 = \frac{L0 - z}{\cos(q_3)\cos(q_4)} - L2 \tag{2.11}$$

2.3 Comprobación del modelo cinemático

Todo ello ha sido tratado de forma simbólica; sin embargo, para realizar pruebas reales y comprobar la validez de ambas cinemáticas; se les dará valor a las variables que se suponen conocidas. Para ello, se ha hecho uso de un PFC (Proyecto Final de Carrera), el cual se basa en el diseño y cálculo de una grúa torre [13]. En él, aunque se detallan cada una de las características del sistema, se obtendrán del apartado 5.3.2 (*Dimensiones principales de la grúa torre*) algunas dimensiones características para poder deducir a partir de ellas las variables a utilizar:

- Altura total = 47.14 m
- Altura máxima de trabajo = 40.7 m
- Distancia de seguridad bajo la carga = 3 m
- Alcance desde su centro de rotación = 30 m

Por tanto, $L0$, que se define como la altura hasta la pluma; se le otorga un valor de 45 metros para dejar margen a la carga bajo la pluma (entre la altura total y la altura máxima de trabajo). $L1B$ Será algo mayor que el alcance de la grúa, ya que es la longitud de la pluma y es necesario dejar un margen a ambos lados; por ello, se le asigna una longitud de 35 metros.
Tanto $L2$ como $L1A$ son longitudes que pueden ser cualquier valor intermedio en los rangos de operación correspondientes, así que $L2$ medirá 15 metros y $L1A$ la mitad de $L1B$.
Todas las longitudes se definen según el esquema de la Figura 2.1a.

Una vez que se tienen las dimensiones de la grúa, se definen los rangos de operación de las variables articulares. No se darán valores exactos, sino que se expresarán en función de las longitudes dadas; así, en caso de modificación, se seguirán respetando los márgenes en los extremos:

$$q_1 \in [0, 2\cdot\pi) \quad rad$$

$$q_2 \in [-L1A\cdot\frac{4}{5}, (L1B - L1A)\cdot\frac{4}{5}] \quad m$$

$$q_3 \in [-\frac{\pi}{3}, \frac{\pi}{3}] \quad rad$$

$$q_4 \in [-\frac{\pi}{3}, \frac{\pi}{3}] \quad rad$$

$$q_5 \in [-L2\cdot\frac{4}{5}, (L0 - L2)\cdot\frac{4}{5}] \quad m$$

Para poder operar con el modelo cinemático, en este caso, numéricamente, se van a crear dos archivos de código nuevos; el Código A.9 con la cinemática directa numérica y el Código A.10 con la cinemática inversa numérica, que incluye un indicador en caso de que alguna variable articular esté fuera de su rango.

Todas las especificaciones previas se deben a la necesidad de comprobar las dos cinemáticas; para ello, se ha realizado un programa (Código A.11) que define, a elección, una trayectoria recta o circular, parametrizables ambas modificando algunas variables en el código. La altura se mantendrá constante.
Además de la posición, se determina la orientación necesaria. Se sabe que es un sistema subactuado, por lo que la orientación no es trivial, sino que está determinada. El objetivo de una grúa es que la carga avance de forma estable, sin que oscile el péndulo; por lo que ni el alabeo (*roll*) ni el cabeceo (*pitch*) de los ángulos de Euler deben cambiar respecto al estado inicial (todas las variables articulares a cero). Los valores de la orientación en reposo se extraen al ejecutar el Código A.9 (cinemática directa numérica) con todos sus argumentos a cero (valores de las variables articulares):

$$\phi = -\frac{\pi}{2} \quad rad$$

$$\theta = 0 \quad rad$$

$$\psi = -\pi \quad rad$$

Si el péndulo no oscila, la guiñada (*yaw*) dependerá del ángulo de giro de la grúa torre, ya que el extensor final está alineado con la pluma. Según la disposición RPY, este ángulo es ϕ, así que se calculará el ángulo de la grúa torre con la posición en el plano XY y se le sumará al ángulo inicial para poder así determinar la orientación en cada punto:

$$\phi_{des} = atan2(y,x) - \frac{\pi}{2} \quad rad$$

$$\theta_{des} = 0 \quad rad$$

$$\psi_{des} = -\pi \quad rad$$

Cuando la pose en cada uno de los puntos de la trayectoria está definida, se pasa como argumento, punto por punto, a la cinemática inversa para que obtenga los valores de las variables articulares en cada punto. Posteriormente, este conjunto se utiliza como argumento, también punto por punto, en la cinemática directa que, en caso de que no haya fallos, deberá devolver exactamente la misma pose inicial.
En caso de que alguna variable articular esté fuera de rango, se pondrán todas a cero y se mostrará un mensaje en el terminal de Matlab indicando esto mismo.
Las gráficas generadas mostrarán la trayectoria deseada junto con la trayectoria real seguida y la evolución de las variables articulares a lo largo de la trayectoria seguida.

A continuación, se muestran las gráficas extraídas en cada caso:

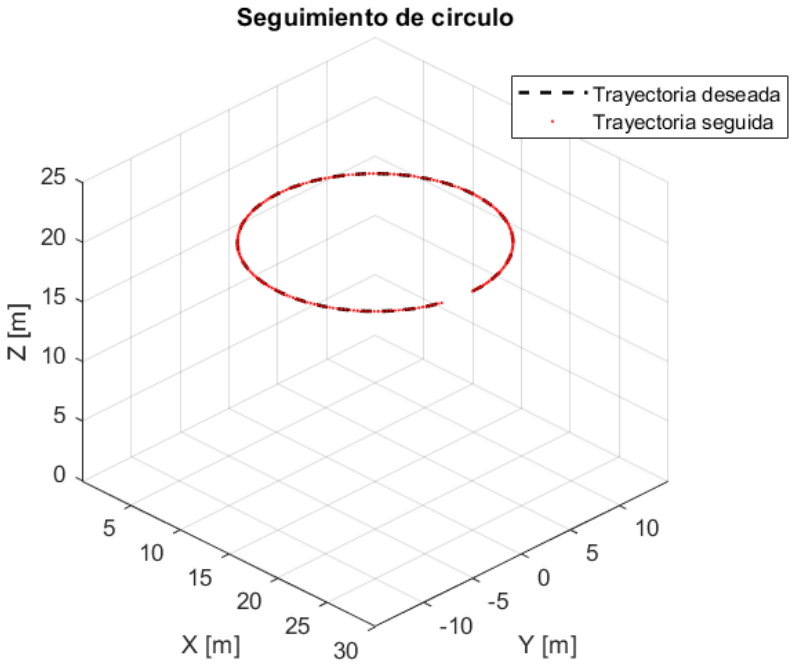


Figura 2.3 Trayectoria circular y seguimiento.

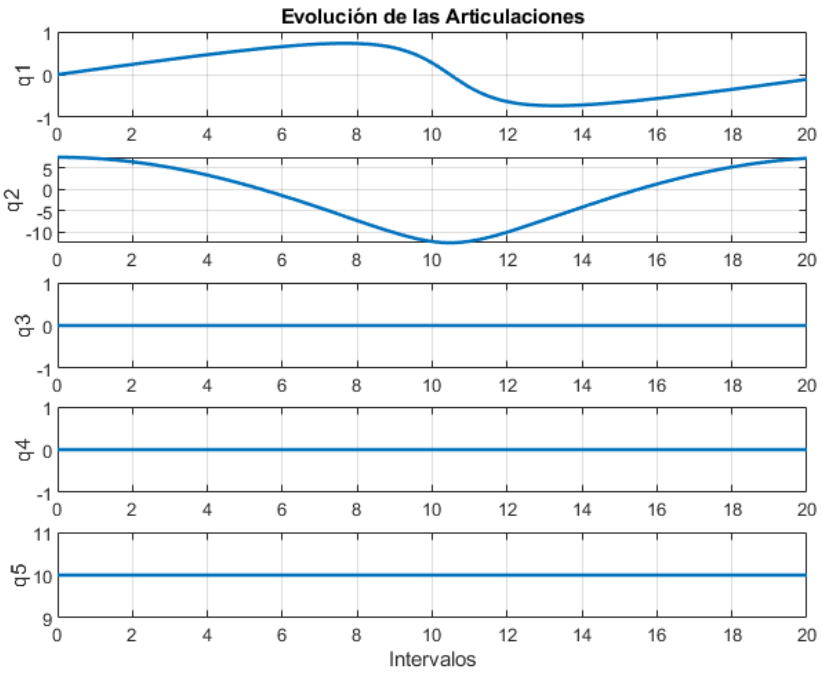


Figura 2.4 Evolución de las variables articulares en la trayectoria circular.

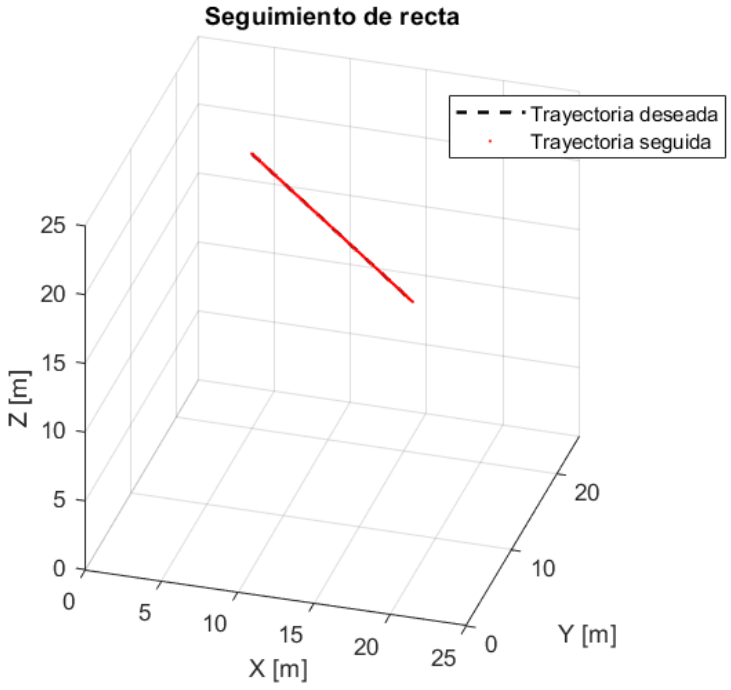


Figura 2.5 Trayectoria recta y seguimiento.

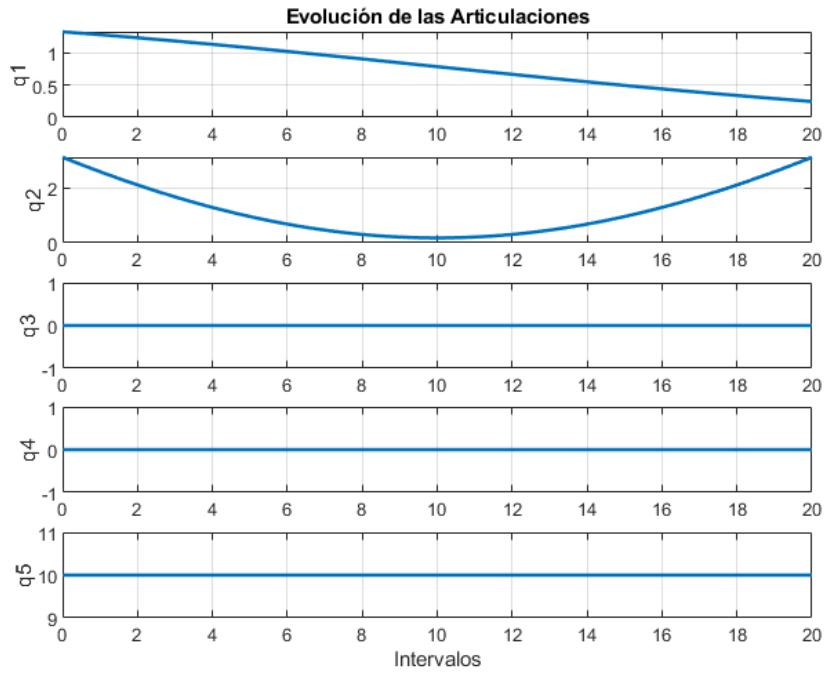


Figura 2.6 Evolución de las variables articulares en la trayectoria recta.

Se muestra también una trayectoria circular en la cual el radio es demasiado grande, lo que produce que la variable q_2 se salga de su rango de operación:

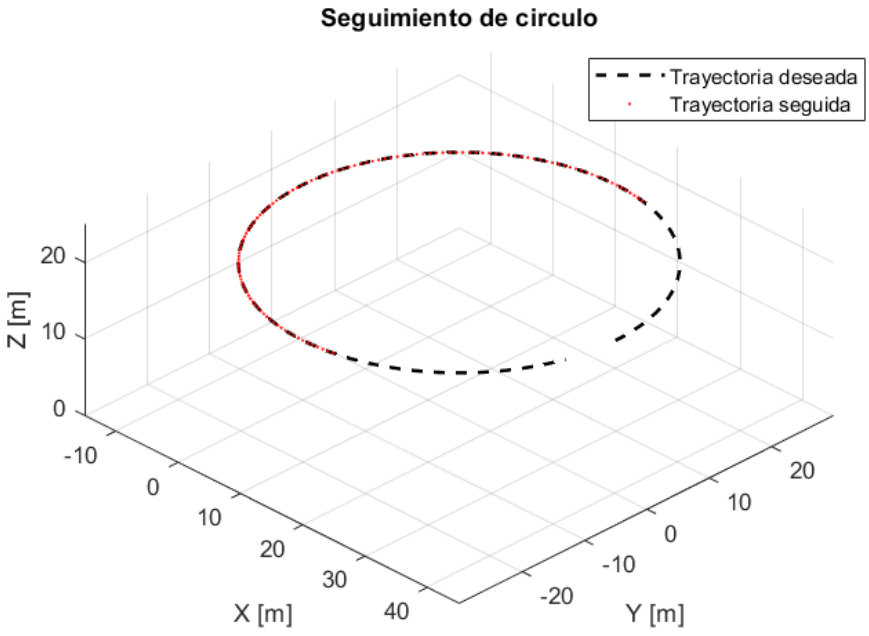


Figura 2.7 Trayectoria circular no realizable y seguimiento.

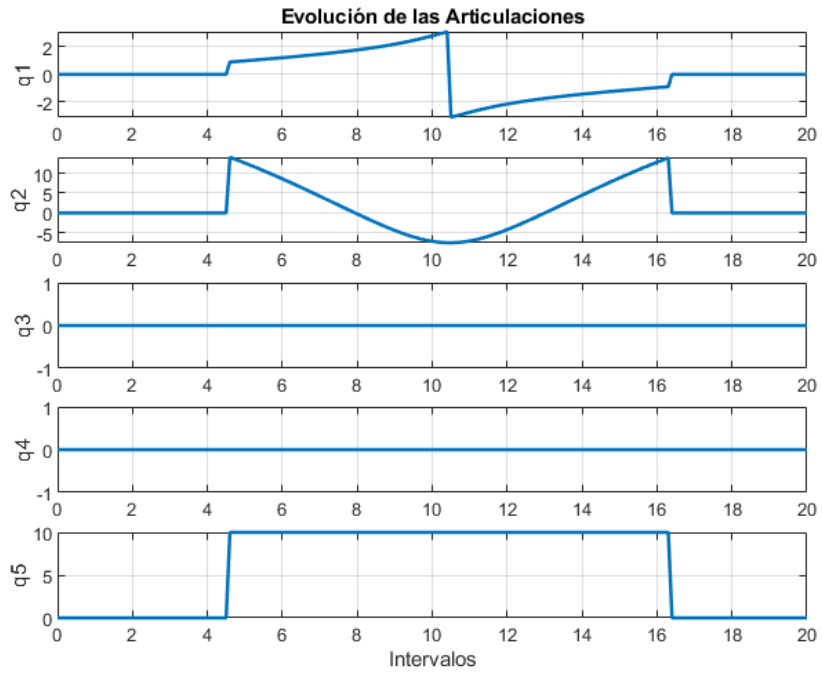


Figura 2.8 Evolución de las variables articulares en la trayectoria circular no realizable.

De las gráficas se deduce que, en principio, no hay error en el modelo cinemático y que los límites impuestos a las variables articulares son lógicos y se cumplen.

2.4 Modelo diferencial

Partiendo de la cinemática de la grúa, se van a extraer las dos matrices del modelo, la matriz Jacobiana y la Jacobiana Inversa.
Con la matriz Jacobiana se expresará el valor de la velocidad del extremo, es decir, la velocidad lineal en los ejes X, Y y Z; y la velocidad angular en los ángulos ϕ, θ y ψ, en función de las posiciones y velocidades de las articulaciones del sistema.
Por tanto, la matriz Jacobiana inversa expresa la relación opuesta; las velocidades articulares en función de la posición y las velocidades lineales y angulares del extremo.
Para la extracción de dichos modelos, se seguirá principalmente el apartado 4.3. del libro *Fundamentos de robótica* de Antonio Barrientos [10].

Además, el cálculo de ambas matrices se realiza de forma simbólica en el Código A.12.

2.4.1 Jacobiano directo

Para la extracción de la Jacobiana, se parte de las ecuaciones de la cinemática directa, ya que se optará por extraer la Jacobiana analítica. La matriz se construye de la siguiente manera:

$$x = f_x(q_1,\dots,q_5) \qquad y = f_y(q_1,\dots,q_5) \qquad z = f_z(q_1,\dots,q_5)$$
$$\phi = f_\phi(q_1,\dots,q_5) \qquad \theta = f_\theta(q_1,\dots,q_5) \qquad \psi = f_\psi(q_1,\dots,q_5)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{J}_a \cdot \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \dot{q}_5 \end{bmatrix} \quad \text{con} \quad \mathbf{J}_a = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \cdots & \frac{\partial f_x}{\partial q_5} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_\psi}{\partial q_1} & \cdots & \frac{\partial f_\psi}{\partial q_5} \end{bmatrix}$$

2.4.2 Jacobiano inverso

Para la obtención de esta matriz, se hace de forma análoga a la Jacobiana; es decir, se parte de las ecuaciones de la cinemática inversa:

$$\begin{aligned} q_1 &= f_1(x,y,z,\phi,\theta,\psi) \\ &\vdots \\ q_5 &= f_5(x,y,z,\phi,\theta,\psi) \end{aligned}$$

$$\begin{bmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_5 \end{bmatrix} = \mathbf{J}_a^{-1} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad \text{con} \quad \mathbf{J}_a^{-1} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \cdots & \frac{\partial f_1}{\partial \psi} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_5}{\partial x} & \cdots & \frac{\partial f_5}{\partial \psi} \end{bmatrix}$$

Hay otras formas de extraer esta matriz pero; principalmente, los problemas recaen en que estas matrices no son cuadradas, por tanto el Jacobiano no se puede invertir y en que en Matlab la carga simbólica del Jacobiano es demasiado grande y no es sencillo operar con ella.

2.4.3 Puntos singulares

Se definen como configuraciones singulares aquellos estados del robot en los que el determinante de su matriz Jacobiana se vuelve cero ($\det(J) = 0$). Matemáticamente, esta condición implica que la matriz no es invertible, por lo que la Jacobiana inversa no existe en dichos puntos, aunque en este caso no aplica.

Desde un enfoque práctico, las singularidades presentan dos problemas críticos:

- Velocidades infinitas: Al aproximarse a una singularidad, un pequeño movimiento en el espacio cartesiano requeriría velocidades articulares extremadamente altas (teóricamente infinitas). En la realidad, esto satura los actuadores del robot, impidiendo un movimiento fluido o controlado.
- Pérdida de movilidad: En estos puntos, el robot pierde uno o más grados de libertad. Esto significa que existe al menos una dirección en el espacio cartesiano en la cual el extremo del robot es incapaz de desplazarse, independientemente de los movimientos que realicen sus articulaciones.

En ocasiones, no hay un problema real; simplemente, debido a la multiplicidad de posibilidades, se dan configuraciones en las que no se presenta ninguna de las posibilidades anteriores.

Para este sistema, el Jacobiano, como se ha comentado previamente, no es cuadrado; por tanto, no se puede hallar el determinante directamente. Por tanto, para hallar las configuraciones singulares, el determinante se calculará a partir de la matriz Jacobiana multiplicada por su matriz traspuesta.

$$\det(J_a^T \cdot J_a) = 0$$

En el Código A.13 se muestra el cálculo realizado y a continuación las soluciones que se obtienen del mismo al igualar a cero el determinante:

$$q_1 = 0$$
$$q_2 = 0$$
$$q_3 = \frac{\pi}{2}$$
$$q_4 = 0$$
$$q_5 = 0$$

Aún así, se sabe que en realidad hay más, pero no son puntos en sí mismos, sino superficies que conforman las variables articulares. Se obtienen; por tanto, los factores del determinante y se evalúan por separado por si se pudiese definir alguna solución más. El determinante consta de seis factores; sin embargo, uno es simplemente un cambio de signo (-1), y cuatro de ellos están repetidos dos a dos; por lo que realmente son tres. Se exponen los factores repetidos, ya que el factor restante es demasiado largo:

$$\cos(q_3)$$

$$\frac{1}{\cos(q_3)^2 \cos(q_4)^2 - \cos(q_3)^2 + 1}$$

Del primer factor, se obtiene la solución que nos proporcionó Matlab. El segundo no tiene solución al igualarlo a cero, ya que el denominador no puede hacerse infinito para que sea cero. Y por último, la expresión del último factor, Matlab indica que tiene tres soluciones, parametrizando todas las variables articulares excepto q_2, que se expresa en función de los cuatro parámetros. Aún así, impone varias condiciones a cada ecuación para que sean válidas, por lo que habría que comprobar que se cumplen estas condiciones para calcular puntos concretos de la superficie.
Aún así se ha comprobado que efectivamente hay infinitos puntos singulares repartidos a lo largo de varias superficies de cuatro parámetros, aunque seguramente con las restricciones aplicadas a las articulaciones no se alcancen nunca.

Para las simulaciones a realizar, se generarán trayectorias que se sabe que son seguras.

2.5 Creación de modelo con Robotics System Toolbox

Una vez que se tiene toda la cinemática, para tener una retroalimentación visual de ella y de la dinámica y control del sistema, que se abordará más adelante, se ha optado por desarrollar la cinemática de la grúa con el paquete mencionado para ver cómo se comporta y comprobar en tiempo real la posición que adquiere en cada uno de los puntos de las trayectorias cinemáticas. Esto es gracias a que simplemente se le asigna en cada instante el valor de la articulación que calcula el modelo y se observa el movimiento poco a poco.
Además, se utilizará el mismo modelo para observar la evolución del robot cuando se le aplique una técnica de control. El método es el mismo que en la cinemática, por lo que el robot en sí no modela la dinámica; sino que las ecuaciones dinámicas calculan las aceleraciones, se integran dos veces y se obtienen las posiciones articulares que se le pasarán al robot para ver la posición.

Para modelar, ha sido necesario crear la instancia del robot en sí mismo con *rigidBodyTree* y, posteriormente, ir poco a poco declarando cada eslabón con su articulación asociada e incluyendo el conjunto al robot. Para el cálculo de los sistemas de referencia, se ha optado por utilizar directamente las matrices de transformación y asociarlas a la articulación De esta forma, no hay lugar a confusión a la hora de interpretar el movimiento de los ejes de una articulación a otra.
Todo el código completo se encuentra en el Código A.14. Para la visualización, cuando se prueba alguna estrategia de control, se ha acelerado el movimiento para no tener que esperar tanto a la hora de la comprobación en escala real del movimiento. También se incorpora un punto que muestra el punto de referencia que sigue el sistema y un haz que dibuja el efector final para comparar la trayectoria a seguir con el movimiento realizado realmente.

Sólo se ha utilizado en este proyecto para comprobar la cinemática y realizar inspecciones visuales rápidas de los controladores implementados. De este último uso no se encontrarán gráficas, ya que con las gráficas de los errores en cada articulación y en las coordenadas cartesianas, además de las señales de control, queda suficientemente definido.
Sin embargo, se muestra un ejemplo de este uso para certificar que funciona y se ha utilizado:

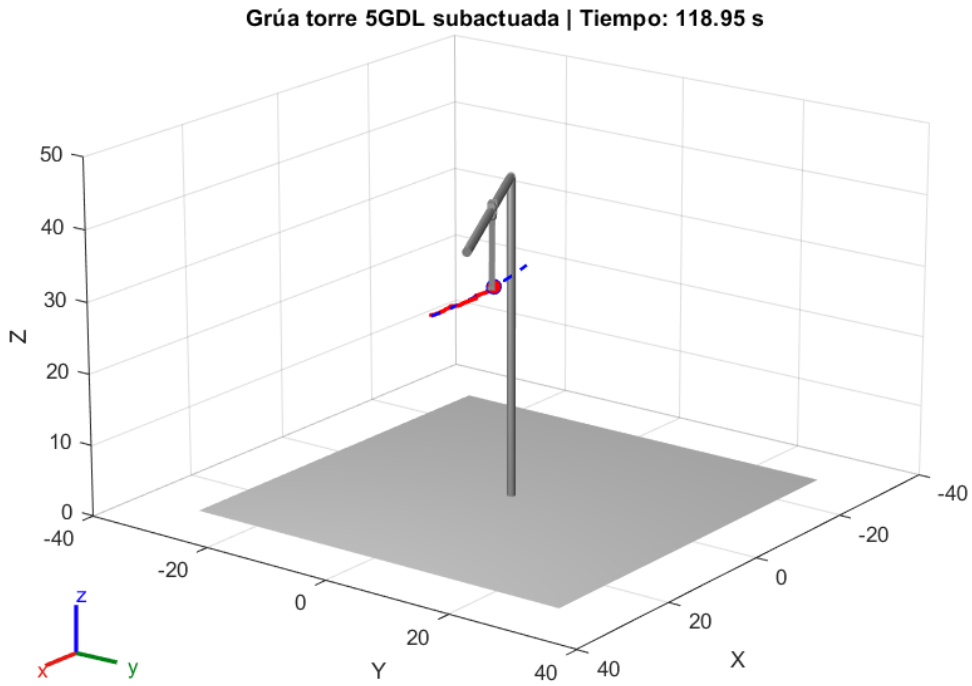


Figura 2.9 Funcionamiento del modelo con Robotics System Toolbox en una prueba de control.

3 Generador de trayectorias

Previo al cálculo de parámetros de controladores y su implementación, lo más importante es poder generar una trayectoria suave que facilite el seguimiento de las referencias al controlador. Se estudiarán los distintos tipos de trayectoria e interpoladores entre puntos para poder gestionar el seguimiento de las variables articulares.

Para el desarrollo de este apartado, se ha hecho uso de nuevo del libro *Fundamentos de robótica* de Antonio Barrientos [10], en este caso del apartado 6.

Una trayectoria es el movimiento a realizar entre dos puntos dados, uno inicial y otro final. Realmente las posibilidades de movimiento espacial son prácticamente infinitas incluso excluyendo las configuraciones que el robot no es capaz de alcanzar.
Aún así, en función de la utilidad para la que se vaya a usar el sistema y de la implementación de la generación de la trayectoria, en robots industriales se tienen principalmente dos tipos.

- Trayectorias punto a punto: Son aquellas en las que cada articulación evoluciona de su posición inicial a la final, sin tener en cuenta las demás.
- Trayectorias continuas: Para garantizar que el efector final de un robot siga una trayectoria definida en el espacio cartesiano (como líneas rectas o arcos circulares), es imprescindible calcular y supervisar continuamente las trayectorias de cada articulación. Aunque estas últimas suelen presentar una evolución temporal compleja y un comportamiento individual que parece errático o desbalanceado, su acción conjunta permite que el extremo del robot ejecute con precisión el movimiento deseado.

Dentro de las trayectorias punto a punto se encuentran diferentes formas de abordar su implementación:

- Movimiento eje a eje: Las articulaciones se irán moviendo una a una de forma sucesiva; es decir, cuando una de ellas complete su recorrido, comenzará la siguiente, pero nunca se moverán de forma simultánea. Este tipo de movimiento conlleva más tiempo y su única ventaja es la reducción del consumo instantáneo de potencia y el posible uso de una sola etapa de potencia.
- Movimiento simultáneo de ejes: Todas las articulaciones comenzarán su movimiento a la vez; sin embargo, debido a que sus distancias a recorrer y sus velocidades son distintas, cada una puede terminar en cualquier momento. El movimiento del robot general acabará cuando termine la articulación más lenta, ya que en ese momento se alcanzará la configuración final. Es posible que otras articulaciones aumenten demasiado su velocidad o aceleración, haciendo un gasto inútil de energía en caso de que no sean las más lentas.
- Trayectorias coordinadas o isócronas: Para optimizar el uso de los actuadores y evitar esfuerzos innecesarios en términos de velocidad y aceleración, se implementa una sincronización basada en el eje más lento. El sistema identifica qué articulación requiere más tiempo para completar su desplazamiento y ajusta la velocidad de las demás para que todas finalicen simultáneamente. En este tipo de trayectorias, el tiempo total es el mínimo posible para la configuración dada, aunque la trayectoria espacial del extremo del robot resulta impredecible al no estar restringida a una geometría específica.

En el caso de una grúa torre, en principio no debería haber problemas al seguir una trayectoria punto a punto, ya que normalmente el espacio de la tarea está despejado; sin embargo, en ocasiones es necesario

seguir una trayectoria relativamente estricta en coordenadas cartesianas.
Debido a esto, se implementará la posibilidad de la trayectoria punto a punto coordinada y también algo parecido a la trayectoria continua, pero sin seguir estrictamente el recorrido.

Una vez que se sabe el tipo de trayectoria que se va a seguir, conviene calcular una serie de puntos intermedios e interpolar posteriormente para poder proporcionar referencias poco a poco, evitando así errores muy grandes que generen señales de control muy bruscas.
Se seguirá en este caso un procedimiento ligeramente diferente al del libro de referencia.

Inicialmente, se parte de una serie de datos:

- Punto inicial y final (coordenadas cartesianas)
- Número de puntos intermedios por los que pasar
- Instante de inicio del movimiento
- Duración del movimiento

El flujo de trabajo a seguir es el siguiente:

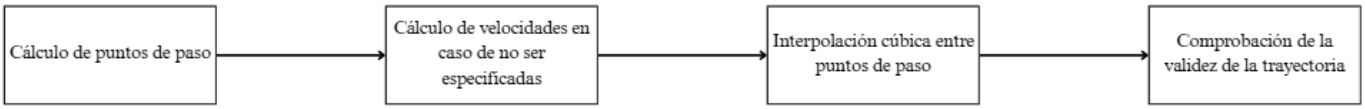


Figura 3.1 Pasos a seguir en la generación de trayectorias.

Para el cálculo de puntos de paso, se encuentran dos posibilidades. Para empezar, a través de la cinemática inversa se extraerá la configuración de las variables articulares en el punto inicial y final, además de comprobar la viabilidad de dichos puntos. Posteriormente, si se quiere una trayectoria punto a punto, a partir de las dos configuraciones articulares obtenidas, se realizará un interpolado lineal para obtener tantos puntos de paso como se hayan especificado.
En caso de querer definir una trayectoria aproximada en coordenadas cartesianas, o bien se interpola generando una línea recta entre el punto inicial y final o bien se genera una parábola entre ambos puntos para que sea una trayectoria curva siendo la altura una línea recta. De esta forma se dispone de tres opciones para generar puntos de paso.

Debido a la complejidad mecánica de sistemas como una grúa, existen limitaciones que impiden alcanzar velocidades excesivamente elevadas. Por este motivo, para realizar el cálculo del recorrido entre puntos intermedios se ha optado por el uso de interpoladores cúbicos, definidos por las posiciones y velocidades en los extremos de cada tramo.

Este enfoque implica la necesidad de definir las velocidades de paso en cada punto intermedio. No obstante, en condiciones reales, habitualmente solo se dispone de la información de los extremos, donde tanto la velocidad inicial como la final son iguales a cero. La ventaja del spline cúbico reside en sus cuatro parámetros internos, los cuales permiten imponer cuatro condiciones de contorno entre cada par de puntos del recorrido.

Existen otras alternativas en el ámbito de la interpolación:

- El interpolador lineal, cuya estructura simple no ofrece la posibilidad de especificar velocidades de paso.
- El interpolador quíntico permite un control más exhaustivo al definir aceleraciones, además de la posición y la velocidad.
- El interpolador trapezoidal, que constituye una solución intermedia al fragmentar el recorrido en tres etapas: un segmento central de velocidad constante y dos extremos donde se emplea un polinomio de segundo grado para que la velocidad varíe linealmente.

Para la ejecución de este proyecto se ha seleccionado el polinomio de tercer orden. A continuación, se presenta la formulación matemática empleada para realizar la interpolación entre dos puntos adyacentes (q^i, q^{i+1}):

$$q(t) = a + b(t-t^i) + c(t-t^i)^2 + d(t-t^i)^3, \quad t^i < t < t^{i+1}$$

Donde los coeficientes son:

$$
\begin{aligned}
a &= q^i \\
b &= \dot{q}^i \\
c &= \frac{3}{T^2}(q^{i+1} - q^i) - \frac{1}{T}(\dot{q}^{i+1} + 2\dot{q}^i) \\
d &= -\frac{2}{T^3}(q^{i+1} - q^i) + \frac{1}{T^2}(\dot{q}^{i+1} + \dot{q}^i) \\
T &= t^{i+1} - t^i
\end{aligned}
$$

Para la selección de velocidades de paso, hay varios métodos; poner todas a cero, lo que en ocasiones puede servir, pero no es lógico, ya que ralentiza el proceso. También se puede considerar que sean cero si cambia el sentido del movimiento y, en caso contrario, hacer la media de las supuestas velocidades constantes que deberían tener en el instante previo y posterior para ir de un punto a otro en el tiempo dado:

$$
\dot{q}^i = \begin{cases} 0 & \text{si signo } (q^i - q^{i-1}) \neq \text{signo } (q^{i+1} - q^i) \\ \frac{1}{2}\left[\frac{q^{i+1}-q^i}{t^{i+1}-t^i} + \frac{q^i - q^{i-1}}{t^i - t^{i-1}}\right] & \text{si } \begin{cases} \text{signo } (q^i - q^{i-1}) = \text{signo } (q^{i+1} - q^i) \\ \text{o} \quad q^{i-1} = q^i \\ \text{o} \quad q^i = q^{i+1} \end{cases} \end{cases}
$$

Estos métodos proporcionan una continuidad lógica en velocidad, pero la aceleración lo más probable es que sea discontinua. Esto puede generar problemas en los actuadores, por lo que, para poder generar una trayectoria a partir de un spline cúbico que sea continua en aceleración, es necesario resolver el siguiente sistema de ecuaciones:

$$
\begin{bmatrix} t^3 & 2(t^2+t^3) & t^2 & 0 & 0 & \dots \\ 0 & t^4 & 2(t^3+t^4) & t^3 & 0 & \dots \\ 0 & 0 & t^5 & 2(t^4+t^5) & t^4 & \dots \\ \dots & \dots & 0 & t^6 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \cdot \begin{bmatrix} \dot{q}^1 \\ \dot{q}^2 \\ \dot{q}^3 \\ \vdots \\ \dot{q}^k \end{bmatrix} = \begin{bmatrix} \frac{3}{t^2 t^3}\left[(t^2)^2(q^3 - q^2) + (t^3)^2(q^2 - q^1)\right] \\ \frac{3}{t^3 t^4}\left[(t^3)^2(q^4 - q^3) + (t^4)^2(q^3 - q^2)\right] \\ \vdots \\ \frac{3}{t^{k-1} t^k}\left[(t^{k-1})^2(q^k - q^{k-1}) + (t^k)^2(q^{k-1} - q^{k-2})\right] \end{bmatrix}
$$

Se tienen k-2 ecuaciones y k incógnitas (siendo k el número de puntos intermedios), por lo que es necesario determinar las velocidades para $\dot{q}^1$ y para $\dot{q}^k$ para hacer compatible y determinado el sistema de ecuaciones. Esta ha sido la solución escogida para el proyecto.

Por último, simplemente es necesario comprobar la validez de la trayectoria revisando los límites de las articulaciones, lo que completa la trayectoria a seguir.

Para la implementación de todo el proceso se ha hecho uso de tres programas diferentes. Dos de ellos se han extraído del libro comentado previamente [10], aunque han sido modificados:

- Código A.15: Se calculan los puntos de paso correspondientes la primera vez que se ejecuta y su salida en cada ejecución será la referencia que corresponda al instante de tiempo en el que se está. Hace llamada al Código A.16. Código de realización propia.
- Código A.16: Se realiza el cálculo de las variables articulares, velocidades y aceleraciones en función del polinomio escogido. Además grafica los valores, aunque se encuentra comentado. Hace llamada al Código A.17. Extraído de la referencia. Modificación: paso por argumento el método para seleccionar las velocidades de paso.
- Código A.17: Cálculo de las velocidades de paso para cada punto intermedio y de los coeficientes del polinomio cúbico. Extraído de la referencia. Modificación: paso por argumento el método para seleccionar las velocidades de paso y diseño del método para obtener continuidad en la aceleración además del resto de métodos.

Como ejemplo, se muestran las gráficas obtenidas del proceso completo para tres trayectorias diferentes, una puramente punto a punto interpolando las variables articulares, otra cuyos puntos intermedios sigan una línea recta y la última donde los puntos intermedios sigan una curva con las características comentadas previamente. El punto inicial será $X_0 = 17.5$, $Y_0 = 0$, $Z_0 = 30$, el final $X_f = -10$, $Y_f = -10$, $Z_f = 10$, con todas las unidades en metros. El movimiento comenzará al tercer segundo de la simulación, durará tres minutos (180 segundos) y tendrá cinco puntos intermedios. Las variables q_3 y q_4 son cero siempre en referencia, así que no se mostrarán:

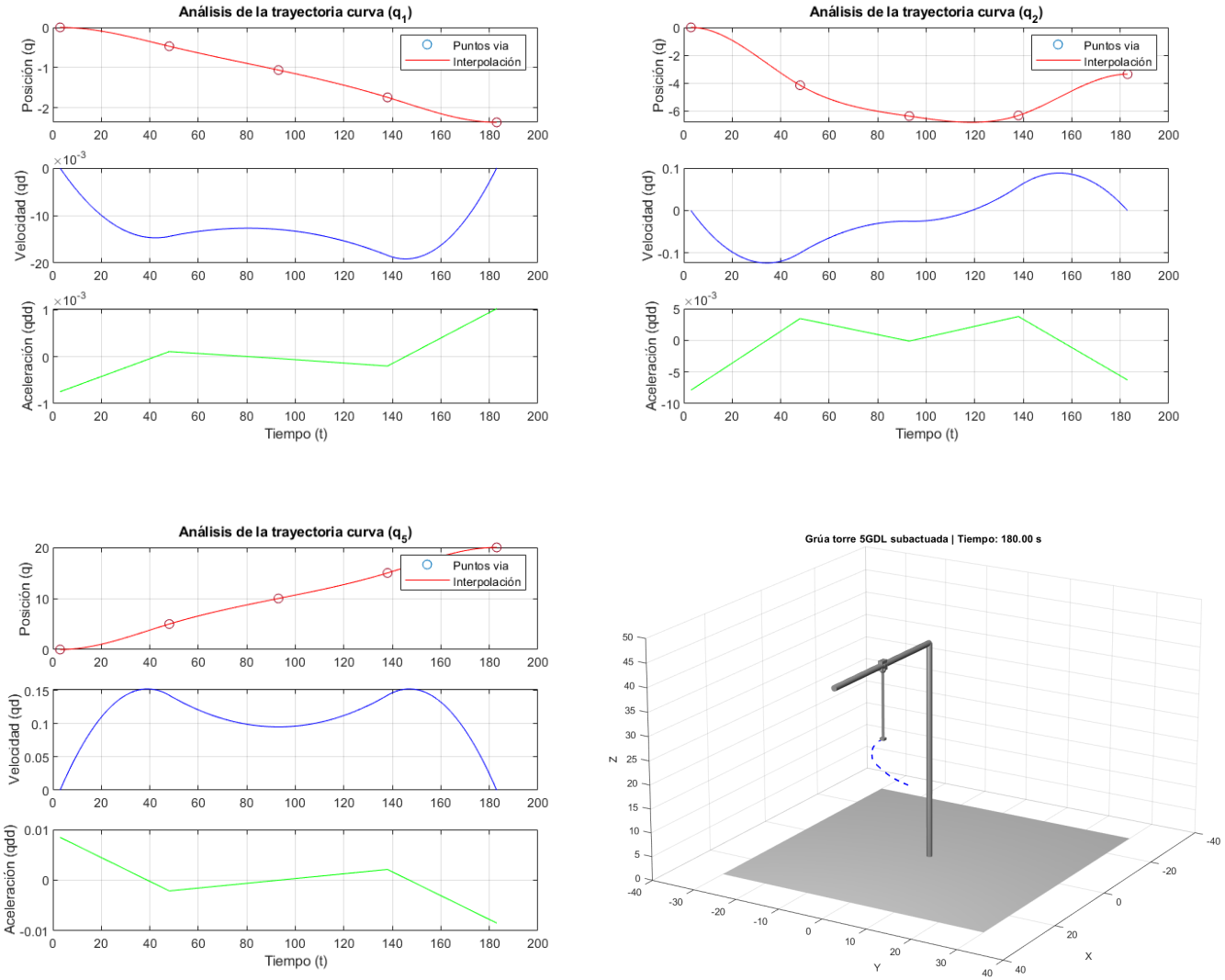


Figura 3.2 Trayectoria curva.

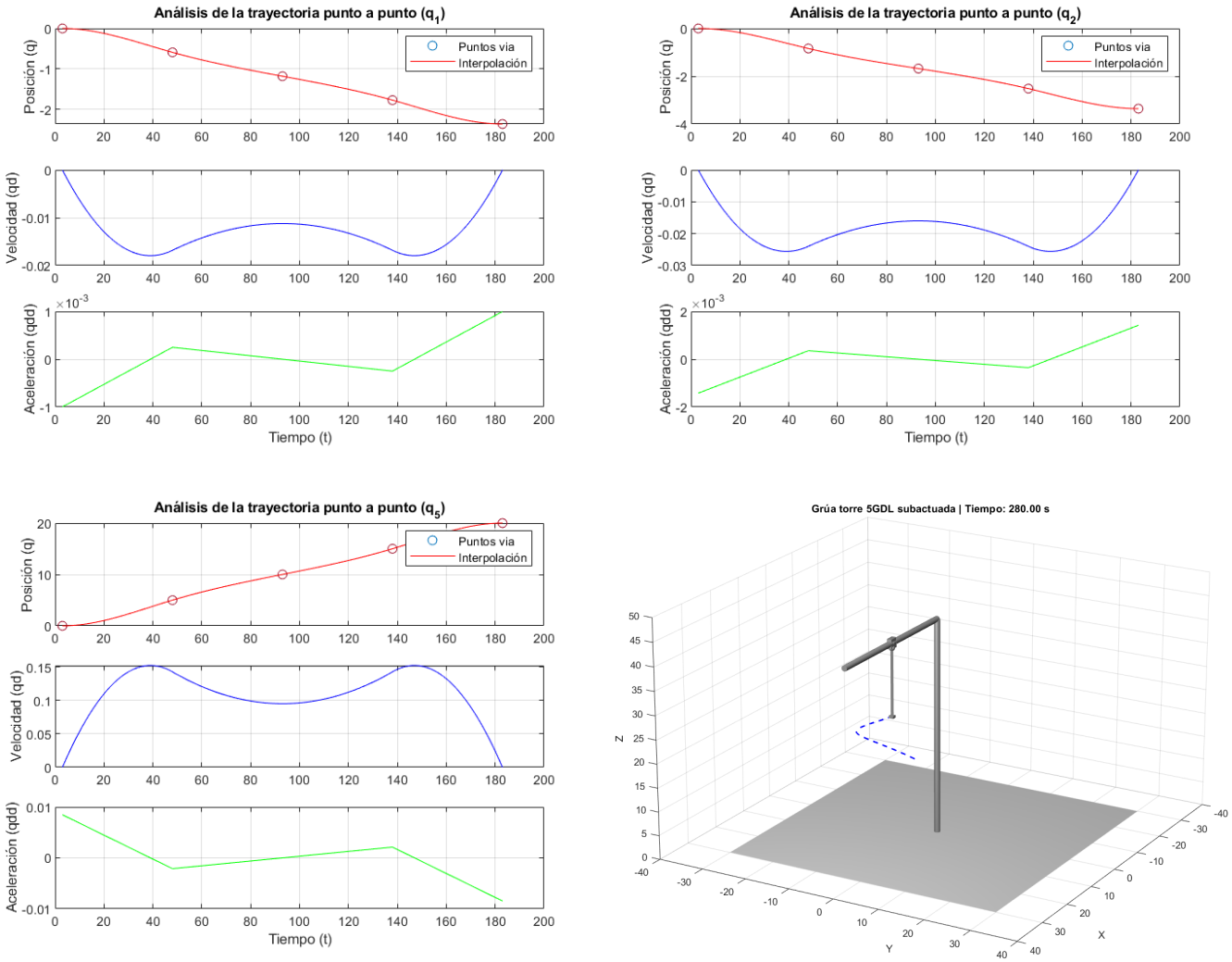
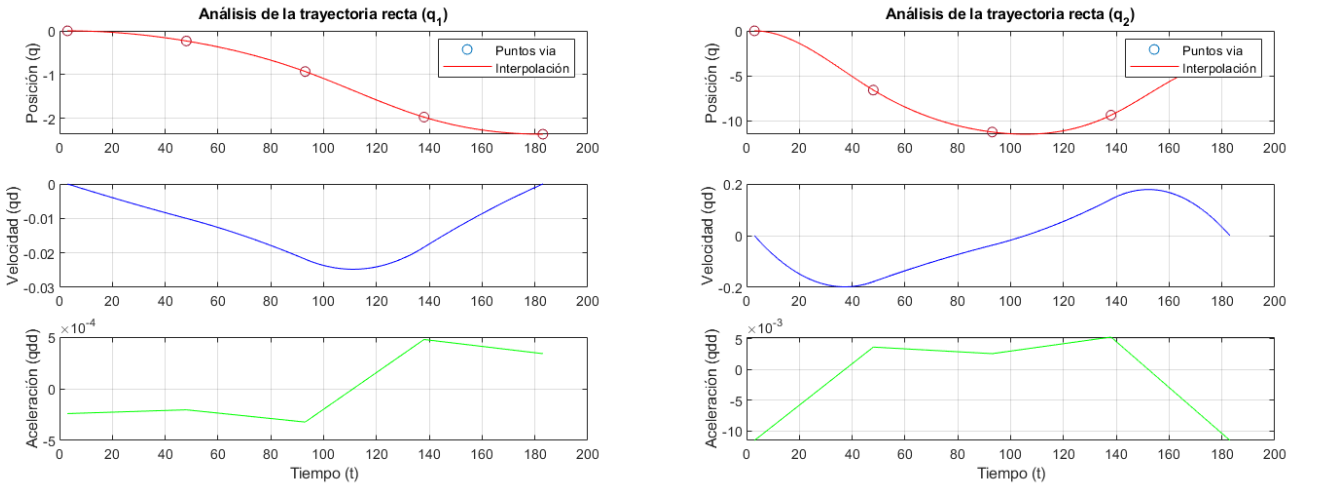


Figura 3.3 Trayectoria punto a punto.



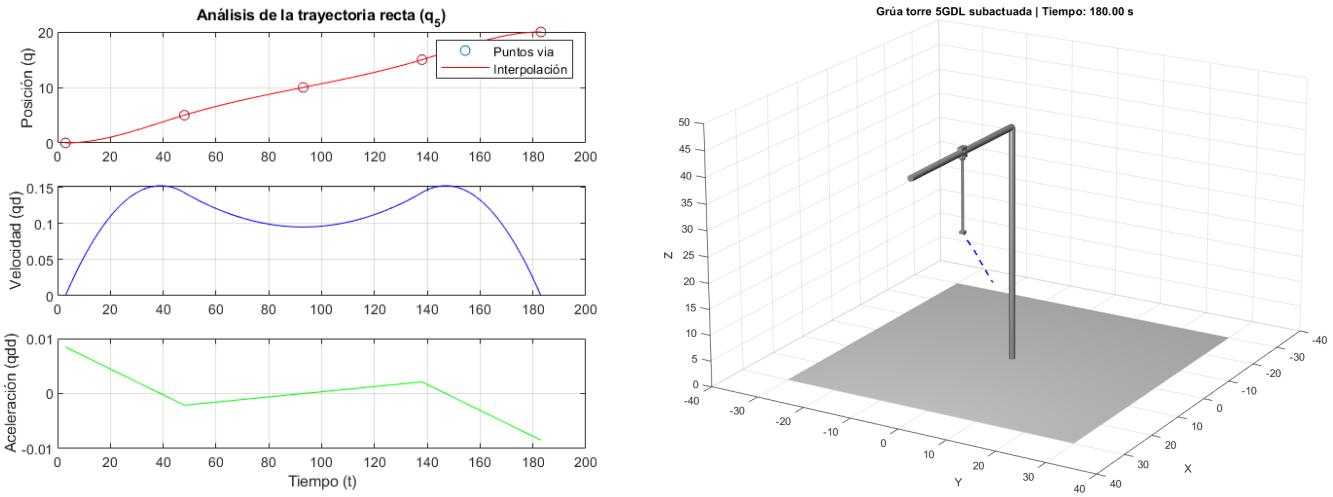


Figura 3.4 Trayectoria recta.

4 Modelo Dinámico del sistema

En este apartado se pasará a tener en cuenta el comportamiento del propio sistema más allá de su movimiento. Se buscará la relación entre las fuerzas y pares implicados en los eslabones o en el extremo de la grúa, las propiedades inerciales físicas de la grúa en cuestión; es decir, las masas de sus eslabones y las inercias, además de sus medidas y la localización del sistema, definida por sus variables articulares o las coordenadas de localización de su extremo, y sus respectivas derivadas.

El modelo dinámico de un robot, por lo general, aumenta su complejidad si aumenta el número de grados de libertad , si existe interrelación entre los movimientos de las articulaciones y si sigue relaciones no lineales. En muchos casos, no es posible su obtención de forma cerrada; es decir, no se logra escribir una única fórmula matemática explícita y finita que relacione las entradas y salidas del sistema. Normalmente, se suele depender de métodos numéricos iterativos junto a simplificaciones de ciertos términos no suficientemente relevantes.

Su importancia recae en lo útil que es a la hora de simular el movimiento del robot; permite un diseño y evaluación tanto de la estructura mecánica como del control dinámico que se realizará más adelante, esto se dede a que forma parte del propio algoritmo de control en ciertos casos. También se utiliza para el dimensionamiento de los actuadores a emplear.

Al igual que en el modelo cinemático, la parte dinámica se divide en dos. El modelado dinámico directo, en el cual se expresa la evolución temporal de las coordenadas articulares en función de las fuerzas y pares que intervienen, y el modelado dinámico inverso, que expresa las fuerzas y pares que intervienen en función de la evolución de las coordenadas articulares y sus derivadas del robot.
La relación es más que evidente y, por tanto, para hallar el modelo directo, simplemente es necesario despejar las variables correspondientes del modelo inverso.
Para la obtención del modelo inverso, se define la ecuación general dinámica de un robot multiarticular:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$$

Donde:

- q: Vector de coordenadas articulares ($n \times 1$).
- τ: Vector de fuerzas o pares aplicados a cada articulación ($n \times 1$).
- $M(q)$: Matriz de inercias simétrica y definida positiva ($n \times n$).
- $C(q,\dot{q})$: Vector de fuerzas de Coriolis y centrífugas ($n \times 1$).
- $G(q)$: Vector de fuerzas de gravedad ($n \times 1$).
- n: Número de grados de libertad del robot.

En este caso, no se ha mencionado nada acerca de los motores que controlan cada articulación. Estos tienen diferentes características que influyen en el modelo; el valor de la relación de reducción de su giro (R),

la inercia del motor (J_m) y el coeficiente de fricción viscosa (B_m).
Por ello, se reescribe la ecuación incluyendo estos términos:

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

Para su obtención, se dispone de diversos métodos:

4.1 Métodos para la obtención del modelo dinámico inverso

4.1.1 Formulación de Euler-Lagrange

Este es el método analítico utilizado habitualmente en la literatura de referencia para el diseño de controladores no lineales. Se fundamenta en el balance energético del sistema, definiendo la función Lagrangiana $\mathscr{L}=\mathscr{K}-\mathscr{P}$, donde $\mathscr{K}$ es la energía cinética y $\mathscr{P}$ la energía potencial.

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right)-\frac{\partial \mathscr{L}}{\partial q_i}=\tau_i$$

4.1.2 Formulación de Newton-Euler

Es un método vectorial recursivo que analiza el equilibrio de fuerzas y momentos en cada eslabón individualmente. Aplica la Segunda Ley de Newton para la traslación y la Ecuación de Euler para la rotación..

$$\sum F=\frac{d}{dt}(m\cdot v)$$

$$\sum T=\frac{d}{dt}(I\cdot\omega)=I\cdot\dot{\omega}+\omega\times(I\cdot\omega)$$

4.1.3 Algoritmos y elección de método

En la mayoría de los casos, la deducción analítica se vuelve excesivamente compleja con cualquier método; por ello, se desarrollan algoritmos computacionales.

- Formulación de Euler-Lagrange:
  - Basado en la representación de D-H (Denavit-Hartenberg).
  - Poca eficiencia computacional: $\mathscr{O}(n^4)$ (donde $n=$ nº grados de libertad).
  - Ecuaciones finales bien estructuradas (M,C,G aparecen por separado).
- Formulación de Newton-Euler
  - Basado en operaciones vectoriales.
  - Mayor eficiencia computacional: $\mathscr{O}(n)$.
  - Ecuaciones finales no estructuradas (los términos M,C,G aparecen sumados numéricamente).

Mientras que el método de Euler-Lagrange es idóneo para sintetizar leyes de control, el método de Newton-Euler ofrece la robustez necesaria para la validación física, además de ser más eficiente computacionalmente. A pesar de que en el segundo método se encuentran los términos sumados, es posible observar la afinidad de los sumandos con las derivadas de las variables articulares que multiplican a las matrices del modelo. Gracias a ello, se puede derivar la expresión obtenida con Newton-Euler respecto a la derivada de las variables deseadas e ir poco a poco obteniendo los elementos de las matrices del modelo.

Se sabe que las matrices obtenidas de esta forma son fiables, por lo que se va a utilizar un código que implementa el algoritmo computacional de Newton-Euler. Este fue creado por Manuel Gil Ortega en 2020, modificado por Carlos Vivas en 2023 y de nuevo retocado por el autor de este trabajo, Alejandro Noci, entre 2025 y 2026.
En él se desarrolla todo el procedimiento necesario para obtener el modelo del sistema en cuestión, es

decir, la grúa subactuada. Se encuentra comentado íntegramente de forma que se pueda seguir de manera intuitiva el algoritmo, sabiendo en todo momento el significado de cada variable, término u operación utilizada.

Para poder extraer las matrices del modelo, es necesario imponer una serie de condiciones y datos en el código; estas se enumeran a continuación:

- Elección de tipo de articulación (R: rotativa, P: prismática)
- Datos cinemáticos del robot (alturas, distancias y longitudes)
- Parámetros de Denavit-Hartenberg
- Datos dinámicos de cada eslabón (masas, centros de gravedad y matrices de inercia)
- Datos de los motores (inercias, coeficientes de fricción viscosa y factores de reducción)
- Condiciones de velocidad y giro iniciales de la base
- Fuerzas y pares aplicados en el extremo del robot

4.2 Obtención de modelos dinámicos inversos simbólicos

A la hora de extraer los datos de cualquier sistema, puede haber diferentes formas de interpretar las masas e inercias de los eslabones; debido a ello, se obtendrán dos modelos diferentes.

4.2.1 Modelo simplificado

El primero se basará en un artículo del *European Journal of Control* [14] que se centra en el control predictivo basado en el modelo con un tipo de observador de una grúa de pluma tipo torre. Inicialmente, tras la explicación del proyecto, se extrae el modelo dinámico del sistema, que es prácticamente idéntico al de este trabajo. Se deducen los siguientes datos para introducir como argumentos en el código:

- J_0 es el elemento I_{xx}, I_{yy} y I_{zz} de la matriz de inercia del eslabón uno.

$$I_{11} = \begin{bmatrix} J_0 & 0 & 0 \\ 0 & J_0 & 0 \\ 0 & 0 & J_0 \end{bmatrix}$$

- Masa puntual m, como masa de la carga en el extremo de la grúa (masa en el extremo del eslabón cinco)
- Masa puntual M_t, como masa del carrito que mueve el cable en el plano horizontal (masa del eslabón dos)

Los centros de gravedad de cada eslabón se posicionan de acuerdo con sus ejes solidarios y con las longitudes simbólicas representadas en la Figura 2.1.
Por ejemplo, el primer eslabón, el cero, tiene su centro de gravedad a una altura igual a la mitad de *L0* desde el suelo. Como su sistema de referencia solidario se encuentra a una altura *L0* y con la misma orientación que el sistema base, las coordenadas realmente serán:

$$\begin{bmatrix} 0 & 0 & -\frac{L0}{2} \end{bmatrix}$$

Para el resto de los eslabones, se calculan los centros de gravedad de forma similar.

Según este artículo, para contabilizar la inercia del eslabón uno, se supone que tiene una masa de un kilogramo. Los eslabones cero, tres y cuatro no disponen de masa. Además, a excepción del eslabón uno, como ya se ha comentado previamente, a ninguno se le asigna inercia.
Por último, no se le asigna valor a ninguna reductora de los motores de las articulaciones, por lo que se pone un uno en su lugar.

Se observa que se realizan muchas simplificaciones en el modelo. El eslabón cinco debería tener inercia definida; esto se debe a que no se trata de un eslabón concentrado en un punto como el dos. En cuanto a los eslabones tres y cuatro, es lógico que no tengan masa ni inercia porque forman parte de la articulación del péndulo esférico y es simplemente fruto del modelado; no existen en la realidad.

Al ejecutar el código y obtener las ecuaciones, se observa que coinciden con las del artículo mencionado previamente. Las únicas diferencias son la nomenclatura de las variables articulares, que se han despreciado los factores de amortiguación tratados como perturbaciones, y que las dos articulaciones prismáticas se han definido en sentido opuesto, por lo que algunos términos están cambiados de signo.

4.2.2 Modelo complejo

Para este modelo, se definen masas no incluidas en el artículo, así como *M0*, *M1* y *M5*, siendo estas las masas del eslabón cero, uno y cinco, respectivamente. También se añadirá la masa del contrapeso *Mcp*, que afecta al segundo eslabón como una masa puntual desplazada fuera del propio cuerpo. Tanto *m* como *Mt* se utilizarán tal como se incluyen en el modelo previo.

La inercia del eslabón cero se calculará sabiendo que la masa se distribuye de forma homogénea a lo largo de la barra. Se hará uso de las fórmulas de los casos particulares que se encuentran en la página web del Departamento de Física Aplicada III de la Universidad de Sevilla [15]:

$I = \frac{1}{12} \cdot M \cdot H^2$	$I = M \cdot R^2$
Inercia de una varilla rectilínea de longitud H, para un eje perpendicular por el centro	Inercia de una superficie cilíndrica de radio R y altura h para el eje del cilindro

La longitud del eslabón cero será *H* para la inercia de los ejes perpendiculares (la longitud predomina sobre las dimensiones de la sección transversal; se aproxima a una varilla). Para el eje que atraviesa longitudinalmente el eslabón, se supondrá que *R* es la media de los lados del rectángulo de su sección transversal (suelen ser huecos y la diferencia entre cilindro y prisma a nivel inercial no es significativa).

Al eslabón uno se le aplicará una dinámica parecida; sin embargo, será necesario incluir la masa del contrapeso. Para poder calcular su influencia en el centro de gravedad de la barra, se debe obtener a través de la fórmula de Steiner. Esta se encuentra reflejada y explicada en el apartado cinco de la misma referencia utilizada para el cálculo de las inercias del primer eslabón [15]:

$$\bar{\bar{I}}^O = \bar{\bar{I}}^G + m \begin{bmatrix} y_G^2 + z_G^2 & -x_G \cdot y_G & -x_G \cdot z_G \\ -x_G \cdot y_G & x_G^2 + z_G^2 & -y_G \cdot z_G \\ -x_G \cdot z_G & -y_G \cdot z_G & x_G^2 + y_G^2 \end{bmatrix}$$

x_G, y_G y z_G son las coordenadas del vector que parte del eje donde se encuentra la masa puntual *m* (*Mcp*) y que es paralelo al eje que pasa por el centro de gravedad del eslabón, en este caso.
Para aplicar esta fórmula de forma directa, sin tener que programar la ecuación cada vez que se necesite, se ha creado una función que la implementa (A.6). Se pasan como parámetros el vector de desplazamiento, la inercia que ya se conoce antes de trasladarla y la masa.

Los eslabones tres y cuatro se tratan de la misma manera que en el modelo anterior. En cuanto al eslabón cinco, su masa es la suma de *M5* y *m*. *M5* se considerará distribuida a lo largo de todo el péndulo, aunque su masa será mucho menor en comparación con *m*, suponiendo que haya carga.
El procedimiento para el cálculo de su inercia será homólogo al del eslabón uno, ya que también se debe usar la fórmula de Steiner. La inercia del eje longitudinal se va a despreciar, ya que es mínima y se trata realmente de una cuerda o polea, no de un prisma hueco como en los casos anteriores.

Las matrices M_a y V_a del modelo se corresponden con *Ma_ne* y *Va_ne* en el código; sin embargo, al ser excesivamente largas, no es posible mostrarlas en el documento. Si se copia el código A.18, que tiene

seleccionado el modelo complejo, y se ejecuta en Matlab, pueden examinarse las ecuaciones. A continuación se muestra la matriz G del modelo (*Ga_ne* en el código):

$$G = \begin{bmatrix} 0 \\ 0 \\ g\cos(q_4)\sin(q_3)\,(M_5+m)\,(0.5L_2+q_5) \\ g\cos(q_3)\sin(q_4)\,(M_5+m)\,(0.5L_2+q_5) \\ -2.0\,g\cos(q_3)\cos(q_4)\,(0.5M_5+0.5m) \end{bmatrix}$$

4.2.3 Obtención de modelos dinámicos inversos numéricos

Tal y como se ha comentado previamente, el objetivo del modelo dinámico es simular el movimiento del sistema y utilizarlo como parte del control, por lo que es necesario introducir los valores reales de las dimensiones y variables del modelo.

En el documento utilizado para extraer el modelo dinámico simplificado [14], aparecen algunos datos al final del apartado cinco que se utilizarán para ese modelo. Para el modelo dinámico complejo, se sacarán los datos de un Proyecto Final de Carrera [13]; que ya se mencionó previamente y sirvió para definir algunas variables:

- $L0 = 45$ m
- $L1B = 35$ m
- $L2 = 15$ m
- $L1A = \frac{L1B}{2}$ m

Se comienza con el modelo complejo, donde es necesario definir la distancia a la que se encuentra el contrapeso del eje central de la grúa, así como todas las masas involucradas y las características de los motores (reductora, inercia y coeficiente de fricción viscosa).

El contrapeso se encuentra en la denominada contrapluma que, según la referencia (apartado 5.3.3.), mide 10.6 metros. Se reducirá la posición de los contrapesos (Lcp) a diez metros, ya que se ha definido como la posición de la masa puntual y no toda la masa está en el extremo.

En cuanto a las masas, en el apartado 7.1.3.1.1 se especifican todas las masas necesarias. Cabe indicar que la carga a transportar puede variar; sin embargo, para el modelo se tomará que es nula. Aún así, se sabe que la carga máxima permitida para estas dimensiones es de 2 toneladas en todo su rango, pero si no se sobrepasan los 26 metros de distancia con el eje, se pueden llegar a cargar 2.5 toneladas.
El eslabón uno (la pluma) tiene una masa de aproximadamente 1236 kg (*M1*). Los contrapesos más el propio peso de la contrapluma tienen una masa conjunta de 7400 kg (*Mcp*). El cable donde se coloca la carga, según el apartado 7.4.5.1., tiene una masa de 103 kg por cada 100 metros; como el rango de operación es de alrededor de 40 metros, se supondrá una masa de 40 kg (*M5*). La masa del carro o *trolley* se va a considerar despreciable con respecto al resto de las masas, ya que, además, este elemento se encuentra incrustado en la pluma y, para los pesos dados, es insignificante. Por último, se calcula el peso de la torre (*M0*), sabiendo que en el apartado 5.4.4 se indica que el peso total de la grúa, sin contrapeso ni carga, es de 9227.31 kg (*mt*); se calcula:

$$M0 = mt - M1 = 9227.31 - 1236 = 7991.3 \quad \text{kg}$$

Para el valor R que se usa para modelar los eslabones en sí, siendo este el supuesto radio de su sección transversal, como en el punto 6.2. se indica que el lado de la sección transversal cuadrada del conjunto de vigas de la torre es de 1.2 metros, se tomará este valor también como R. El eslabón de la pluma no varía mucho en dimensiones; por eso se toma como equivalente, aunque su sección transversal sea un triángulo y no cuadrada.

Finalmente, para los motores, es necesario definir las reductoras, las inercias y los coeficientes de fricción viscosa. Se tiene que las reductoras son las siguientes según el apartado 7.2. :

- Reductora q1, R1= 94.21
- Reductora q2, R2= 91.53
- Reductora q5, R5= 177

Para las inercias, como el documento no lo especifica, pero sí indica el modelo del motor, se acude a la página oficial donde se encuentran las hojas de características [16]. A partir del nombre y los datos que se conocen del motor, se busca el dato de la inercia en el apartado *Datos técnicos de los motores*, en el punto de motores eléctricos.

- Motor q1 (M4LC4): $J_{m1} = 360 \cdot 10^{-4} \quad \mathrm{kg \cdot m^2}$
- Motor q2 (M3SA4): $J_{m2} = 34 \cdot 10^{-4} \quad \mathrm{kg \cdot m^2}$ [1]
- Motor q5 (M4LA4): $J_{m5} = 270 \cdot 10^{-4} \quad \mathrm{kg \cdot m^2}$

Los coeficientes de fricción viscosa no se especifican, ni la potencia no útil debido a las pérdidas mecánicas, ni un par de frenado correspondiente a estas pérdidas; por lo que no es posible calcularlo. La forma de estimar las pérdidas mecánicas de un motor, que se suponen prácticamente constantes independientemente del punto de operación en el que trabaje, es a través del ensayo de vacío.
Consiste en hacer girar el motor sin ninguna carga acoplada en el eje, alimentando el motor a diferentes voltajes hasta que este se detenga. Se mide la potencia absorbida en cada punto y se grafica la potencia frente al cuadrado de la tensión.
Al extrapolar esa curva hasta que la tensión sea cero, el valor de potencia resultante representa únicamente las pérdidas mecánicas (rozamiento y ventilación). Esto se debe a que, a voltaje cero, no hay magnetismo ni pérdidas en el hierro, pero el motor sigue teniendo "inercia" de giro por un instante.

Según el artículo *The Study of the Stray Load Loss and Mechanical Loss of Three Phase Induction Motor considering Experimental Results* [17], tras realizar este experimento con varios motores, se proporciona una orientación sobre la potencia perdida por fricción y ventilación en función de la potencia del motor y de los polos que lo componen.
En este caso, se tienen motores de cuatro polos con diferentes potencias, por lo que se seguirán las directrices dadas por el apartado 3.2. del artículo:

- Motor q1 (M4LC4)
  - $P = 11 \quad \mathrm{kW}$
  - [2] $P_{ml} = P \cdot 0.01 = 110 \quad \mathrm{W}$
- Motor q2 (M3SA4)
  - $P = 1.5 \quad \mathrm{kW}$
  - $P_{ml} = P \cdot 0.015 = 22.5 \quad \mathrm{W}$
- Motor q5 (M4LA4)
  - $P = 7.5 \quad \mathrm{kW}$
  - $P_{ml} = P \cdot 0.01 = 75 \quad \mathrm{W}$

Ya que se tiene este valor de las pérdidas mecánicas en potencia, se sabe que se puede expresar en función del coeficiente de fricción viscosa (B_m) y la velocidad de giro:

$$P_{ml} = B_m \cdot \omega^2$$

Si se despeja la variable deseada:

$$B_m = \frac{P_{ml}}{\omega^2}$$

Con la expresión obtenida, se calculan los coeficientes para cada uno de los motores:

[1] En el documento se indica que el motor vuelve a ser M4LC4; sin embargo, esto no es posible, ya que los datos no son iguales a los del motor de q1. Se ha elegido el motor con las especificaciones más parecidas a las que aparecen.

[2] P_{ml}: Potencia del motor correspondiente a las pérdidas mecánicas

- Motor q1 (M4LC4)

$$P_{ml} = 110 \quad \text{W}$$
$$\omega = 1440 \quad \text{rpm}$$
$$B_{m1} = \frac{110}{(\frac{1440}{60})^2} = 0.191 \quad \frac{\text{N}\cdot\text{m}}{\frac{\text{rad}}{\text{s}}}$$

- Motor q2 (M3SA4)

$$P_{ml} = 22.5 \quad \text{W}$$
$$\omega = 1390 \quad \text{rpm}$$
$$B_{m2} = \frac{22.5}{(\frac{1390}{60})^2} = 0.042 \quad \frac{\text{N}\cdot\text{m}}{\frac{\text{rad}}{\text{s}}}$$

- Motor q5 (M4LA4)

$$P_{ml} = 75 \quad \text{W}$$
$$\omega = 1440 \quad \text{rpm}$$
$$B_{m5} = \frac{75}{(\frac{1440}{60})^2} = 0.130 \quad \frac{\text{N}\cdot\text{m}}{\frac{\text{rad}}{\text{s}}}$$

Todo ello se refiere al modelo dinámico complejo. Para el modelo simple se utilizarán otros datos del documento que se usó para la comprobación de las ecuaciones de donde se extrae:

- $m = 500 \quad \text{kg}$
- $J_0 = 10215 \quad \text{kg}\cdot\text{m}^2$
- $M_t = 1140.75 \quad \text{kg}$
- $h = 48 \quad \text{m}$

En comparación con el modelo complejo, los datos son similares; tanto la altura (h, no se usa en el propio modelo) como el valor de J_0, que se asemeja a los valores de la inercia calculados en el eslabón uno. La masa de la carga se cambia a cero para que concuerde con el otro modelo y, en cuanto a la fricción de los motores, se encuentra en potencia, por lo que se tomarán los valores calculados para el otro modelo, debido a que no especifica cómo se obtienen ni sus especificaciones, y las dimensiones son parecidas.

A continuación, se muestran todos los datos seleccionados para cada uno de los modelos:

Valores modelo complejo	**Valores comunes**	**Valores modelo simplificado**
$M_t = 0 \quad \text{kg}$	$m = 0 \quad \text{kg}$	$M_t = 1140.75 \quad \text{kg}$
$M0 = 7991.3 \quad \text{kg}$	$J_{m1} = 360\cdot10^{-4} \quad \text{kg}\cdot\text{m}^2$	$J_0 = 10215 \quad \text{kg}\cdot\text{m}^2$
$M1 = 1236 \quad \text{kg}$	$J_{m2} = 34\cdot10^{-4} \quad \text{kg}\cdot\text{m}^2$	
$M5 = 40 \quad \text{kg}$	$J_{m5} = 270\cdot10^{-4} \quad \text{kg}\cdot\text{m}^2$	
$Mcp = 7400 \quad \text{kg}$	$B_{m1} = 0.191 \quad \frac{\text{N}\cdot\text{m}}{\frac{\text{rad}}{\text{s}}}$	
$R = 1.2 \quad \text{m}$		
$Lcp = 10 \quad \text{m}$	$B_{m2} = 0.042 \quad \frac{\text{N}\cdot\text{m}}{\frac{\text{rad}}{\text{s}}}$	
$L0 = 45 \quad \text{m}$		
$L1B = 35 \quad \text{m}$	$B_{m5} = 0.130 \quad \frac{\text{N}\cdot\text{m}}{\frac{\text{rad}}{\text{s}}}$	
$L1A = 17.5 \quad \text{m}$		

$$L2 = 15 \quad \text{m} \qquad\qquad R1 = 94.21$$

$$R2 = 91.53$$

$$R5 = 177$$

$$g = 9.81 \quad \text{m/s}^2$$

Por último, se muestran las matrices obtenidas en cada modelo. En el modelo complejo, tanto M como C son muy extensas y no se pueden mostrar; si se ejecuta el Código A.19 con el argumento 'C', se obtienen todas las matrices del modelo complejo:

- Modelo dinámico inverso complejo

$$G = \begin{bmatrix} 0 \\ 0 \\ 392.0 \cdot \cos(q_4) \cdot \sin(q_3) \cdot (q_5 + 7.5) \\ 392.0 \cdot \cos(q_3) \cdot \sin(q_4) \cdot (q_5 + 7.5) \\ -392.0 \cdot \cos(q_3) \cdot \cos(q_4) \end{bmatrix}$$

- Modelo dinámico inverso simplificado

$$M = \begin{bmatrix} 1140.0 \cdot {q_2}^2 + 1.02 \cdot 10^4 & 0 & 0 & 0 & 0 \\ 0 & 1140.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.027 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.001 \cdot \mathrm{qd}_1 \cdot \left(2.28 \cdot 10^6 \cdot q_2 \cdot \mathrm{qd}_2 + 191.0\right) \\ 0.042 \cdot \mathrm{qd}_2 - 1140.0 \cdot q_2 \cdot {\mathrm{qd}_1}^2 \\ 0 \\ 0 \\ 0.13 \cdot \mathrm{qd}_5 \end{bmatrix}$$

$$G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

5 Control dinámico del sistema

Con todos los modelos disponibles y comprobados, el siguiente paso natural es realizar el control dinámico del sistema de forma ideal.

Para todos los controles a realizar, hay que tener en cuenta ciertas características del sistema:

- Gran diferencia dimensional entre variables de actuación.
- Sistema altamente no lineal.
- Acoplamiento entre articulaciones.
- Sistema de cinco grados de libertad subactuado.

La primera se evidencia empíricamente; con el sistema totalmente libre, se han realizado pruebas aplicando fuerzas de forma independiente a cada una de las articulaciones. La variable articular q_1 no tiene un movimiento significativo hasta que se alcanzan los 47500 $N \cdot m$, mientras que el carro de traslación se mueve con apenas 20 N y el cable, con una fuerza ejercida por la gravedad de 300 N, se mueve lentamente, por lo que hay mucha disparidad de magnitudes.

El resto de las particularidades mencionadas se reflejan directamente en las ecuaciones dinámicas del sistema. Este modelo matemático consta de cinco expresiones (una por cada grado de libertad), donde se observa un acoplamiento mutuo entre las variables y una marcada no linealidad, derivada principalmente de las dependencias trigonométricas (senos y cosenos) asociadas a la cinemática del mecanismo.

En cuanto a los límites de las variables de actuación, en el documento que se utilizó para la comprobación de las ecuaciones dinámicas que hacía uso de otro modelo más simple [14], al final del apartado cinco especifica una masa en el extremo que usará así como unas restricciones a las señales de control de las variables actuadas:

- Masa en el extremo: 500 kg
- Límite de τ_1: $\pm$1000 Nm
- Límite de τ_2: $\pm$200 N
- Límite de τ_5: $\pm$7500 N

Sin embargo, como el modelo a utilizar tiene más en cuenta la dinámica de la grúa y no dispone de masa en el extremo, se modificarán los límites a los siguientes:

- Masa en el extremo: 0 kg
- Límite de τ_1: $\pm\ 5 \times 10^6$ Nm
- Límite de τ_2: $\pm$50 N
- Límite de τ_5: $\pm$500 N

El máximo de τ_1 se ha vuelto tan grande debido a que, al hacer un ensayo de movimiento de cada una de las variables, q_1 no comenzaba a moverse a un tiempo relativamente parecido al de las demás variables hasta que no se alcanzaba un par de 5×10^4 Nm aproximadamente.
A pesar de incorporar dichos límites, no se incluirán saturaciones, ya que han sido límites extraídos empíricamente y contrastados con el artículo comentado. Se realizarán todas las pruebas y, finalmente, se definirán. Esto también se debe a que, cuando se haga uso del efecto integral, si las señales suelen saturar, será necesario incorporar técnicas de anti-windup, y no es uno de los objetivos del proyecto.

A lo largo de este capítulo, se evaluará el desempeño de los diferentes controladores frente a tres escenarios principales: el seguimiento de referencias (trayectorias), el rechazo de perturbaciones externas y, por último, un análisis de robustez. Este último caso se validará modificando la masa en el extremo final del sistema, simulando así una carga útil real suspendida del gancho.

Las técnicas de control que se desarrollarán y compararán incluyen algoritmos PD y PID, sintonizados mediante asignación de polos y basados en el desacoplo articular; así como estrategias de control avanzado como el Regulador Cuadrático Lineal (LQR) y el Control Predictivo Basado en Modelo (MPC). Todos ellos se modelarán desde un enfoque continuo al ser probados en un entorno de simulación. Cabe destacar que la simulación no se realizará con un sistema dinámico real, sino sobre el modelo complejo calculado a partir del algoritmo Newton-Euler, por lo que se trabaja sobre un modelo ideal matemático que no refleja de forma fidedigna la dinámica del cable, que ha sufrido algunas simplificaciones y que no cambia con el tiempo.
Referente a las trayectorias de referencia, se establecen de acuerdo con la metodología descrita en el Capítulo 3. Dado que estas se componen de secuencias discretas de posiciones articulares asociadas a instantes específicos, la señal de referencia en cada instante continuo adoptará el valor correspondiente al paso de tiempo inmediatamente anterior. Se pretende que el tiempo entre muestras sea lo menor posible para poder considerar que el seguimiento de la trayectoria también es continuo. El intervalo entre referencias es de 0,05 segundos aproximadamente, por lo que no se van a realizar controladores más rápidos que ese tiempo ya que entonces se trataría como una secuencia de escalones de referencia y no es o que se busca.

Previo al diseño de cada uno de los controladores, se realizarán una serie de simplificaciones debido a la complejidad matemática del modelo. Esto es válido en muchos modelos de robots; facilitan el diseño del controlador y los resultados suelen ser aceptables. Sin embargo, esto suele aplicarse a robots cuyos grados de libertad disponen de actuación, por lo que los resultados pueden diferir en este sistema al ser subactuado. Las variables sin actuación son q_3 y q_4, por lo que no se tendrán en cuenta aunque aparezcan en las ecuaciones dinámicas.

5.1 Estructura del sistema en Simulink

Previo al desarrollo de controladores, se va a mostrar cómo se va a implementar de forma general cada uno de ellos.
Se ha creado un sistema en simulink que visualmente se verá de esta forma:

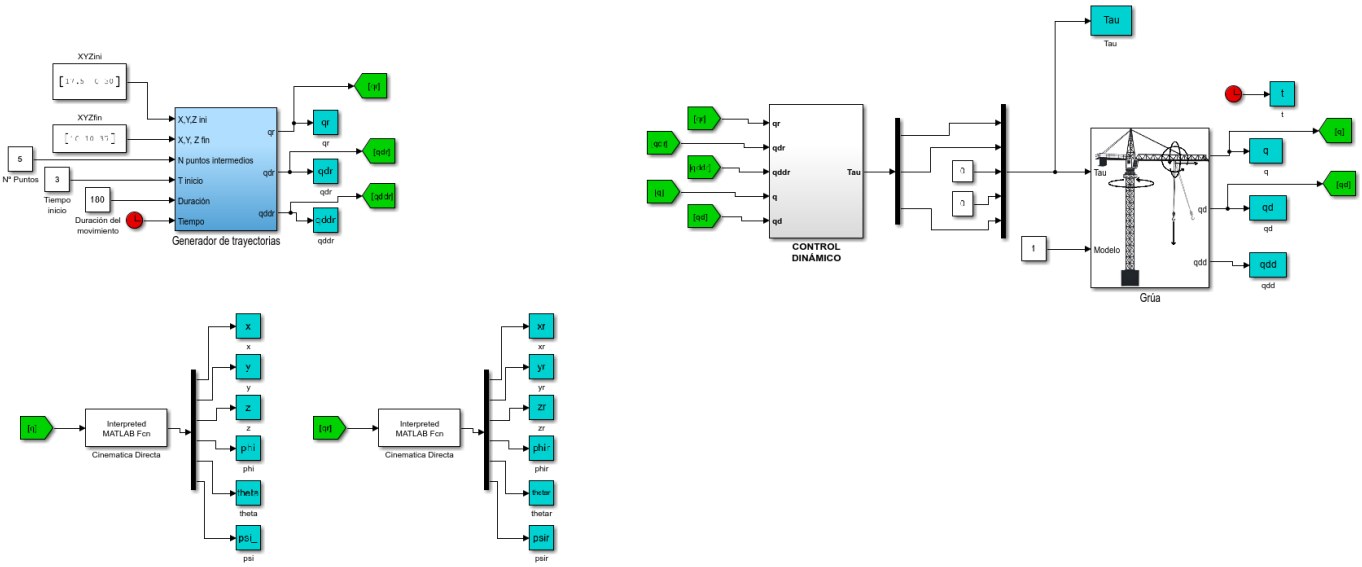


Figura 5.1 Archivo de simulink para la simulación.

Se diferencian tres grandes áreas dentro del mismo; a continuación, se desarrollarán de forma más detallada.

5.1.1 Control y sistema

Se encuentra arriba a la derecha; es el sistema en sí basado en el modelo diseñado y el control actuando sobre él:

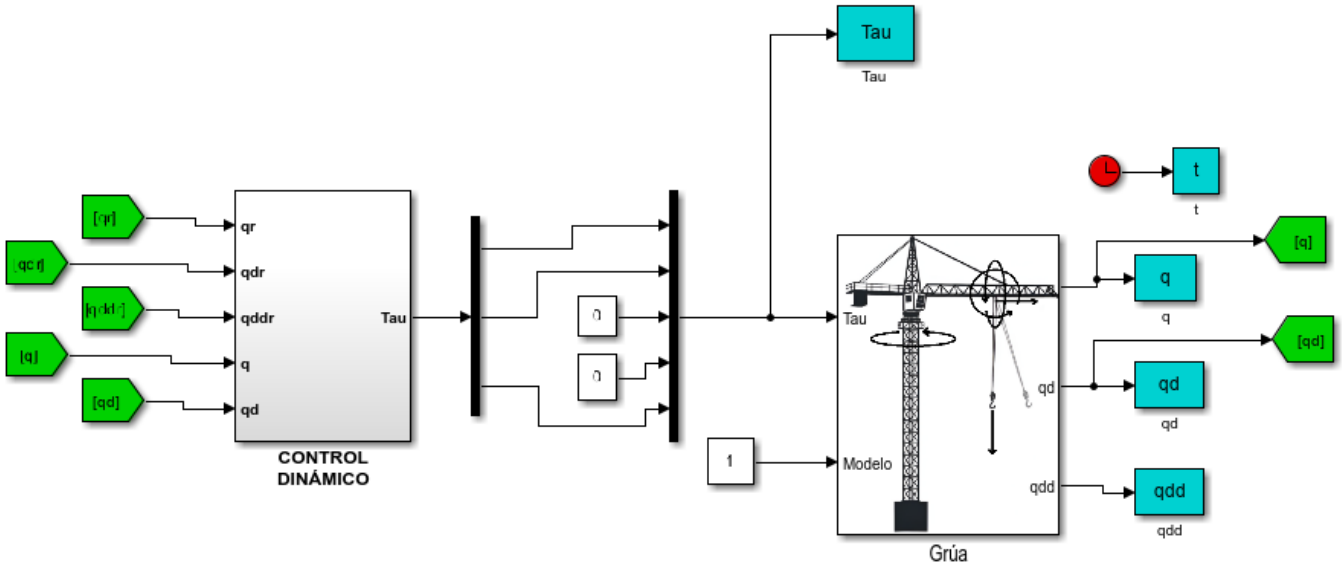


Figura 5.2 Área de control y del sistema.

La parte de control en este caso muestra el bloque que incluye la función donde se implementan los PDs, los PIDs y los controles LQR (Código A.23); en el caso del MPC, simplemente se sustituye por el bloque de Matlab.
Tiene como entradas las referencias de las variables articulares, sus velocidades y sus aceleraciones, aunque estas últimas no se utilicen; además de las salidas en velocidad y posición de las mismas. Como salidas, produce las tres señales de control asociadas a la articulación de q_1, q_2 y q_5.
Internamente, el conjunto agrupa las referencias y las salidas en un mismo vector e implementa la parte integral para cuando se requiera el efecto integral en los PID:

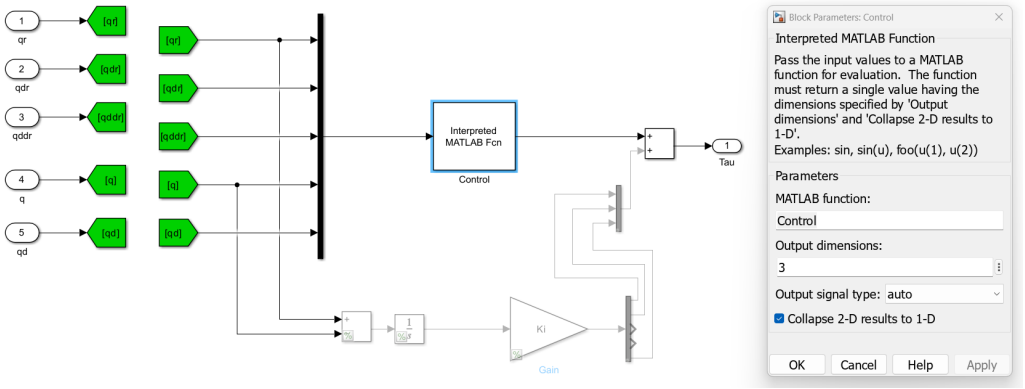


Figura 5.3 Subsistema de control.

En referencia al sistema, no es un sistema dinámico real, sino que se hace uso del modelo extraído completo para simular el comportamiento (Código A.20). Dentro del subsistema, se utilizan los integradores para extraer de la aceleración articular la velocidad y la posición, y se imponen las restricciones a las articulaciones. Si se alcanzan dichos límites, se detiene la simulación inmediatamente, lo que evita esperar a una simulación que de base, no es válida más allá de que matemáticamente sea posible:

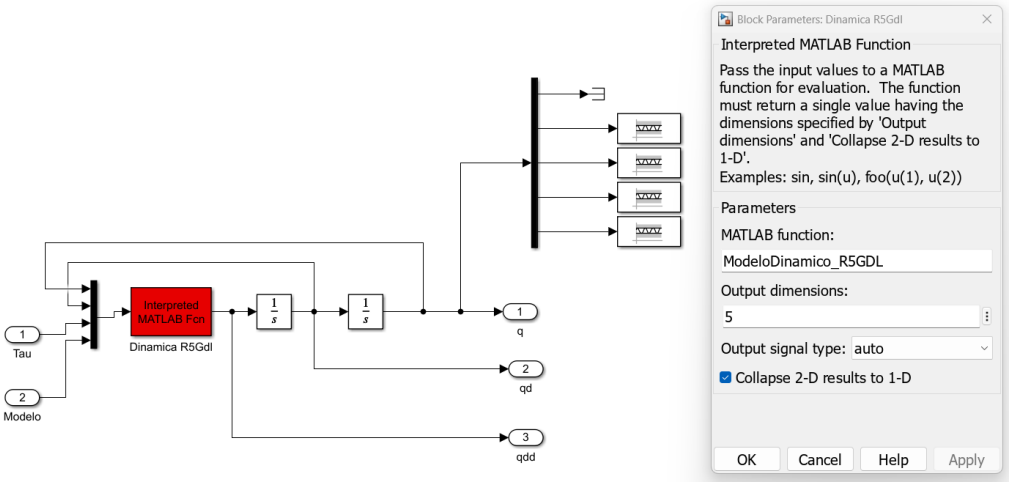


Figura 5.4 Subsistema del sistema.

5.1.2 Generador de trayectorias

Arriba a la izquierda, es dónde se gestiona la generación de la trayectoria y el envío de la referencia en cada instante.

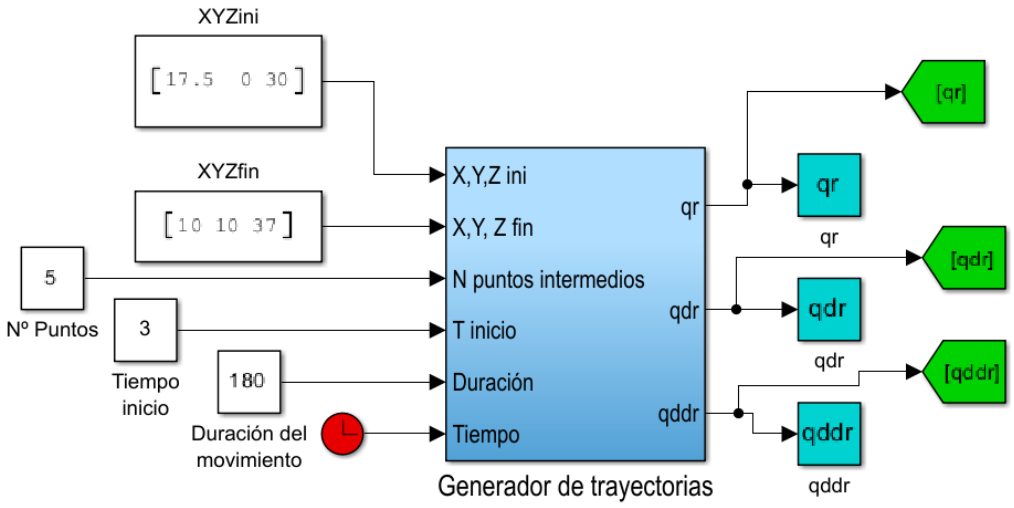


Figura 5.5 Área de generación de trayectorias.

En el Capítulo 3 se muestra el desarrollo y los códigos usados para generar la trayectoria deseada a partir de ciertos parámetros. Se incorpora el Código A.15 y se incluyen las entradas correspondientes; el punto inicial y final, el número de puntos por los que debe pasar sí o sí la trayectoria, el instante de inicio, la duración del movimiento y, por último, el instante actual en la simulación.
Dentro del subsistema no se hace más que agrupar y demultiplexar señales para que no se muestre todo en la pantalla principal y visualmente se vea más limpio:

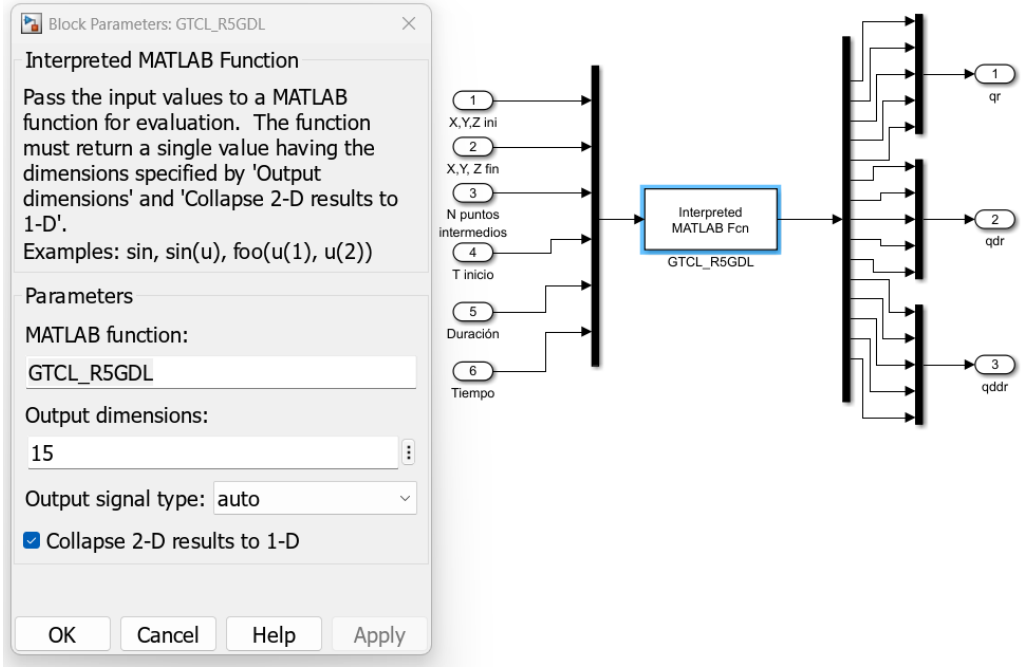


Figura 5.6 Área de generación de trayectorias.

5.1.3 Cálculo cartesiano

Por último, abajo a la izquierda, se encuentra la transformación de los datos de variables articulares a variables cartesianas.

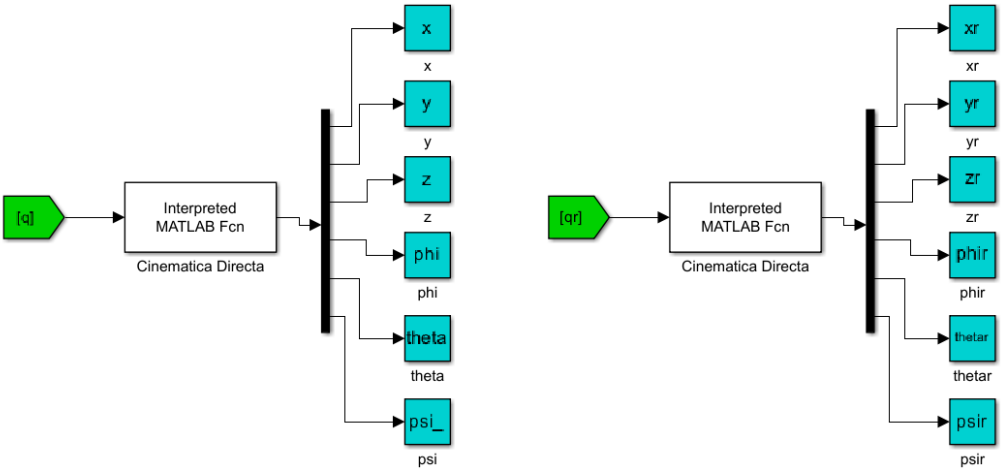


Figura 5.7 Área de cálculo cartesiano.

En este caso no hay mucha complejidad, simplemente se usa el código de la cinemática directa (Código A.9) para hacer dicho cálculo partiendo de las referencias y de las salidas del sistema.

5.2 Obtención de modelos orientados al control

5.2.1 Linealización física y desacoplamiento (Modelo SISO)

El desarrollo de la explicación se ejecuta en el Código A.21, que proporciona las funciones de transferencia de cada una de las articulaciones. Se sabe que la ecuación del modelo es

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

tal y como se ve y se explica en el Capítulo 4; sin embargo, para construir ciertos controles, es necesario linealizar el sistema.
Comenzando por la matriz de inercia, al linealizar según la expansión en series de Taylor de primer orden, queda aproximadamente la matriz en el punto de equilibrio multiplicada por el incremento de la aceleración:

$$M_a(q)\ddot{q}=M_a(q_{eq})\underbrace{\ddot{q}_{eq}}_{=0}+\underbrace{\ddot{q}_{eq}}_{=0}\left.\frac{\partial M_a(q)}{\partial q}\right|_{q=q_{eq}}\Delta q+M_a(q_{eq})\Delta\ddot{q}+\ldots\approx M_a(q_{eq})\Delta\ddot{q}$$

Para conseguir esto, es necesario hacer uso únicamente de la diagonal y elegir los valores máximos de cada término. Todo ello puede suponer un gran cambio en la dinámica del sistema, sobre todo al quedarse con la diagonalización de la matriz. De hecho, no siempre es posible; depende principalmente de los factores de reducción. Si las matrices del sistema se encuentran en torno a dimensiones relativamente parecidas, lo que suele ser muy habitual, dado que el valor de las reductoras influye cuadráticamente en la diagonal, a partir de cierto número, estos términos suelen volverse muy superiores al resto. Dicho efecto permite despreciar los términos no diagonales de las matrices.
En este caso, como los valores de las reductoras se encuentran entre 90 y 180 en función de la articulación, sus cuadrados son lo suficientemente grandes como para opacar el resto de los términos, aunque se multipliquen

por los coeficientes de inercia o de fricción viscosa del motor. La matriz queda de la siguiente manera:

$$M_a = \begin{bmatrix} A_1 & 0 & 0 & 0 & 0 \\ 0 & \frac{137}{2} & 0 & 0 & 0 \\ & & \dots & & \\ & & \dots & & \\ 0 & 0 & 0 & 0 & 886 \end{bmatrix}$$

Donde el término A_1 se define como:

$$\begin{aligned} A_1 = &\cos(q_3)\Bigg(\cos(q_3)(40q_5^2+600q_5+3000)\sin^2(q_4) \\ &+\cos(q_3)(20q_2+350)\Big(2q_2+15\cos(q_4)\sin(q_3)+2q_5\cos(q_4)\sin(q_3)+35\Big)\Bigg) \\ &+\sin(q_3)\Bigg(750\sin(q_3)+(40q_5+300)\left(\frac{35\cos(q_4)}{2}+\frac{15\sin(q_3)}{2}+q_2\cos(q_4)+q_5\sin(q_3)\right) \\ &+\sin(q_3)(20q_2+350)\Big(2q_2+15\cos(q_4)\sin(q_3)+2q_5\cos(q_4)\sin(q_3)+35\Big)\Bigg) \\ &+9120000 \end{aligned}$$

Respecto a la determinación de los valores máximos de cada término, aquellos asociados a q_2 y q_5 son constantes, por lo que pueden omitirse en este análisis. Por el contrario, el término correspondiente a q_1 depende de todas las demás variables. Para hallar su máximo, es necesario evaluar primero q_3 y q_4, que son variables no actuadas y aparecen exclusivamente como argumentos de funciones trigonométricas (seno y coseno).

Si bien el valor máximo absoluto de estas funciones es la unidad, ambas no pueden ser máximas simultáneamente para un mismo ángulo. Adicionalmente, aunque q_3 y q_4 están restringidos al intervalo $\pm\frac{\pi}{3}$, el objetivo real del sistema es que permanezcan siempre nulos. Por consiguiente, se adoptará el valor de cero como el adecuado para realizar la sustitución.
Al sustituir la diagonal queda:

$$M_{a_{diag}} = \begin{bmatrix} (2q_2+35)\,(20q_2+350)+9120000 \\ \frac{137}{2} \\ \dots \\ \dots \\ 886 \end{bmatrix}$$

q_5 ha desaparecido de la ecuación del primer término, aunque en caso de que no hubiese sido así, se le hubiera dado su valor máximo permitido, de la misma manera que se va a hacer con q_2 (14 m):

$$M_{a_{diag}} = \begin{bmatrix} 9159690 \\ \frac{137}{2} \\ \dots \\ \dots \\ 886 \end{bmatrix}$$

Además de linealizar el sistema, también está siendo desacoplado, hecho que se usará para simplificar el resto de matrices. Referente a la matriz de velocidades, como en el equilibrio las velocidades son nulas, la linealización de la matriz C de la fórmula no aporta términos; sin embargo, los términos de fricción viscosa sí aportan valor, por lo que la matriz V_a no es nula. La demostración es la siguiente:

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

$$C(q,\dot{q})\dot{q} = \underbrace{C(q_{eq},\dot{q}_{eq})}_{=0}\underbrace{\dot{q}_{eq}}_{=0} + \left(\frac{\partial C(q,\dot{q})}{\partial q}\Delta q + \frac{\partial C(q,\dot{q})}{\partial \dot{q}}\Delta\dot{q}\right)\underbrace{\dot{q}_{eq}}_{=0} + \underbrace{C(q_{eq},\dot{q}_{eq})}_{=0}\Delta q + \ldots \approx 0$$

$$R^2 B_m \dot{q} = R^2 B_m \dot{q}_{eq} + R^2 B_m \Delta\dot{q}$$

De esta forma, la matriz V_a linealizada es la siguiente:

$$V_a = \begin{bmatrix} R1^2 B_{m1} \\ R2^2 B_{m2} \\ \dots \\ \dots \\ R5^2 B_{m5} \end{bmatrix} = \begin{bmatrix} 1700 \\ 352 \\ \dots \\ \dots \\ 4070 \end{bmatrix}$$

La linealización del sistema culmina con la matriz de términos gravitatorios que se tomará como nula y su efecto real se tratará como si fuesen perturbaciones en las articulaciones correspondientes.

Para extraer las funciones de transferencia, se parte de las ecuaciones de cada articulación actuada, que, al haber linealizado el sistema, quedan de la siguiente forma:

$$\tau_1 = M_a(1,1)\ddot{q_1} + V_a(1)\dot{q_1}$$

$$\tau_2 = M_a(2,2)\ddot{q_2} + V_a(2)\dot{q_2}$$

$$\tau_5 = M_a(5,5)\ddot{q_5} + V_a(5)\dot{q_5}$$

Es necesario pasar al dominio frecuencial a través de la transformada de Laplace para poder expresar la ecuación del sistema lineal e invariante en el tiempo (LTI) como función de transferencia. La expresión general de la transformada es la siguiente:

$$f(s) = \mathscr{L}(f(t)) = \int_0^\infty f(t)e^{-st}dt$$

Sin embargo, hay tablas con las transformadas directas, por lo que hace mucho más sencillo pasar de un dominio a otro sin necesidad de aplicar la fórmula. Para las ecuaciones que se tienen, la transformación queda de la siguiente manera:

$$\tau_1(s) = M_a(1,1)\cdot q_1(s)\cdot s^2 + V_a(1)\cdot q_1(s)\cdot s$$

$$\tau_2(s) = M_a(2,2) \cdot q_2(s) \cdot s^2 + V_a(2) \cdot q_2(s) \cdot s$$
$$\tau_5(s) = M_a(5,5) \cdot q_5(s) \cdot s^2 + V_a(5) \cdot q_5(s) \cdot s$$

Una función de transferencia parte de ecuaciones en el dominio de la frecuencia, y se sabe que es una relación entrada-salida (descripción externa); es decir, relaciona incrementos de salida con incrementos de entrada en torno a un punto de operación y tiene la siguiente forma:

$$G(s) = \frac{y(s)}{u(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_0}$$

Aplicado a este caso, sabiendo que la entrada es la q y la salida es τ, a partir de las ecuaciones y tras aplicar la transformada de Laplace, se despeja para lograr la misma estructura que la de la función de transferencia:

$$G_{11}(s) = \frac{q_1(s)}{\tau_1(s)} = \frac{1}{M_a(1,1)s^2 + V_a(1)s} = \frac{1}{9.16 \times 10^6 s^2 + 1700s}$$

$$G_{22}(s) = \frac{q_2(s)}{\tau_2(s)} = \frac{1}{M_a(2,2)s^2 + V_a(2)s} = \frac{1}{68.5s^2 + 352s}$$

$$G_{55}(s) = \frac{q_5(s)}{\tau_5(s)} = \frac{1}{M_a(5,5)s^2 + V_a(5)s} = \frac{1}{886s^2 + 4070s}$$

5.2.2 Linealización por Jacobianos: Espacio de Estados (Modelo MIMO)

Para la extracción del espacio de estados, inicialmente es necesario definir los estados (x), las salidas (y) y las señales de control del sistema (u). En este caso, los estados y las salidas coincidirán y serán las variables articulares y sus respectivas velocidades, mientras que las señales de control serán las actuaciones de las variables q_1, q_2 y q_5. Se supone que todas las variables son observables.

$$x = y = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \end{bmatrix} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \qquad u = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_5 \end{bmatrix}$$

En este caso, no se van a realizar simplificaciones físicas tan determinantes como en el apartado previo. El objetivo es obtener las matrices representativas capturando todas las dinámicas cruzadas posibles del modelo. Las aceleraciones del sistema se encuentran definidas por la ecuación dinámica no lineal, que despejada resulta de la siguiente manera:

$$\ddot{q} = f(q, \dot{q}, u) = M_a(q)^{-1}(\tau - G(q) - V_a(q, \dot{q}))$$

Cualquier sistema lineal e invariante en el tiempo (LTI) tiene la siguiente forma al expresarlo en forma de espacio de estados en torno a un punto de operación:

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

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

Dado que la función de las aceleraciones es no lineal, es necesario linealizar el sistema, al igual que en el apartado anterior, evaluado en un punto de equilibrio (x_{eq}, u_{eq}). Esto se logra calculando las matrices jacobianas del sistema:

$$A = \frac{\partial}{\partial x}\begin{bmatrix} \dot{q} \\ f(q,\dot{q},u) \end{bmatrix}\Bigg|_{x_{eq},u_{eq}} \qquad B = \frac{\partial}{\partial u}\begin{bmatrix} \dot{q} \\ f(q,\dot{q},u) \end{bmatrix}\Bigg|_{x_{eq},u_{eq}}$$

Una vez evaluados los jacobianos, y asumiendo por simplicidad notacional que x y u representan a partir de ahora las desviaciones respecto al equilibrio, las matrices A y B para la grúa torre quedan estructuradas de la siguiente manera:

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0_{5\times5} & I_{5\times5} \\ A1_{5\times5} & A2_{5\times5} \end{bmatrix}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0_{5\times3} \\ B1_{5\times3} \end{bmatrix}\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_5 \end{bmatrix}$$

Donde $A1$, $A2$ y $B1$ contienen los términos cruzados resultantes de las derivadas parciales de $f(q,\dot{q},u)$. Por otro lado, al ser todas las variables de estado medibles, las matrices de salida se definen de forma directa:

$$\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = I_{10\times10}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} + 0_{10\times3}\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_5 \end{bmatrix}$$

El estado para el que se ha linealizado es el siguiente:

$$q_1 = \frac{\pi}{5} \quad rad$$
$$q_2 = 14 \quad m$$
$$q_3 = \frac{\pi}{18} \quad rad$$
$$q_4 = \frac{\pi}{18} \quad rad$$
$$q_5 = 24 \quad m$$

Se ha elegido este punto debido a varias razones; los ángulos del péndulo serán $\frac{\pi}{18}$, ya que es un valor muy cercano al único punto realmente estable sin fuerzas aplicadas, el cero, que es el también es el deseado. En caso de haber escogido el cero para ambas variables, hubiera generado grandes simplificaciones en las matrices, omitiendo información de acoplamiento de variables a los controladores que hagan uso del espacio de estados.

Para q_1 y q_2, cualquier valor es estable. En cambio cualquier punto de q_5 es inestable debido a la gravedad, sin embargo las aceleraciones que genera desde el reposo son muy pequeñas. Por tanto, para q_1 se ha tomado un valor cualquiera que no sea cero para que no anule términos, y con q_2 y q_5 se ha optado por los valores máximos, siendo la posición obtenida la más desfavorable.

El resto de variables, es decir; las señales de actuación, las velocidades de las articulaciones y sus aceleraciones, se han tomado como nulas suponiendo que se parte del reposo.

Las matrices A y B obtenidas del espacio de estados son:

$$A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 1.0\times10^{0} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.0\times10^{0} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0\times10^{0} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0\times10^{0} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0\times10^{0} \\
0 & -6.529\times10^{-6} & 1.203\times10^{-4} & -1.133\times10^{-3} & -2.352\times10^{-7} & -1.864\times10^{-4} & -4.445\times10^{-5} & 0 & 0 & -1.062\times10^{-4} \\
0 & -1.508\times10^{-5} & 1.020\times10^{1} & -6.847\times10^{-1} & 5.444\times10^{-3} & -2.152\times10^{-4} & -1.160\times10^{1} & 0 & 0 & 1.036\times10^{0} \\
0 & 1.604\times10^{-6} & -6.123\times10^{-1} & 1.223\times10^{-3} & 3.387\times10^{-3} & 3.907\times10^{-5} & 3.615\times10^{-1} & 0 & 0 & -3.225\times10^{-2} \\
0 & -1.377\times10^{-5} & 2.942\times10^{-2} & -2.873\times10^{-1} & 1.549\times10^{-3} & -2.127\times10^{-4} & -1.095\times10^{-2} & 0 & 0 & 8.519\times10^{-4} \\
0 & -3.109\times10^{-6} & -2.423\times10^{-1} & -6.821\times10^{-2} & -4.209\times10^{-5} & -4.436\times10^{-5} & 8.957\times10^{-2} & 0 & 0 & -4.602\times10^{0}
\end{bmatrix}$$

$$B = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
1.096\times10^{-7} & 1.263\times10^{-7} & 2.610\times10^{-8} \\
1.266\times10^{-7} & 3.296\times10^{-2} & -2.545\times10^{-4} \\
-2.298\times10^{-8} & -1.027\times10^{-3} & 7.924\times10^{-6} \\
1.251\times10^{-7} & 3.111\times10^{-5} & -2.093\times10^{-7} \\
2.609\times10^{-8} & -2.545\times10^{-4} & 1.131\times10^{-3}
\end{bmatrix}$$

Además, se sabe que C es la matriz identidad y D es nula. De esta forma, queda definido el espacio de estados.

A la hora de realizar el código (Código A.22), se ha calculado directamente el jacobiano, sustituyendo posteriormente, y se observa que las observaciones previas se han cumplido a la perfección, validando de esta forma la estructura del sistema en el espacio de estados. En el código se pueden encontrar tres sistemas distintos; realmente es el mismo, pero para algunos controladores ha sido necesario modificar levemente algunas características.

5.3 Diseño de estrategias de control: Seguimiento de trayectorias

5.3.1 PD y PID: Control monoarticular

El control monoarticular parte de la suposición de que dichas articulaciones están desacopladas, por lo que se utilizarán las funciones de transferencia obtenidas en la linealización física del modelo, que permiten tratar las articulaciones de forma independiente.

Como se ha comentado previamente, se van a sintonizar los controladores siguiendo una asignación de polos. Esto se debe a que el modelo, además de haberse simplificado con la linealización, tampoco era exacto, por lo que realizar un buen control con diseño analítico puede no dar los resultados esperados.
De la misma hipótesis se descarta la opción de realizar cancelación dinámica, ya que lo más probable es que la cancelación no sea exacta y el comportamiento difiera por completo del esperado.

Por tanto, la asignación de polos se realizará con el lugar de las raíces, ya que permite una perspectiva gráfica de las posibilidades de diseño. El lugar de las raíces es el lugar geométrico que forman los polos en bucle cerrado de un sistema al variar la ganancia del sistema en bucle abierto.

El procedimiento a seguir parte de representar los polos y ceros de la función de transferencia de cada una de las articulaciones dentro del lugar de las raíces y, posteriormente, añadir el cero adicional en el caso del PD, o los dos ceros y el integrador en el caso del PID. El valor de los ceros se estimará en función del tiempo de subida deseado, es decir, la rapidez del control; aunque al tratar la referencia como continua, no se apreciará la respuesta ante escalón. Posteriormente, se observará la respuesta temporal que proporciona la propia herramienta de Matlab y se extraerán los parámetros.

Antes de realizar un ejemplo completo, diseñar diferentes controladores y comprobar en las simulaciones, es necesario analizar las funciones de transferencia obtenidas:

Articulación 1 (rotación de la grúa sobre su eje):

$$G_{11}(s) = \frac{1}{9.16 \times 10^6 s^2 + 1700s}$$

Articulación 2 (movimiento de traslación del carrito):

$$G_{22}(s) = \frac{1}{68.5s^2 + 352s}$$

Articulación 5 (elongación del cable del que cuelga la carga):

$$G_{55}(s) = \frac{1}{886s^2 + 4070s}$$

Todas las funciones de transferencia son de orden dos y de tipo uno; es decir, el mayor grado de la variable independiente en el denominador es dos (orden) y contiene un único polo (raíces del denominador) igual a cero (tipo uno). A estos polos se les denomina integradores.
La característica fundamental de este tipo de sistemas es que son inestables, ya que uno de sus polos está en el cero, por lo que la respuesta ante una entrada constante; como un escalón, aumentará sin parar, haciéndolos inestables. Se interpreta como una dinámica infinitamente lenta que nunca alcanza la estabilidad. En cambio, son controlables por lo que es posible llevarlos a un punto de estabilidad; más aún, el sistema real no sigue esa función de transferencia exacta por lo que puede que sea estable en la realidad.
A pesar de ello, el otro polo también aporta influencia en la respuesta ante la entrada ya que es como si fuese un sistema de primer orden con una respuesta exponencial que termina siendo estable. Al multiplicarse con el integrador realmente se combinan ambas dinámicas, cuando el polo termina su dinámica exponencial el integrador sigue aumentando llevando la respuesta al infinito.
La articulación dos y cinco tienen su polo en torno al menos cinco en la recta real, lo que quiere decir que su constante de tiempo es de aproximadamente 0,2 segundos. Si el tiempo de subida es de aproximadamente tres veces esta constante, en tres cuartos de segundo el integrador domina sobre el polo. En cambio, para la primera articulación, el polo se encuentra en 0,00018, por lo que su constante de tiempo es muy superior a las anteriores, siendo mayor de 5000 segundos. Claramente, esta articulación necesitará señales de control muy altas para poder controlarse más rápidamente debido a su lenta dinámica, y muy probablemente no se podrá tratar de la misma forma que las otras. En el control multiarticular se abordará la descompensación de las dinámicas articulaciones, que necesitará tratamiento adicional; pero en este caso no es necesario al partir directamente de la función de transferencia y actuar de forma independiente.

El diagrama para incorporar el control a las articulaciones por separado es el siguiente:

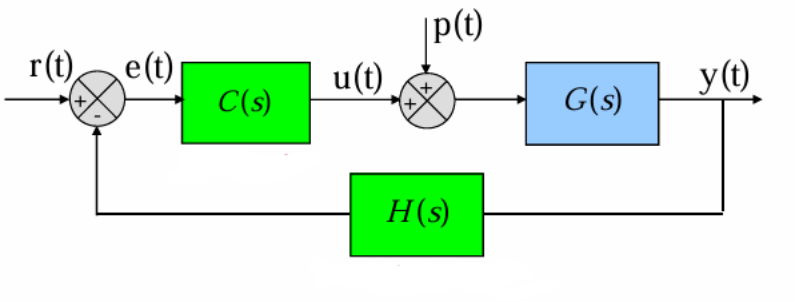


Figura 5.8 Diagrama de control.

En él $G(s)$ corresponde a la función del sistema (cada una de las articulaciones) en el dominio frecuencial mientras que $C(s)$ es el controlador y $H(s)$ la función que podrían seguir las medidas si no fuesen exactas, pero como en este proyecto se consideran ideales e inmediatas, $H(s)$ es igual a uno.
La expresión que va a definir el seguimiento del sistema una vez que se implemente el controlador va a ser la función en bucle cerrado que es la siguiente:

$$G_{BC}(s) = \frac{y(s)}{r(s)} = \frac{C(s)G(s)}{1+C(s)G(s)H(s)}$$

Al analizar las expresiones, la función de la articulación que se va a controlar al tener un integrador asegura por sí misma (sin necesidad de integrador en el controlador) errores en régimen permanente nulos en bucle cerrado ante escalones de referencia.

Controles PD

Inicialmente, se sintonizará un controlador tipo PD con el mismo tiempo de subida para las tres articulaciones actuadas. La estructura de la función de transferencia de este controlador es:

$$C(s) = K_p + K_d \cdot s = K_p(1 + T_d \cdot s)$$

Para implementarlo se aplica la expresión en el dominio temporal que corresponde a:

$$u(t) = K_p \cdot e(t) + K_d \cdot \frac{d\ e(t)}{dt}$$

Este tiempo de subida en bucle cerrado debe ser menor que los polos más rápidos de las articulaciones, ya que sino, el efecto del PD pierde sentido, así como el del control. El tiempo escogido inicialmente será de 0,1 segundos.

Lo primero es obtener la función de transferencia en el lugar de las raíces; el ejemplo se hará para la articulación dos:

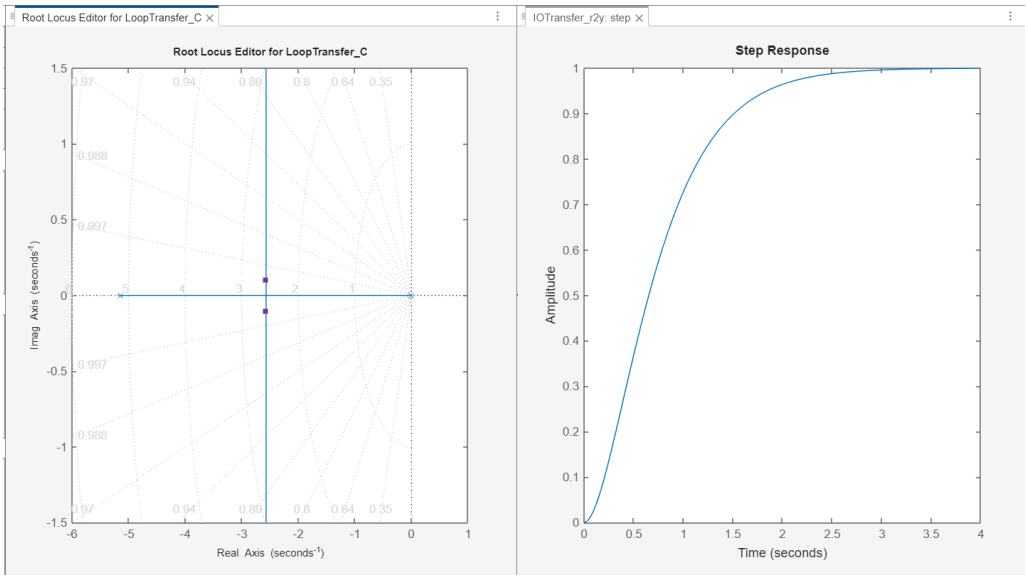


Figura 5.9 Lugar de las raíces de la función de transferencia de la articulación dos.

El cero del controlador se coloca aproximadamente de forma que, al sumarse los tiempos de los polos en bucle cerrado, den el tiempo de subida requerido. Se ajustará la ganancia en bucle abierto para que los polos en bucle cerrado sean iguales, por lo que cada uno aportará la mitad del tiempo de subida. Estos polos deberán situarse en torno a la inversa de un tercio del tiempo que les corresponda, es decir:

$$p_{bc} = -\frac{3}{\frac{0,1}{2}} = -\frac{3}{0,05} = -60$$

El cero se colocará cerca de -30 para colocar ambos polos en el -60. El resultado es el siguiente:

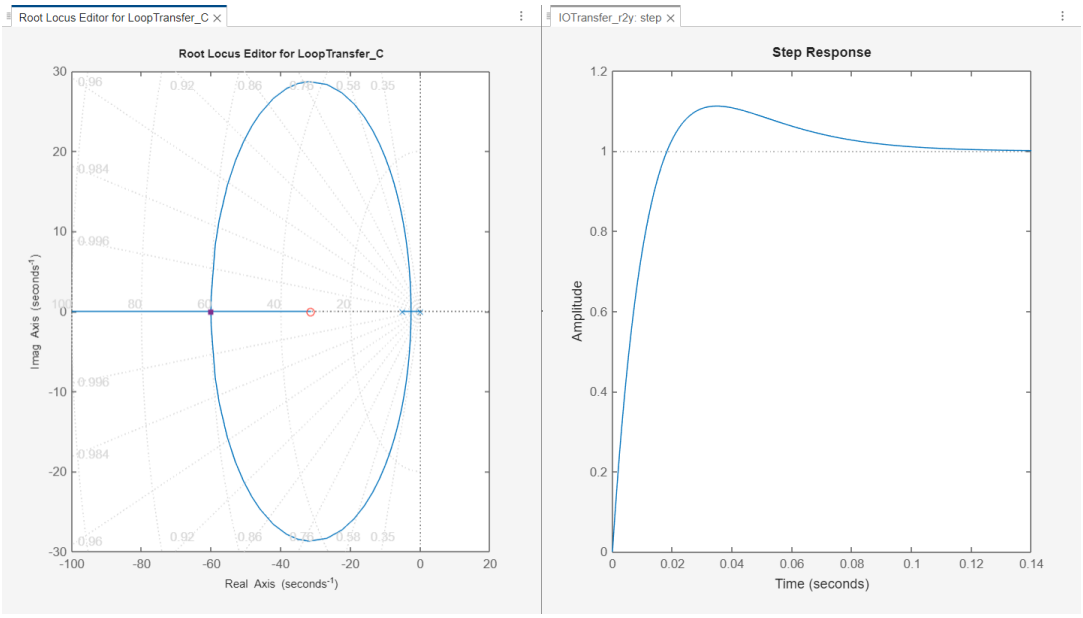


Figura 5.10 Lugar de las raíces de la función de transferencia de la articulación dos con un cero en el controlador.

Como se puede observar, hay una sobreoscilación que provoca que se alcance la referencia antes de lo previsto; aún así, se establece en torno a los 0,1 segundos previstos.
Se extraen los parámetros del controlador:

- $K_{p2} = 247260$
- $K_{d2} = 247260 \cdot 0,032 = 7912,32$

Los parámetros del resto de las articulaciones se calculan de la misma forma, quedando:

- $K_{p1} = 33291000000$
- $K_{d1} = 33291000000 \cdot 0,033 = 1098603000$
- $K_{p5} = 3229800$
- $K_{d5} = 3229800 \cdot 0,032 = 103353,6$

Se incorporan dichos parámetros en el código del controlador (Código A.23) y se realiza una simulación. Esta consta de un movimiento de 180 segundos que comienza a los 3 segundos, interpolando en variables articulares. El desplazamiento se da entre los puntos (17,5 0 30) y (10 10 37) pasando por cinco puntos de paso. De dicha trayectoria se obtienen los siguientes resultados:

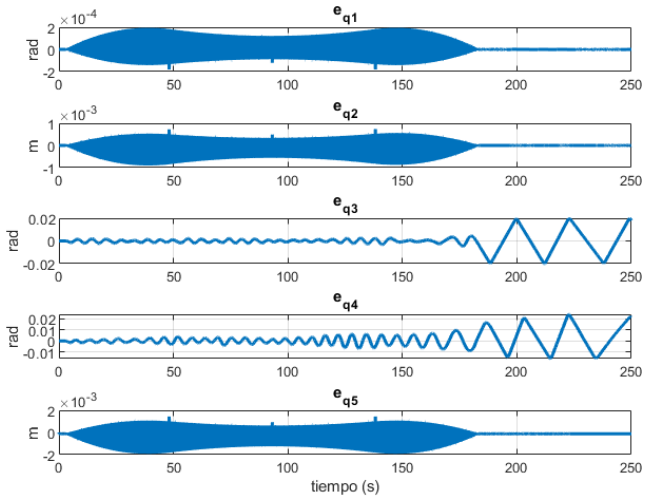


Figura 5.11 Errores en las articulaciones. PD sintonizado a 0,1 segundos.

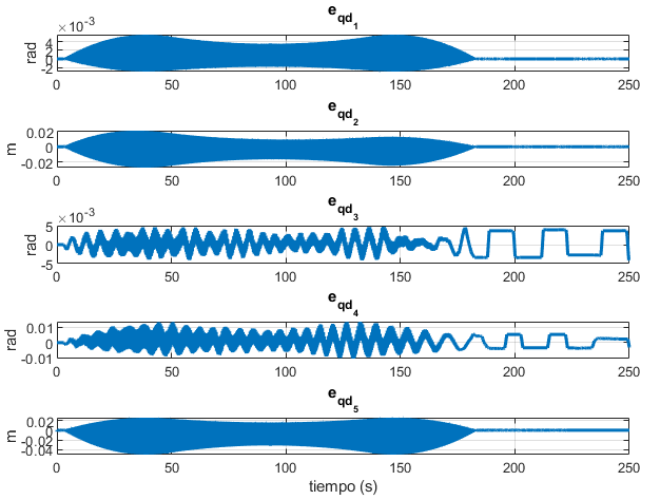


Figura 5.12 Errores en las velocidades articulaciones. PD sintonizado a 0,1 segundos.

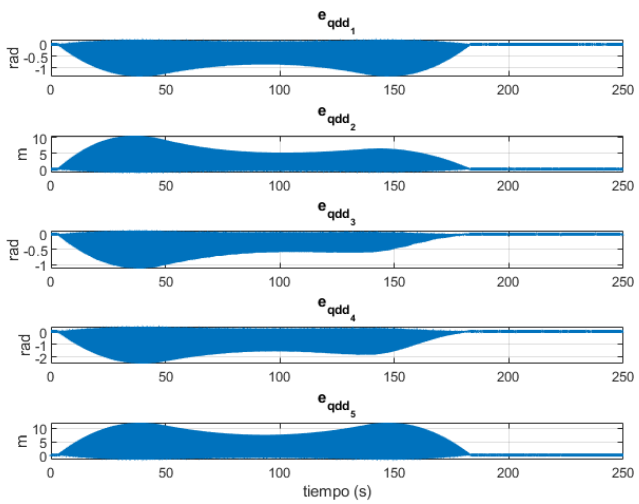


Figura 5.13 Errores en las aceleraciones articulaciones. PD sintonizado a 0,1 segundos.

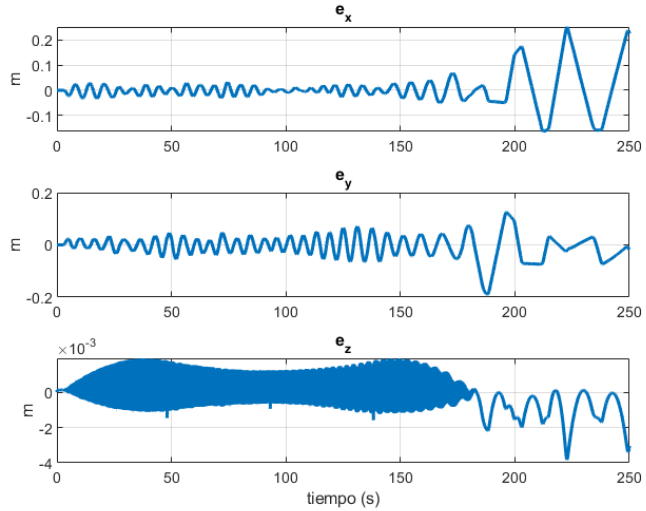


Figura 5.14 Errores cartesianos. PD sintonizado a 0,1 segundos.

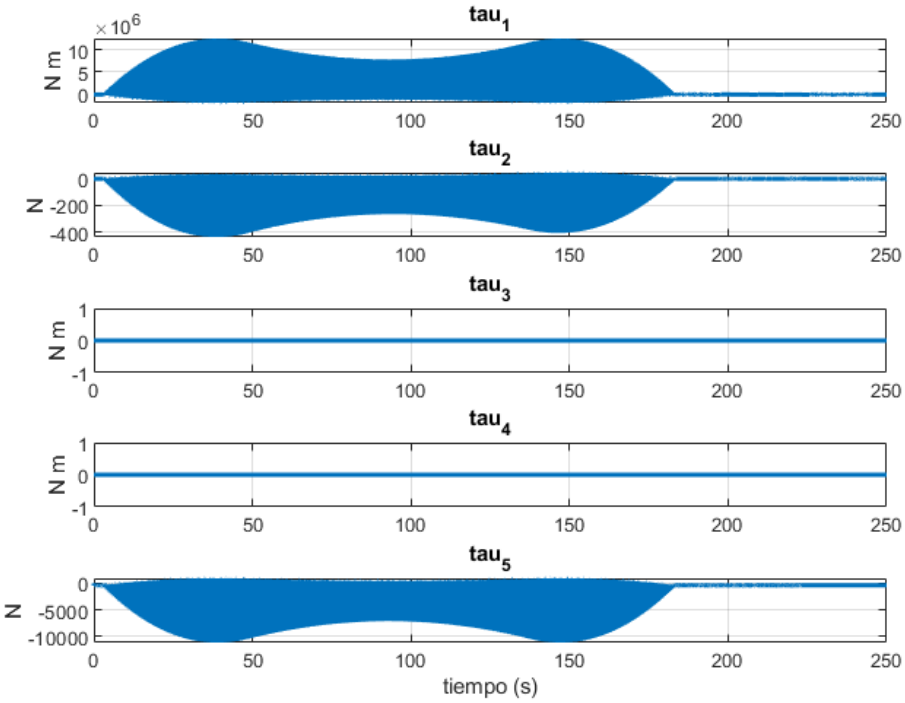


Figura 5.15 Señales de control. PD sintonizado a 0,1 segundos.

Como se puede observar, los errores son bastante pequeños tanto en las articulaciones como en el espacio cartesiano, lo que indica que el control en seguimiento es bueno; sin embargo, las señales de control son muy altas, oscilan mucho y se aprecia, en los errores de velocidad y aceleración, que son mayores. Puede ser que físicamente no sea posible dicha variación en las señales de control y que fallen los actuadores. Se buscará ralentizar el tiempo de subida para disminuir este efecto, que también puede ser provocado por las oscilaciones del péndulo no controladas, lo que indicaría que no se puede solucionar con este tipo de control. A partir del segundo 183, el control recibe una referencia fija y constante, ya que termina el movimiento. Debido a esto, los errores en las articulaciones actuadas son prácticamente cero y las señales de control son pequeñas; en cambio, se acentúan los errores del péndulo (q_3 y q_4); si estaba en un punto de inestabilidad (distinto de cero), mantiene la inercia que llevaba y se balancea

El movimiento planteado realmente es lento. La grúa es muy grande y, aunque sea un sistema automático, debe mantener unos niveles seguros de velocidad y aceleración; por lo que se aumentará el tiempo de subida a 0,5 segundos. Los parámetros calculados para cada PD son:

- $K_{p1} = 1318200000$
- $K_{d1} = 1318200000 \cdot 0,17 = 224094000$
- $K_{p2} = 9923,9$
- $K_{d2} = 9923,9 \cdot 0,13 = 12901,07$
- $K_{p5} = 127130$
- $K_{d5} = 127130 \cdot 0,13 = 16526,9$

Se realiza la misma simulación; se exponen los resultados a continuación:

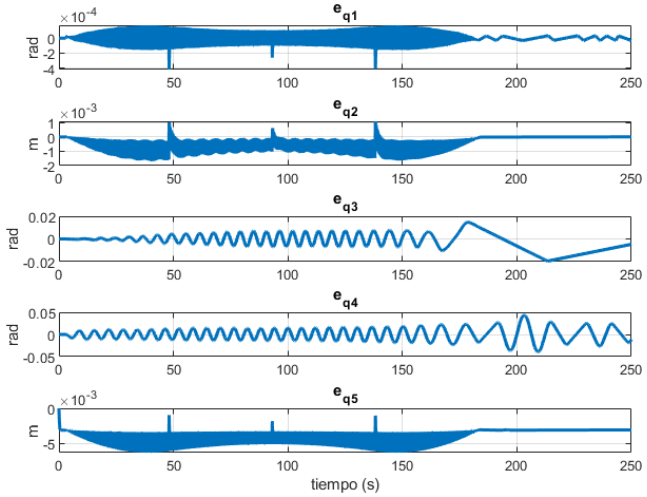


Figura 5.16 Errores en las articulaciones. PD sintonizado a 0,5 segundos.

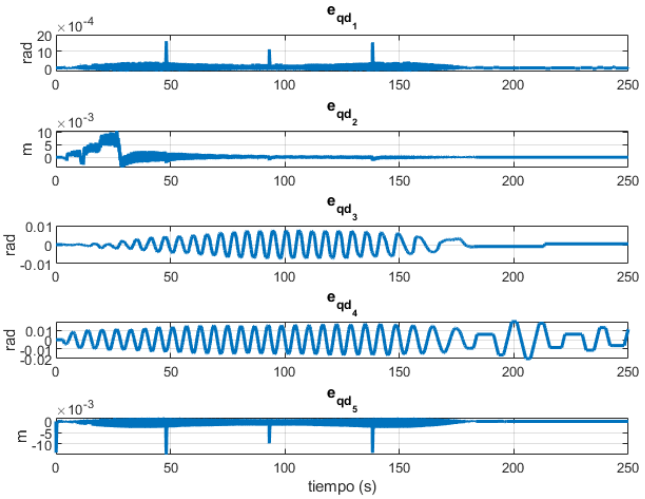


Figura 5.17 Errores en las velocidades articulaciones. PD sintonizado a 0,5 segundos.

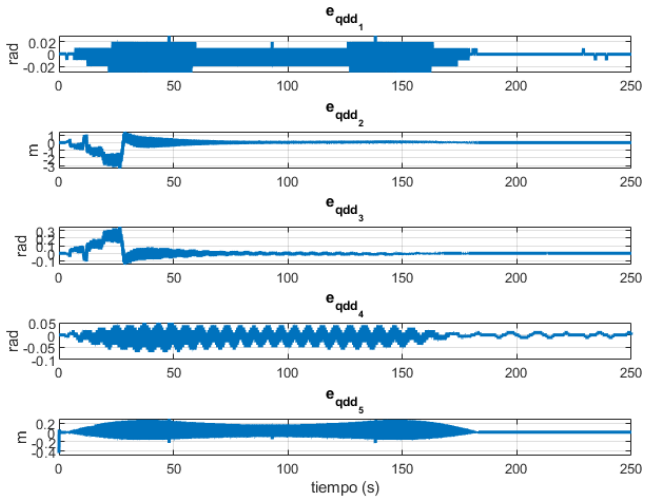


Figura 5.18 Errores en las aceleraciones articulaciones. PD sintonizado a 0,5 segundos.

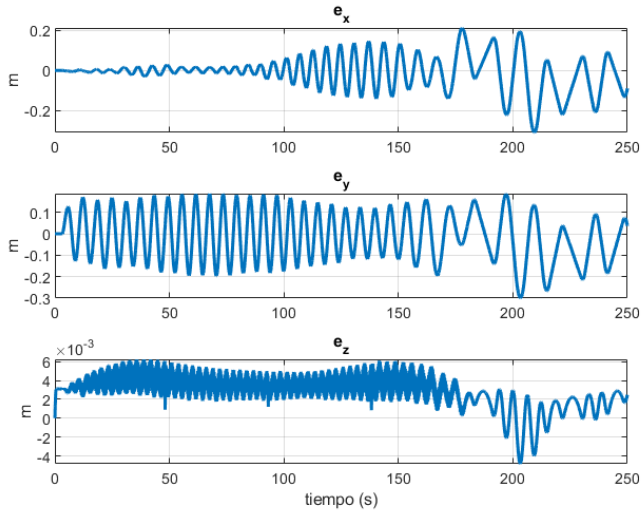


Figura 5.19 Errores cartesianos. PD sintonizado a 0,5 segundos.

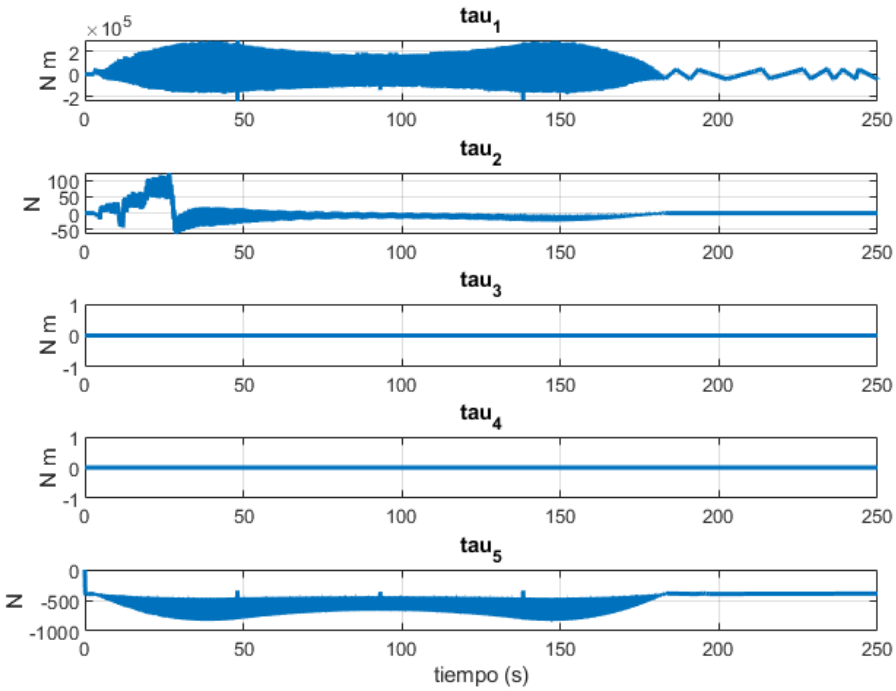


Figura 5.20 Señales de control. PD sintonizado a 0,5 segundos.

Tal y como se preveía, las señales de control disminuyen considerablemente y también mejoran los errores en las articulaciones, tanto en posición como en velocidad y en aceleración. Sin embargo, las oscilaciones del péndulo no se amortiguan tan bien y aumentan sus errores, lo que conlleva que, en coordenadas cartesianas, también se acentúan estas desviaciones con respecto a la referencia. En conclusión, parece que las oscilaciones en las señales de control se deben a estos errores inducidos por el balanceo del efector final, por lo que no se puede mejorar con un control PD.

En este punto, el tiempo de subida no puede aumentarse mucho más para q_2 y q_5, pero para q_1, que es la articulación más lenta, se realizará un control con un mayor tiempo de subida para limitar la agresividad de su señal de control.
El tiempo propuesto en este caso será de dos segundos, y los parámetros de q_2 y de q_5 se mantienen, mientras que los del control de q_1 son:

- $K_{p1} = 82481000$
- $K_{d1} = 82481000 \cdot 0,67 = 55262270$

Cuyos resultados son los siguientes:

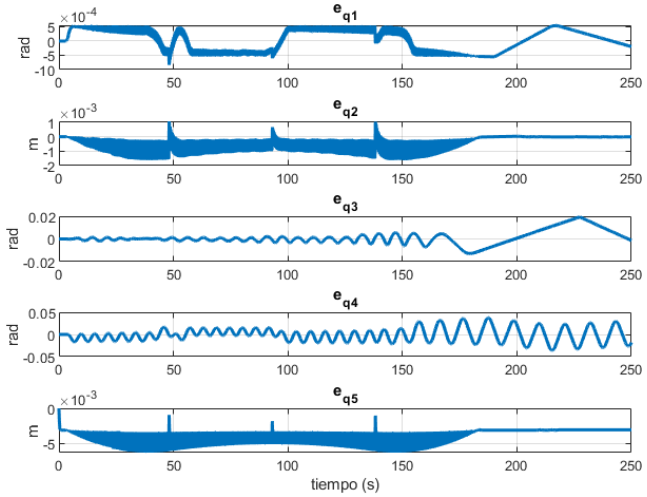


Figura 5.21 Errores en las articulaciones. PD de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

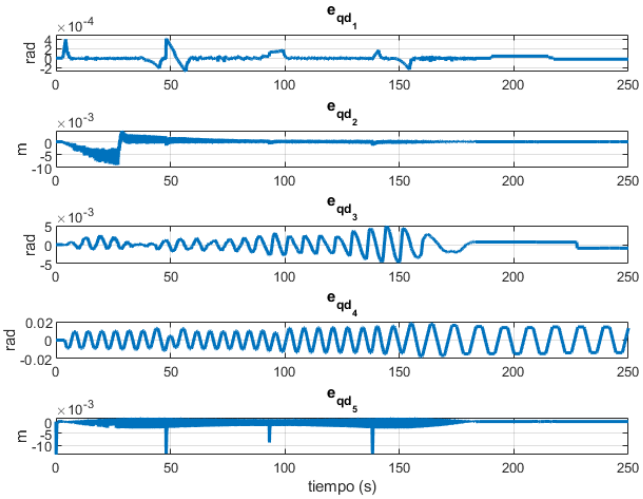


Figura 5.22 Errores en las velocidades articulaciones. PD de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

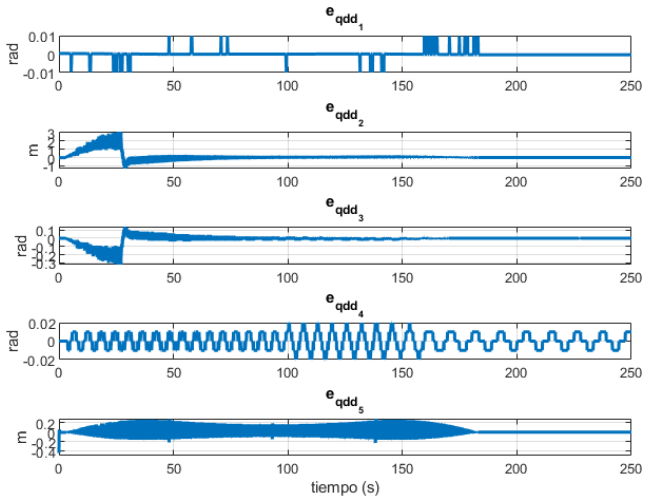


Figura 5.23 Errores en las aceleraciones articulaciones. PD de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

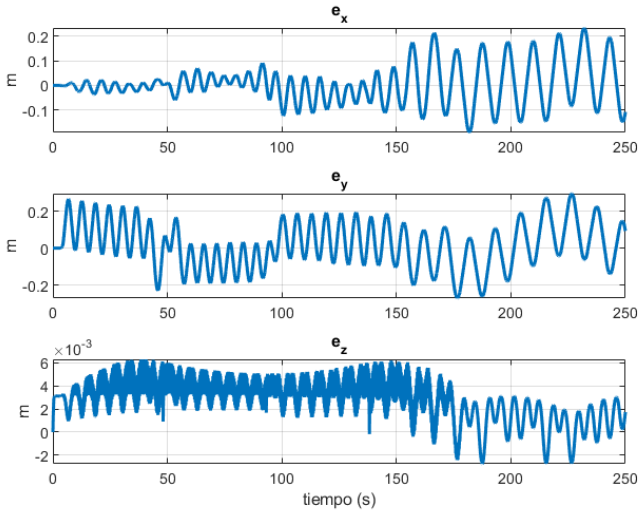


Figura 5.24 Errores cartesianos. PD de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

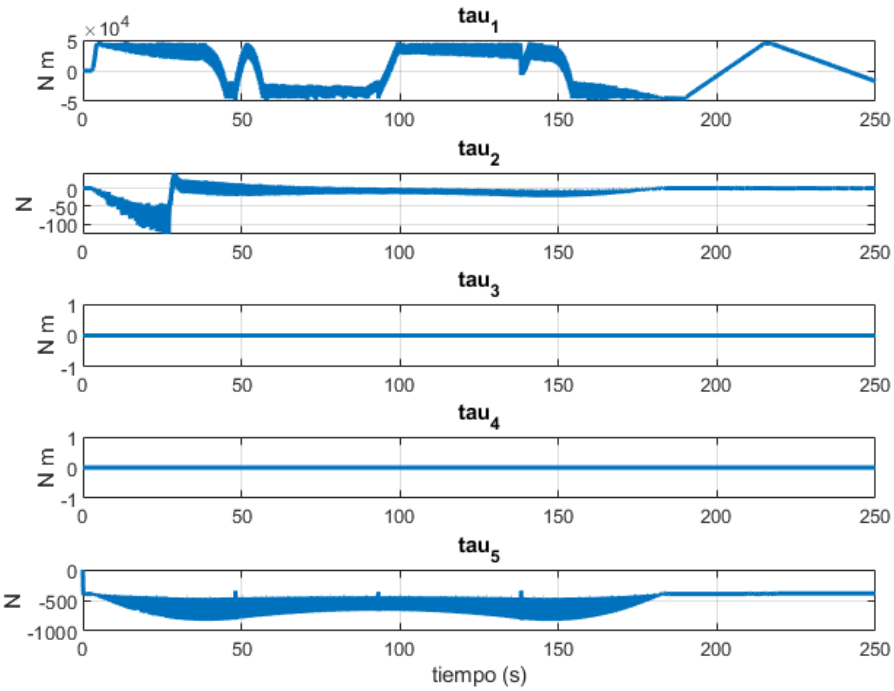


Figura 5.25 Señales de control. PD de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

Con respecto a q_2 y a q_5, no hay mucho cambio ni en los errores ni en las señales de control, lo que era esperable, ya que sus controladores no han sido modificados. En cambio, la señal de control de q_1 sí ha disminuido, tal y como se pretendía, además de ser más estable y no oscilar tanto como en los anteriores controladores. Esto ha mejorado principalmente los errores en velocidad y aceleración, tanto en magnitud como en forma, eliminando gran parte de las oscilaciones.
Indirectamente, los errores en velocidad y aceleración de las variables no actuadas se han reducido levemente; además, se ha atenuado la gran variación presente en dichos errores. Sin embargo, los errores cartesianos se han mantenido prácticamente iguales.

Controles PID

Habiendo comprobado el buen funcionamiento de los controladores PD; aunque, con el objetivo de mejorarlo, se plantea incluir el efecto integral en el control con vistas también a reducir el error cuando se incluyan perturbaciones.

La estructura de estos controladores en el dominio frecuencial es la siguiente:

$$C(s) = K_p + \frac{K_i}{s} + K_d \cdot s = K_p(1 + \frac{T_i}{s} + T_d \cdot s) = \frac{K_c(1 + c_1 \cdot s)(1 + c_2 \cdot s)}{s}$$

Mientras que, en el caso de estar basado en el tiempo, se expresa de esta forma:

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau) d\tau + K_d \cdot \frac{d\ e(t)}{dt}$$

El objetivo es el mismo que con los controladores PD, por lo que los tiempos de subida serán los mismos; se realizarán los mismos tres controladores.
En este caso, en el lugar de las raíces, es necesario incluir un integrador y dos ceros; por lo que, para la articulación prismática del carro, si se quiere un tiempo de subida en bucle cerrado ante escalón de 0,1 segundos, el lugar de las raíces queda conforme al diagrama que se encuentra inmediatamente debajo:

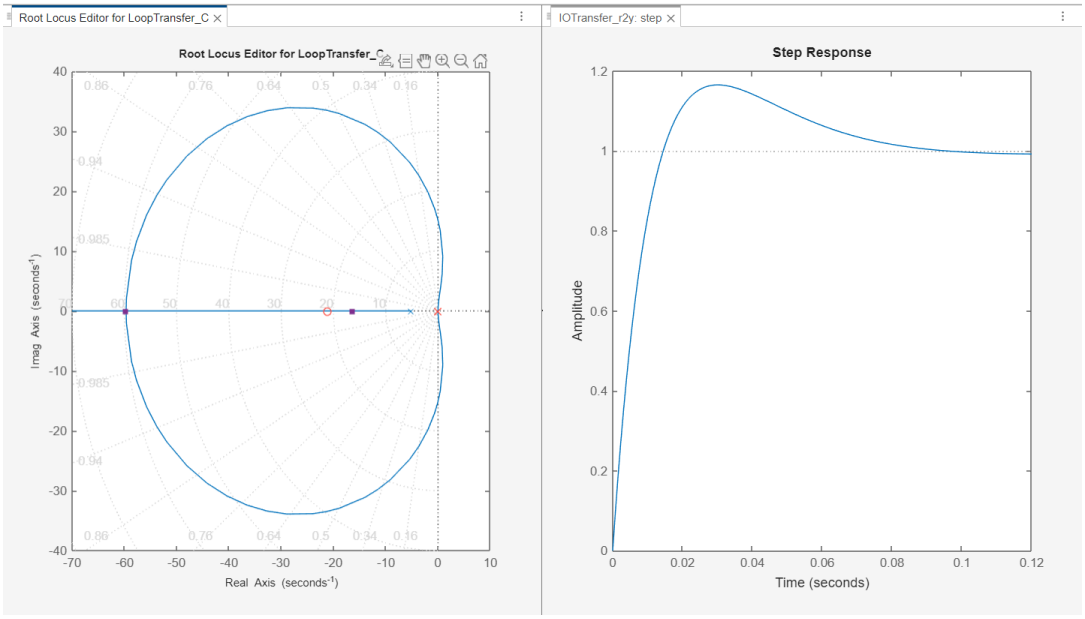


Figura 5.26 Lugar de las raíces de la función de transferencia de la articulación dos con dos ceros y un integrador en el controlador.

Se observa que hay tres polos en bucle cerrado, por lo que se han colocado dos rápidos iguales y reales al final de las dos ramas, mientras que es el otro el que domina a los demás por su lentitud, pero se aprecia que la respuesta se establece cerca de 0,1 segundos por lo que se van a mantener los parámetros que son:

- $K_{p2} = 2 \cdot 3980300 \cdot 0,047 = 374148,2$
- $K_{d2} = 3980300 \cdot 0,047^2 = 8792,4827$
- $K_{i2} = 3980300$

Para las demás articulaciones, se calculan los parámetros del controlador de forma análoga:

- $K_{p1} = 2 \cdot 494620000000 \cdot 0,05 = 49462000000$
- $K_{d1} = 494620000000 \cdot 0,05^2 = 1236600000$
- $K_{i1} = 494620000000$
- $K_{p5} = 2 \cdot 51995000 \cdot 0,047 = 4887530$
- $K_{d5} = 51995000 \cdot 0,047^2 = 114856,955$
- $K_{i5} = 51995000$

Los resultados obtenidos se muestran a continuación;

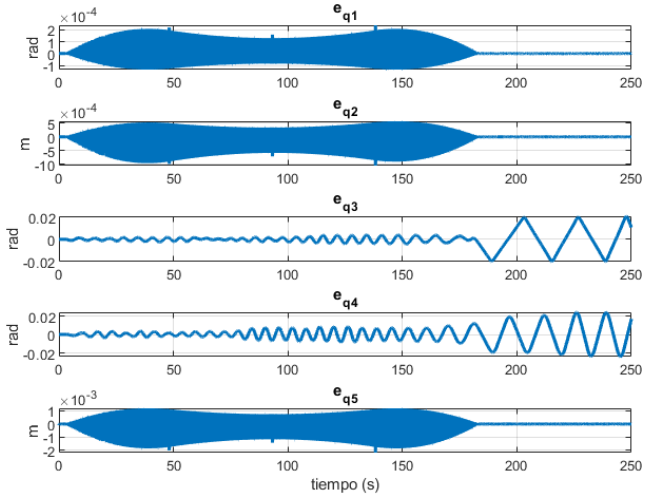


Figura 5.27 Errores en las articulaciones. PID sintonizado a 0,1 segundos.

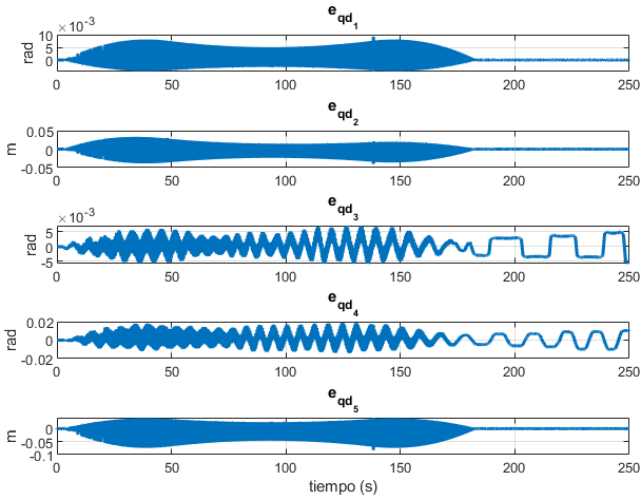


Figura 5.28 Errores en las velocidades articulaciones. PID sintonizado a 0,1 segundos.

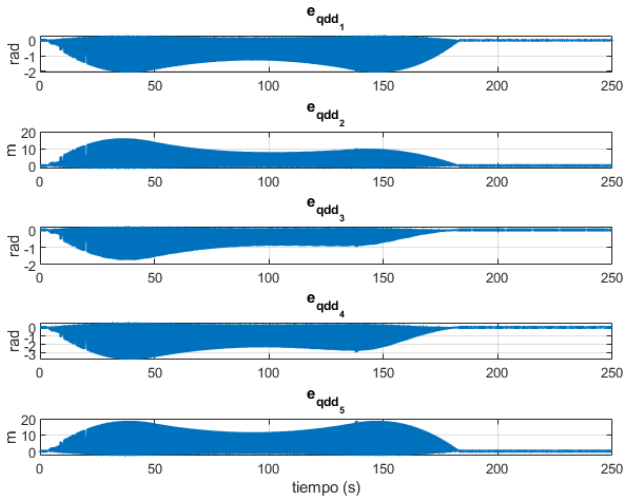


Figura 5.29 Errores en las aceleraciones articulaciones. PID sintonizado a 0,1 segundos.

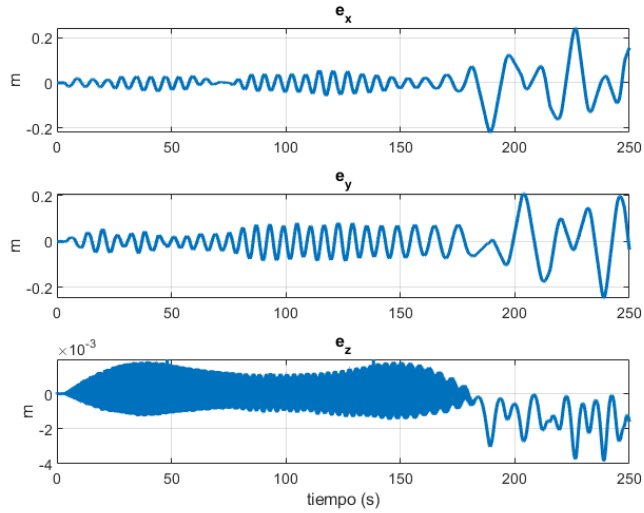


Figura 5.30 Errores cartesianos. PID sintonizado a 0,1 segundos.

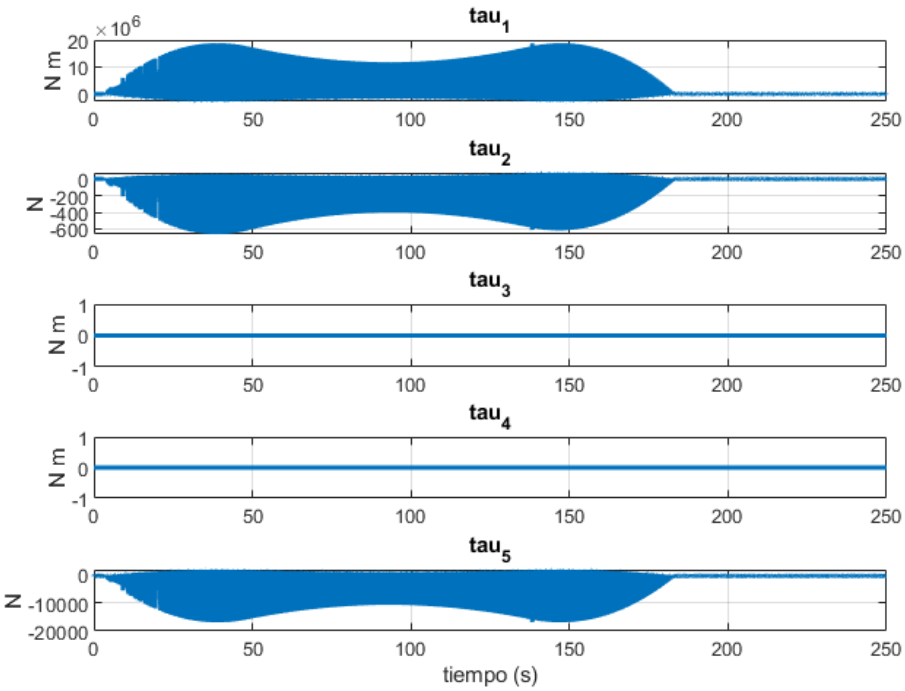


Figura 5.31 Señales de control. PID sintonizado a 0,1 segundos.

Los resultados son muy parecidos al PD sintonizado al mismo tiempo de subida, simplemente aumenta ligeramente la señal de control debido al efecto integral que mejora los errores en posición pero los aumenta en velocidad y posición. No se aprecia gran diferencia en el seguimiento entre ambos.

Durante la evaluación de los controladores PD, se observó que el desempeño del sistema mejoraba al incrementar el tiempo de subida. En particular, la configuración en la que q_1 se sintonizó con un tiempo de subida de 2 segundos, mientras que q_2 y q_5 se ajustaron a 0,5 segundos, logró reducir notablemente las oscilaciones en las señales de control, asumiendo a cambio un ligero aumento en el error. Con base en este resultado, se implementará esta misma estrategia empleando controladores PID. Por consiguiente, se descarta el diseño con un tiempo de subida uniforme de 0,5 segundos para todas las variables, al considerar que no aportaría ventajas significativas al estudio.

Los parámetros de dichos controladores son:

- $K_{p1} = 2 \cdot 61824000 \cdot 1 = 123648000$
- $K_{d1} = 61824000 \cdot 1^2 = 61824000$
- $K_{i1} = 61824000$
- $K_{p2} = 2 \cdot 51817 \cdot 0,18 = 18654,12$
- $K_{d2} = 51817 \cdot 0,18^2 = 1678,8708$
- $K_{i2} = 51817$
- $K_{p5} = 2 \cdot 632510 \cdot 0,19 = 240353,8$
- $K_{d5} = 632510 \cdot 0,19^2 = 22833,611$
- $K_{i5} = 632510$

Los resultados del PID con tiempos de subida en bucle cerrado distintos en las articulaciones son:

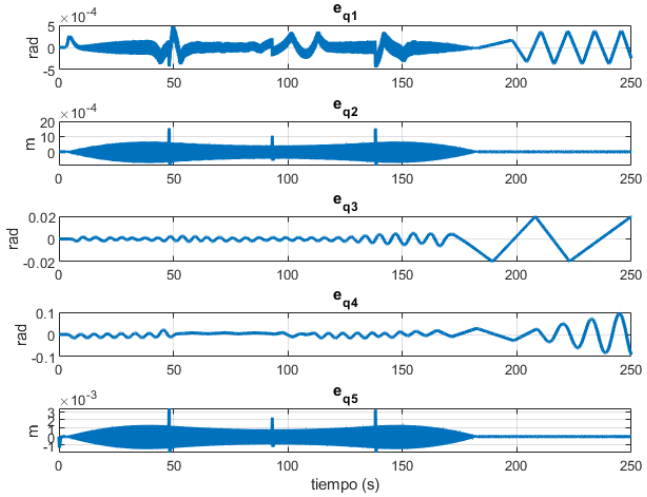


Figura 5.32 Errores en las articulaciones. PID de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

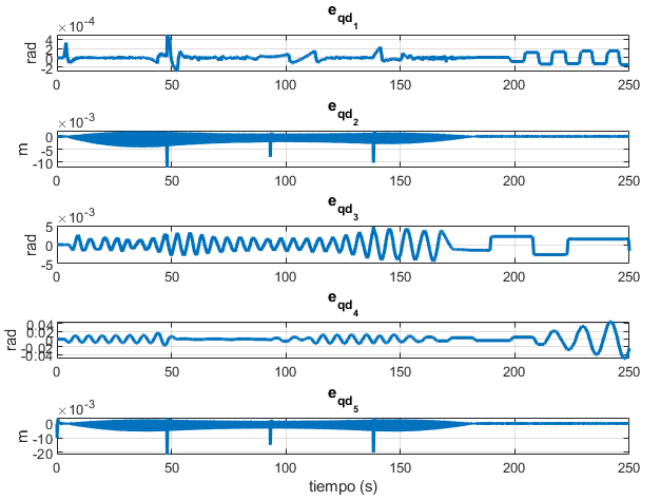


Figura 5.33 Errores en las velocidades articulaciones. PID de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

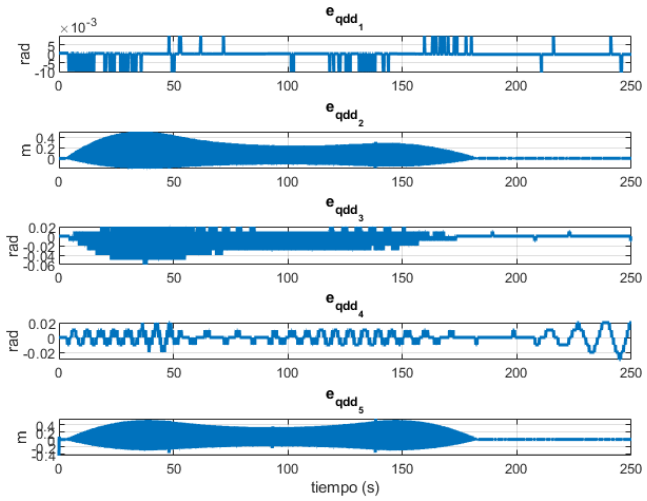


Figura 5.34 Errores en las aceleraciones articulaciones. PID de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

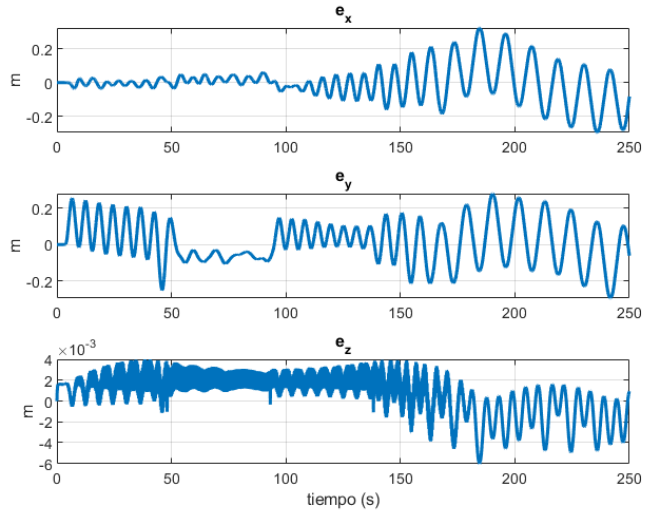


Figura 5.35 Errores cartesianos. PID de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

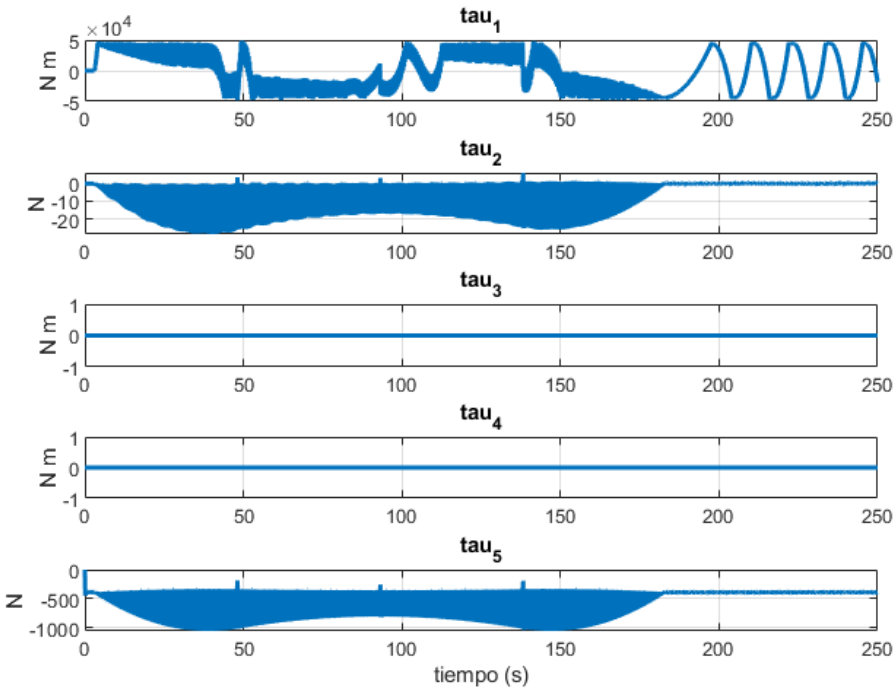


Figura 5.36 Señales de control. PID de q_1 sintonizado a 2 segundos y los demás a 0,5 segundos.

Se observa que la señal de control de q_2 ha disminuido considerablemente, mientras que q_1 ha mantenido sus niveles de magnitud en comparación con el PD sintonizado de la misma forma, y q_5 ha aumentado ligeramente su actuación.
Esto ha dado pie a que los errores de q_2 se reduzcan en posición, velocidad y aceleración. Los errores en velocidad y aceleración de q_5 han aumentado y se han reducido ligeramente los errores en posición de q_1.

De los controladores del tipo PD o PID, este último se considera uno de los mejores en seguimiento, ya que mejora ligeramente la rotación de la grúa y notablemente el desplazamiento del carrito con respecto a su versión en PD. Aunque se incremente ligeramente la señal de la elongación del cable y dichos errores se mantengan prácticamente iguales, la mejoría de las demás articulaciones hace que sea un buen controlador.
El criterio escogido es evaluar tanto los errores en posición, velocidad y aceleración en cada una de sus articulaciones como la señal de control, aunque todo sea de forma indirecta, ya que sólo se tiene control sobre la posición.

5.3.2 LQR y MPC: Control Multiarticular

El control multiarticular indica que, a la hora de generar una señal de control para una articulación, no se usa únicamente su estado, como en el control desacoplado, sino que se evalúan todas las articulaciones involucradas, y en función de ellas se calcula una señal de actuación concreta para actuar en consecuencia de lo que se desee.
Para ello, se hará uso del espacio de estados linealizado diseñado previamente y, en función de la estrategia a utilizar, se manejará de una forma u otra.

LQR: Regulador Cuadrático Lineal

El LQR resuelve un problema de control continuo óptimo; toda la teoría y desarrollo matemático se puede ver en la tesis doctoral de Francisco Gordillo Álvarez [18] en el apartado 2.4.
De forma resumida, se define una función de coste que involucra las desviaciones de los estados y la señal de control a aplicar sobre el sistema, y el objetivo se basa en calcular qué señal de control minimiza la función de coste completa para llevar los estados a un punto estable; es decir, desviación nula.

La función de coste es la siguiente:

$$\mathscr{J} = \int_{t_0}^{t_1} \left[x^T Q_c x + u^T R_c u\right] dt + x^T(t_1) S x(t_1)$$

Como se puede ver, la expresión está definida por dos matrices constantes y simétricas, Q y R, que representan los costes de desviación de cada estado y el esfuerzo de control, respectivamente, por lo que se definirán previamente. También pueden existir términos de control terminal (S), pero al hacer el horizonte infinito ($t_1 \to \infty$), como las desviaciones de los estados tienden a cero, este término también lo hace. Para resolver dicha función, se realiza la hamiltoniana a la que se le puede aplicar el Principio del Mínimo de Pontryagin, que reduce la expresión aplicando condiciones de optimalidad. Con todo ello, se llega a la denominada ecuación de Riccati, que, al resolverla, se determina la ley de control óptima para el sistema lineal en cuestión. La solución se puede agrupar en una matriz constante (K) que a la hora de implementar el controlador se debe hacer de la siguiente forma:

$$u = -K \cdot x$$

Realmente, no es necesario programar toda la parte matemática; Matlab dispone de una función llamada *lqr* que, si se pasan como argumentos las matrices del espacio de estados, devuelve directamente la matriz K resultante de todo el cálculo cambiada de signo.

Es necesario comprender que este sistema realmente no es lineal, por lo que no es una solución óptima, sólo para el punto de operación escogido en la linealización. Aún así, suele funcionar bastante bien en algunos casos cuyos sistemas también son no lineales.

Una vez que se ha comprendido cómo funciona internamente, es necesario definir las matrices Q y R. Normalmente no hay un método que indique cuáles deberían de ser los valores de la diagonal de dichas matrices. Los términos cruzados pueden eliminarse mediante un cambio de variables en la señal de control, siempre que la matriz R sea definida positiva, por lo que no son relevantes.
Se realizarán varias pruebas hasta llegar a un resultado aceptable. Se utilizará la matriz unidad como punto de partida para Q y R.

Para realizar el cálculo de K se usa el Código A.24 y queda de la siguiente forma:

$$K = \begin{bmatrix} 0,03526 & -0,02871 & 2,171 & 0,601 & -0,0856 & 21,67 & -0,007059 & 0,2181 & 5,688 & -0,02149 \\ -0,9909 & 2,051 & -578,0 & -153,6 & 12,73 & -745,1 & 0,6452 & -81,67 & -1534,0 & 3,309 \\ -0,1298 & 0,4394 & -23,82 & -24,72 & 3,139 & -128,5 & 0,8326 & 21,7 & -55,47 & 0,708 \end{bmatrix}$$

Los resultados claramente no son buenos, ya que no se ha sintonizado bien el controlador; sin embargo, aporta información para poder seguir modificando las matrices Q y R.

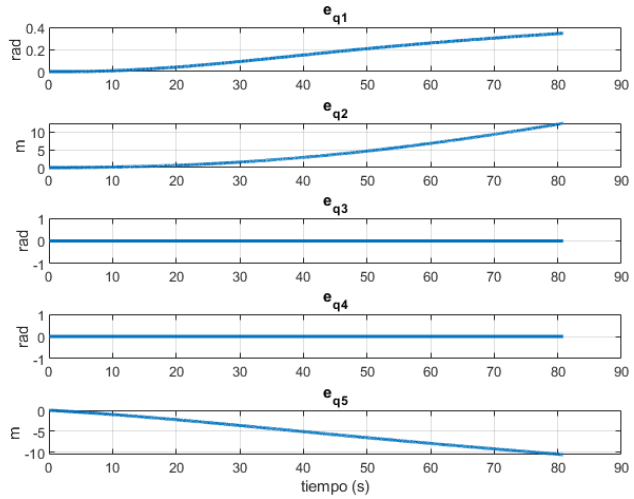


Figura 5.37 Errores en las articulaciones. LQR con Q y R siendo la identidad.

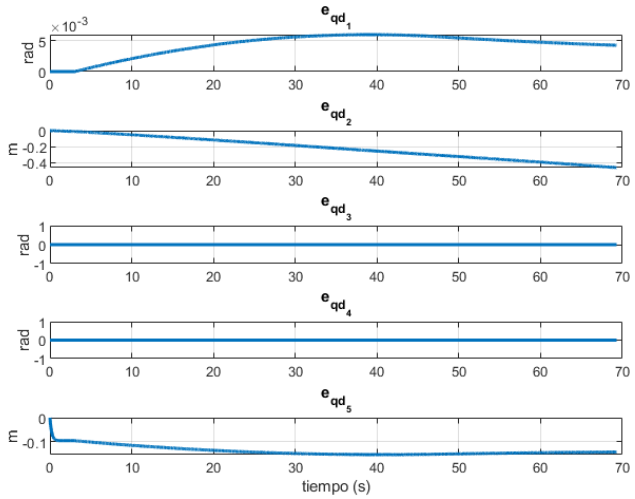


Figura 5.38 Errores en las velocidades articulaciones. LQR con Q y R siendo la identidad.

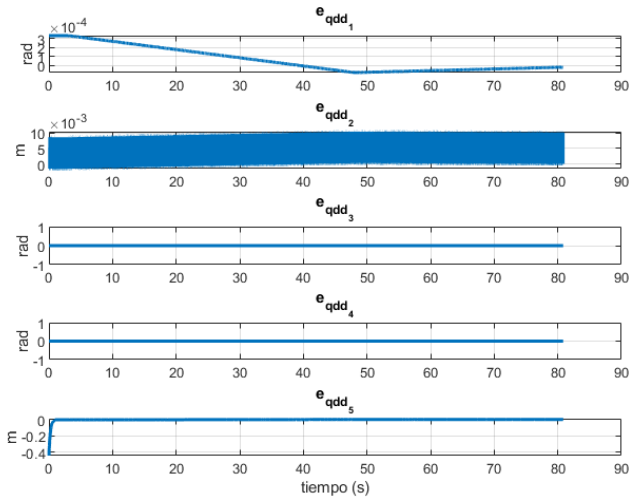


Figura 5.39 Errores en las aceleraciones articulaciones. LQR con Q y R siendo la identidad.

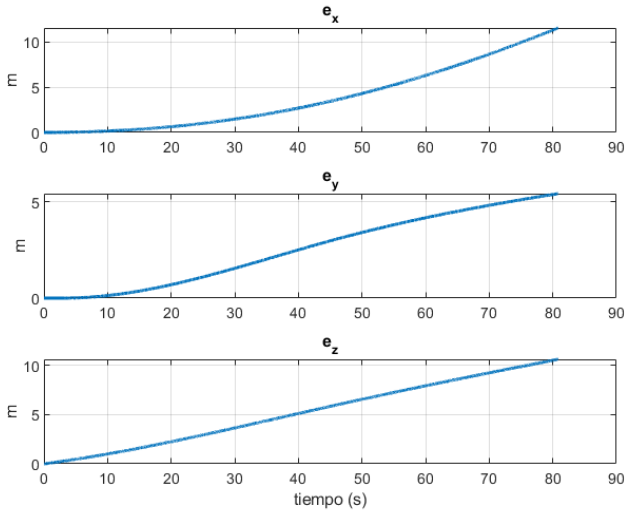


Figura 5.40 Errores cartesianos. LQR con Q y R siendo la identidad.

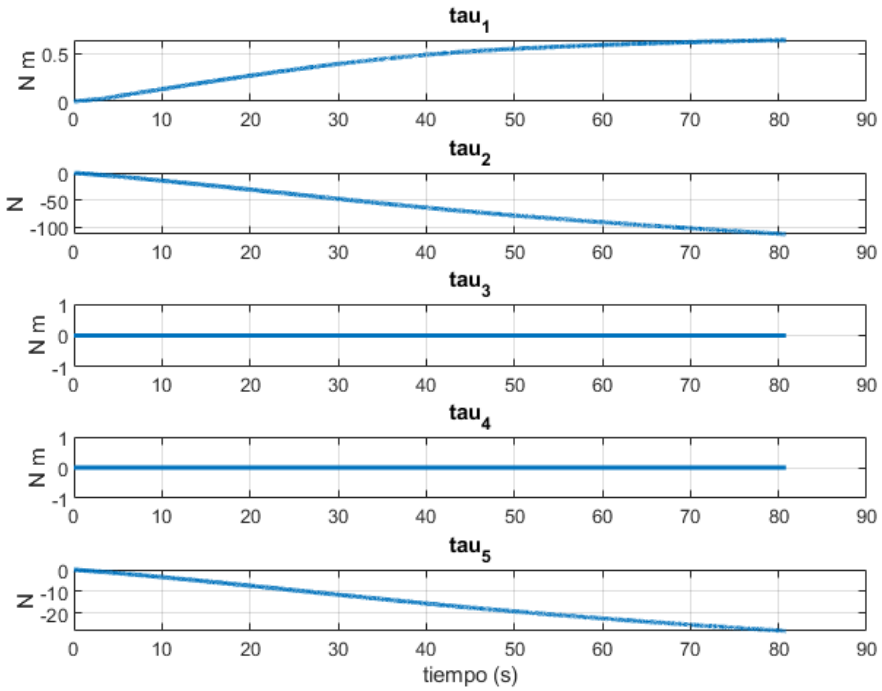


Figura 5.41 Señales de control. LQR con Q y R siendo la identidad.

Se observa que las señales de control no son especialmente dispares, tal y como se comentó al inicio de la sección. Sin embargo, se observa que q_1 prácticamente no se mueve y el control es tan malo que no logra controlar el sistema y lleva a alguna articulación fuera de sus límites, y por ello se detiene la simulación.

Para solucionar el problema de la descompensación, se utiliza una regla que suele funcionar bien a la hora de dar valores iniciales a las matrices llamada la Regla de Bryson. En muchos artículos de aplicación real se usa [19] y consiste en dar valores a la diagonal de Q y R siguiendo las siguientes expresiones:

$$Q_{ii} = \frac{1}{x^2_{mx_{ii}}}$$

$$R_{ii} = \frac{1}{u^2_{mx_{ii}}}$$

Sólo se va a emplear en R, ya que la medida de los incrementos de los estados no está descompensada y simplemente será necesario ir probando hasta encontrar unos valores adecuados de forma empírica. Suponiendo que los valores máximos de las señales de control son 5000000 $N \cdot m$, 50 N y 500 N para q_1, q_2 y q_5 respectivamente, la diagonal de R adquiere los valores que se muestran a continuación:

$$\begin{bmatrix} 4.0 \times 10^{-14} & 0 & 0 \\ 0 & 0.0004 & 0 \\ 0 & 0 & 4.0 \times 10^{-6} \end{bmatrix}$$

Para Q, después de varias pruebas, se ha escogido que sea:

$$\begin{bmatrix} 700 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 700 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 700 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1000 \end{bmatrix}$$

La matriz K resultante de la función *lqr* de matlab es la siguiente:

$$K = \begin{bmatrix} 1.299 \times 10^8 & 2.333 \times 10^7 & -6.543 \times 10^7 & 7.854 \times 10^7 & 9.188 \times 10^6 & 3.491 \times 10^8 & 2.76 \times 10^7 & -1.439 \times 10^8 & -1.132 \times 10^8 & 1.186 \times 10^7 \\ -236.3 & 1301.0 & 239.6 & -233.7 & 42.3 & -653.6 & 1283.0 & 767.1 & 244.1 & 53.09 \\ -806.1 & -552.2 & -276.6 & -833.3 & 13200.0 & -2380.0 & -610.1 & 2616.0 & 912.7 & 12910.0 \end{bmatrix}$$

Al implementar dicha matriz los resultados obtenidos son:

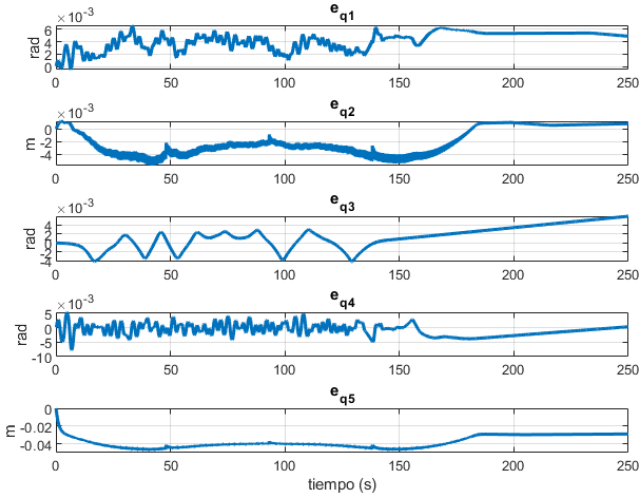


Figura 5.42 Errores en las articulaciones. LQR con Q empírica y R según Bryson.

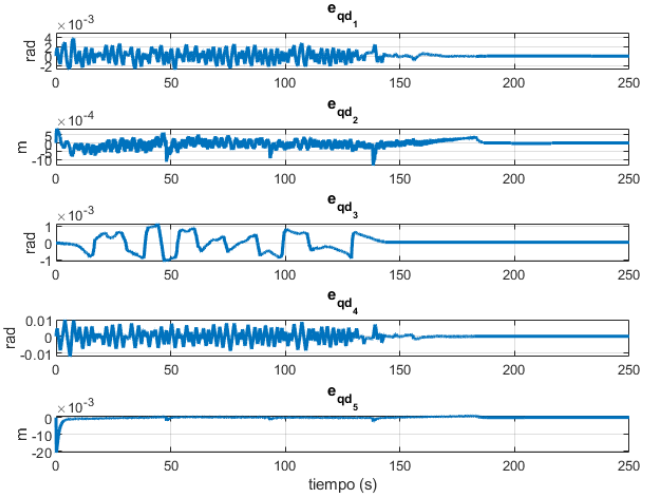


Figura 5.43 Errores en las velocidades articulaciones. LQR con Q empírica y R según Bryson.

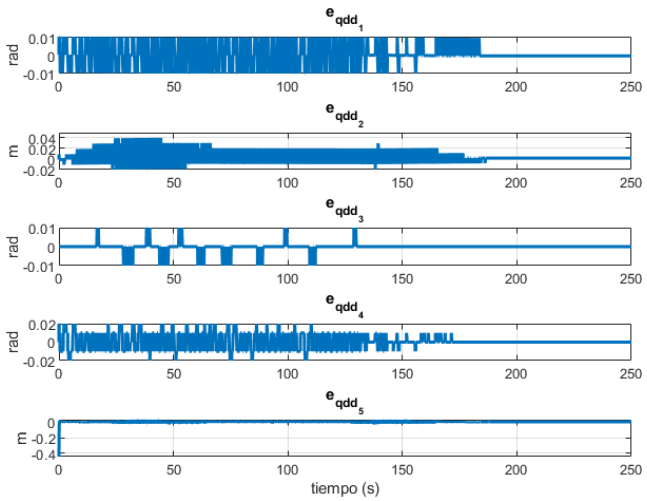


Figura 5.44 Errores en las aceleraciones articulaciones. LQR con Q empírica y R según Bryson.

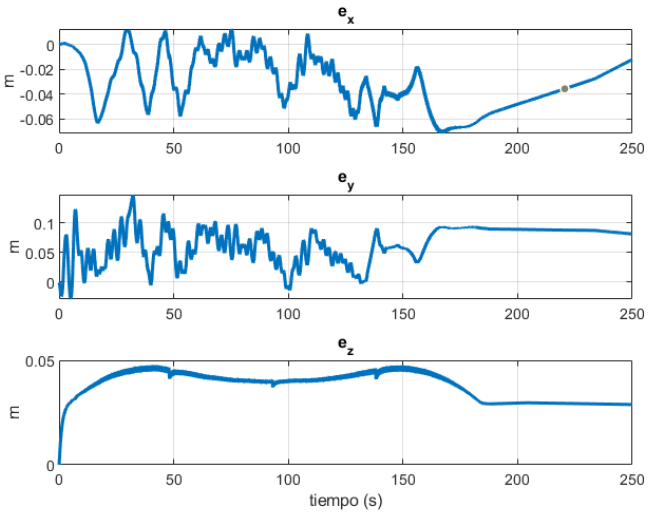


Figura 5.45 Errores cartesianos. LQR con Q empírica y R según Bryson.

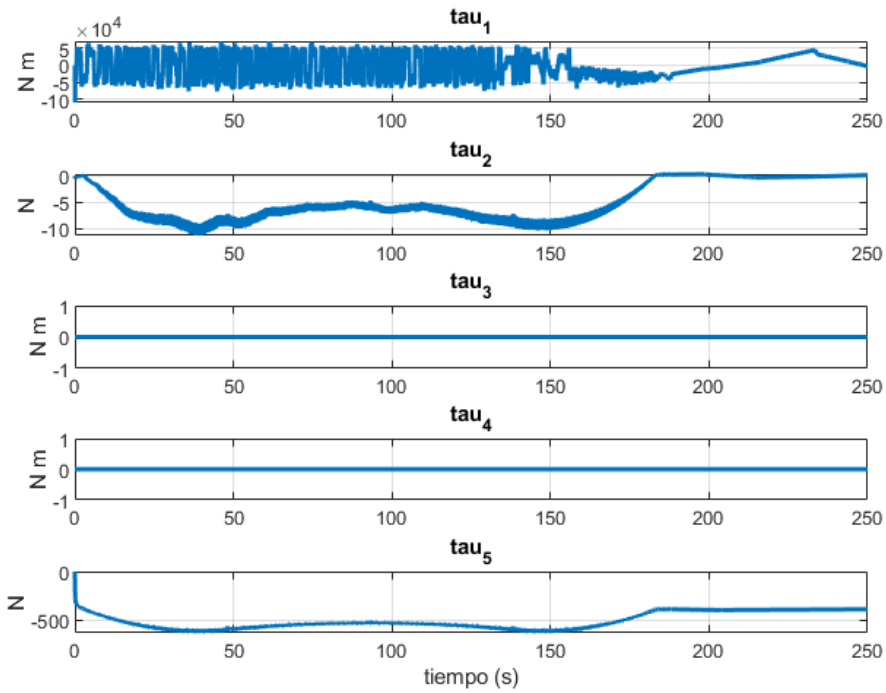


Figura 5.46 Señales de control. LQR con Q empírica y R según Bryson.

Si se observan las señales de control, las oscilaciones de q_1 aumentan significativamente; sin embargo, no lo hace su magnitud. En cuanto a q_2 y q_5, disminuyen las oscilaciones y también la magnitud aproximadamente a la mitad , por lo que se mejora notablemente en comparación con cualquier otro controlador.
En cambio, esto no degrada la calidad del controlador; todos los errores son, por lo general, muy pequeños.

Por último, se usará la matriz R para reducir dichas oscilaciones en la señal de control de q_1. Se multiplicará por cien y, además, se reducirá la señal máxima de q_1:

$$R = \begin{bmatrix} 4.0 \times 10^{-10} & 0 & 0 \\ 0 & 0.04 & 0 \\ 0 & 0 & 4.0 \times 10^{-4} \end{bmatrix}$$

Con este cambio, la ganancia del LQR queda de la siguiente manera:

$$K = \begin{bmatrix} 1.3208 \times 10^{6} & -7.4493 \times 10^{4} & 1.4186 \times 10^{5} & 1.2930 \times 10^{6} & -1.2941 \times 10^{4} & 6.3114 \times 10^{6} & -1.0265 \times 10^{5} & -3.4820 \times 10^{6} & 5.2290 \times 10^{5} & 2.6494 \times 10^{3} \\ 7.4459 \times 10^{0} & 1.3208 \times 10^{2} & 7.3010 \times 10^{1} & -1.8513 \times 10^{1} & 2.7462 \times 10^{0} & 4.8111 \times 10^{1} & 5.2249 \times 10^{1} & 2.6387 \times 10^{2} & -3.0799 \times 10^{1} & 4.4878 \times 10^{0} \\ 3.2680 \times 10^{0} & 1.2742 \times 10^{0} & -1.5982 \times 10^{2} & -5.6226 \times 10^{1} & 1.3235 \times 10^{3} & -3.2822 \times 10^{0} & -1.2309 \times 10^{1} & 6.0693 \times 10^{1} & -3.0799 \times 10^{1} & 4.4878 \times 10^{0} \end{bmatrix}$$

Y los resultados obtenidos son:

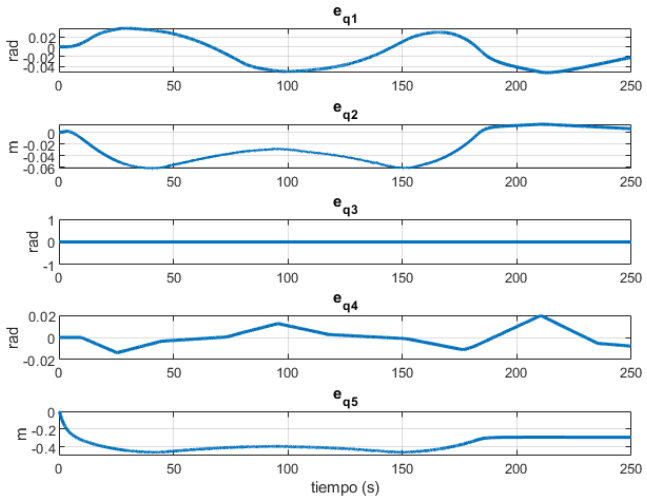


Figura 5.47 Errores en las articulaciones. LQR con cambio en R.

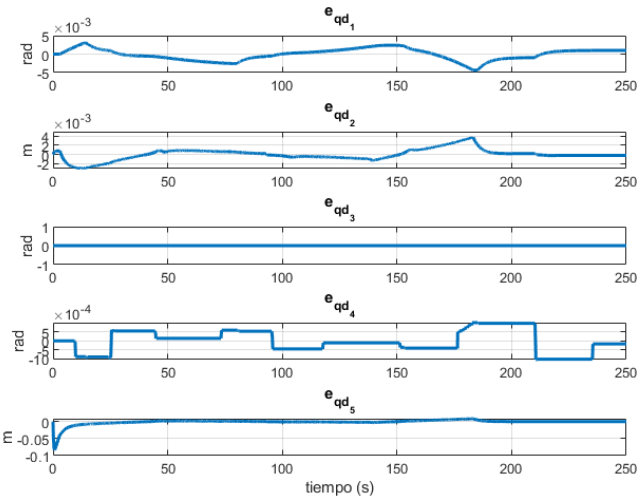


Figura 5.48 Errores en las velocidades articulaciones. LQR con cambio en R.

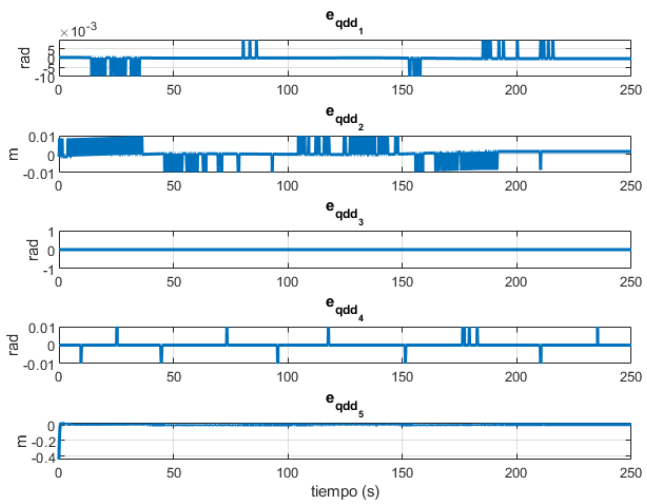


Figura 5.49 Errores en las aceleraciones articulaciones. LQR con cambio en R.

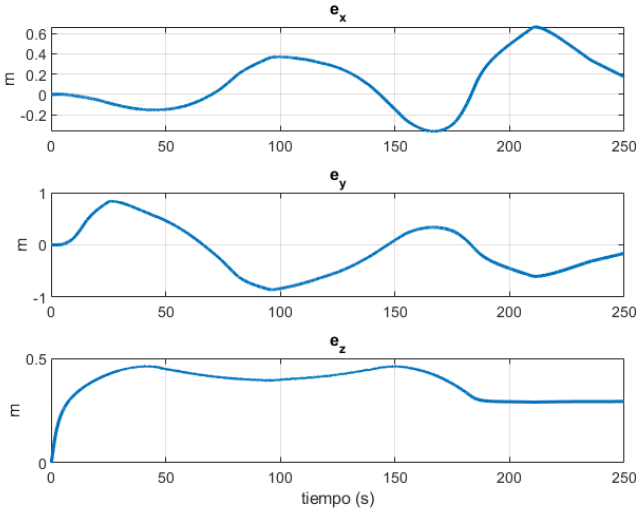


Figura 5.50 Errores cartesianos. LQR con cambio en R.

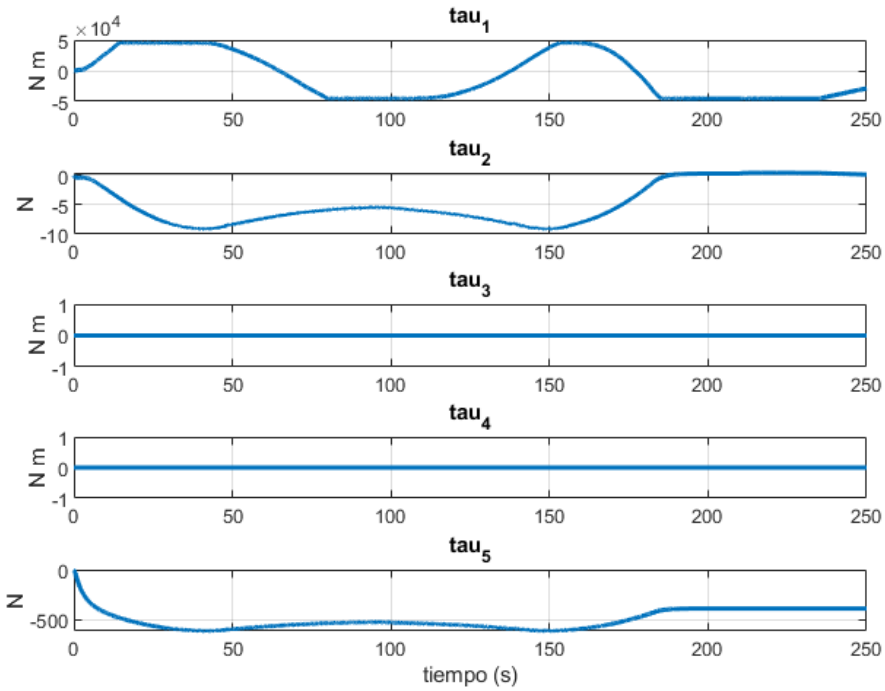


Figura 5.51 Señales de control. LQR con cambio en R.

Se observa que, en general, todos los errores y señales de control se han suavizado; y, por ende, se han eliminado las oscilaciones debido a la agresividad del control. Esto provoca que el orden de los errores aumente ligeramente y sea algo más lento, aunque es relativamente bueno, considerando las dimensiones de la grúa.

MPC: Control Predictivo basado en Modelo

En principio, este tipo de control, que puede ser multivariable, es utilizado en plantas químicas y no tanto en robótica; sin embargo, debido a las variables articulares subactuadas, es posible que los resultados sean buenos. Sobre todo, gracias a la predicción del horizonte de control y a que soporta sistemas no lineales. Más aún, este horizonte finito permite la incorporación de restricciones al control, lo que es de gran ayuda en sistemas reales, principalmente por motivos de seguridad.

La estrategia es parecida a la del LQR; se utilizan una serie de métodos de control que hacen uso del modelo dinámico del sistema para minimizar una función objetivo. Se hace uso del denominado horizonte de control, el cuál determina las N salidas consecutivas al instante para el que se calcula la señal de control, aunque sólo se aplique este primero. Tanto cuando este horizonte como el de predicción tienden a infinito, el problema se convierte en el problema de LQR ya visto previamente.
Todas las ventajas que conlleva el uso de dicha estrategia, así como las matemáticas asociadas a los diferentes métodos y su comportamiento en términos de robustez, se encuentran en un artículo publicado en la Revista iberoamericana de automática e informática industrial (RIAI) [20]. De él se ha extraído la información básica para comprender las bases del problema y la capacidad real.

A la hora de implementarlo, Matlab, al igual que con el LQR, dispone de un bloque al cual se le especifica un objeto que contiene la información necesaria para realizar los cálculos. Matlab realmente implementa este control en tiempo discreto, pero la transformación la realiza de forma interna. El objeto necesario para el bloque debe programarse y se le especifica el sistema en espacio de estados, el horizonte de control y predicción, el tiempo de discretización, las restricciones tanto de las señales de control como de las salidas, Q, R y otra matriz que penaliza la brusquedad (las variaciones) de las señales de control. El código empleado para ello es el Código A.25.

La primera prueba a realizar utilizará la misma Q y R que la última sintonización del LQR y se comprobará si sirve como buena aproximación inicial. La configuración completa será:

- Horizonte de predicción: 100 pasos (3 segundos).
- Horizonte de control: 4 pasos (0,12 segundos).
- Tiempo de discretización: 0,03 segundos (20 veces más pequeño que la dinámica más rápida).
- Restricciones sobre las variables manipuladas:
  - Motor de giro: $[-5 \times 10^6,\ 5 \times 10^6]$ Nm.
  - Motor del carro: $[-50,\ 50]$ N.
  - Motor de elevación: $[-500,\ 500]$ N.
- Restricciones sobre las salidas:
  - Posición del carrito: $[-12,\ 14]$.
  - Longitud del cable: $[-14,\ 24]$.
  - Ángulos del péndulo: $[-\frac{\pi}{3},\ \frac{\pi}{3}]$.
- Matriz de pesos de las salidas (Q), considerada diagonal:

$$Q = \text{diag}(700,\ 700,\ 50,\ 50,\ 700,\ 1000,\ 1000,\ 1000,\ 1000,\ 1000)$$

- Matriz de pesos de las variables manipuladas (R), considerada diagonal:

$$R = \text{diag}\left(\frac{1}{(5 \times 10^6)^2},\ \frac{1}{50^2},\ \frac{1}{500^2}\right)$$

- Matriz de pesos sobre la variación de la señal de control (Δu), considerada diagonal:

$$R_\Delta = \text{diag}(1,\ 1,\ 1)$$

El resultado de la simulación con dichos datos es:

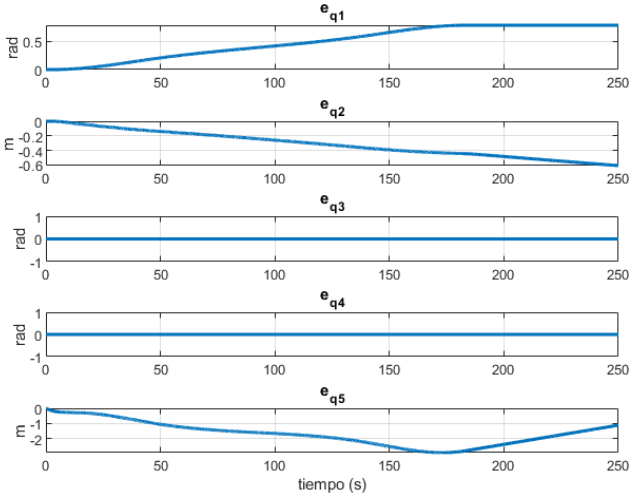


Figura 5.52 Errores en las articulaciones. MPC con Q y R como el LQR rápido.

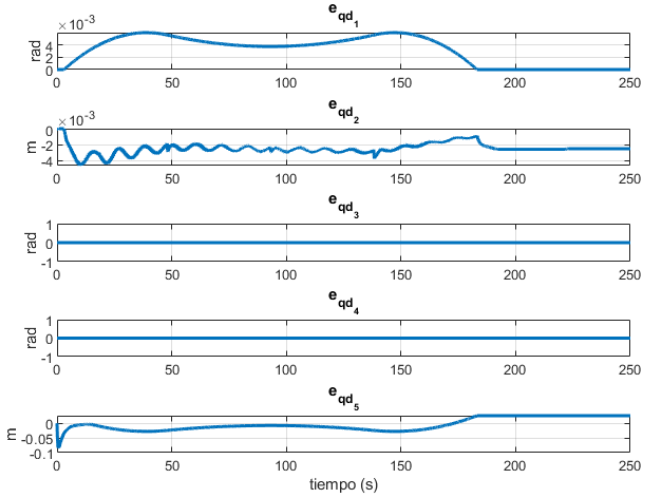


Figura 5.53 Errores en las velocidades articulaciones. MPC con Q y R como el LQR rápido.

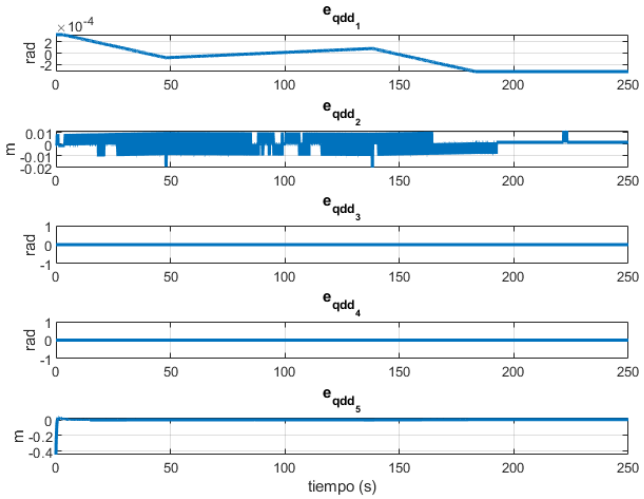


Figura 5.54 Errores en las aceleraciones articulaciones. MPC con Q y R como el LQR rápido.

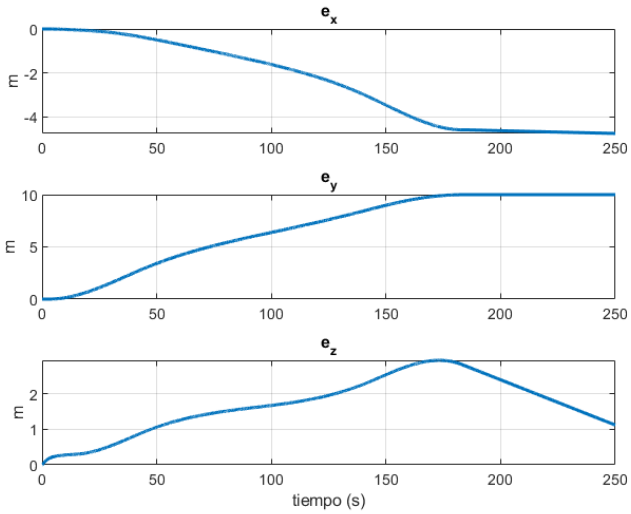


Figura 5.55 Errores cartesianos. MPC con Q y R como el LQR rápido.

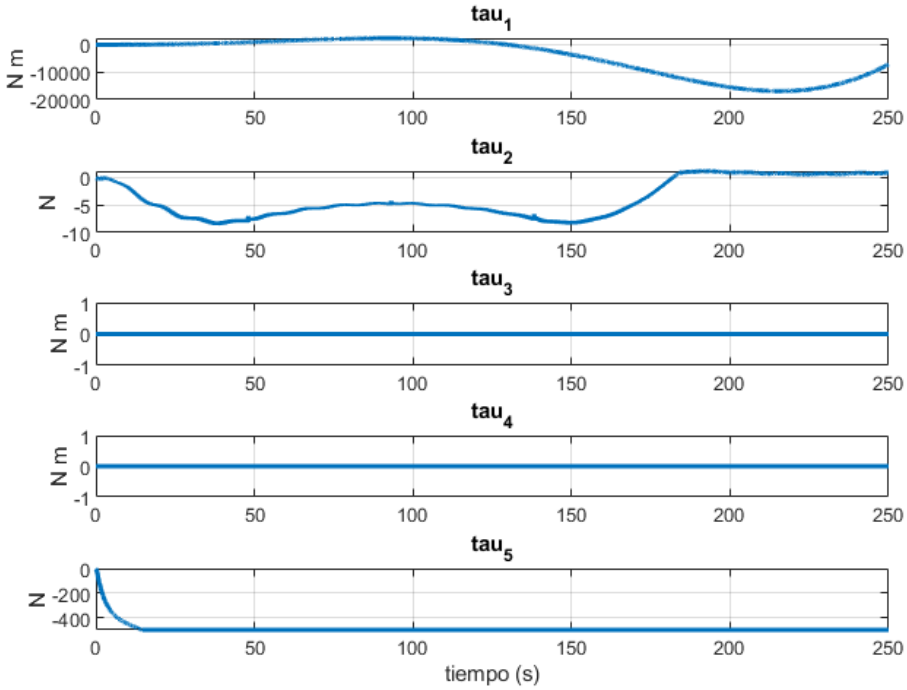


Figura 5.56 Señales de control. MPC con Q y R como el LQR rápido.

Como se puede ver, no es bueno; de hecho, la señal de control de q_1 nunca llega al mínimo y no se mueve. Esto se aprecia al observar directamente q_1, no en el error:

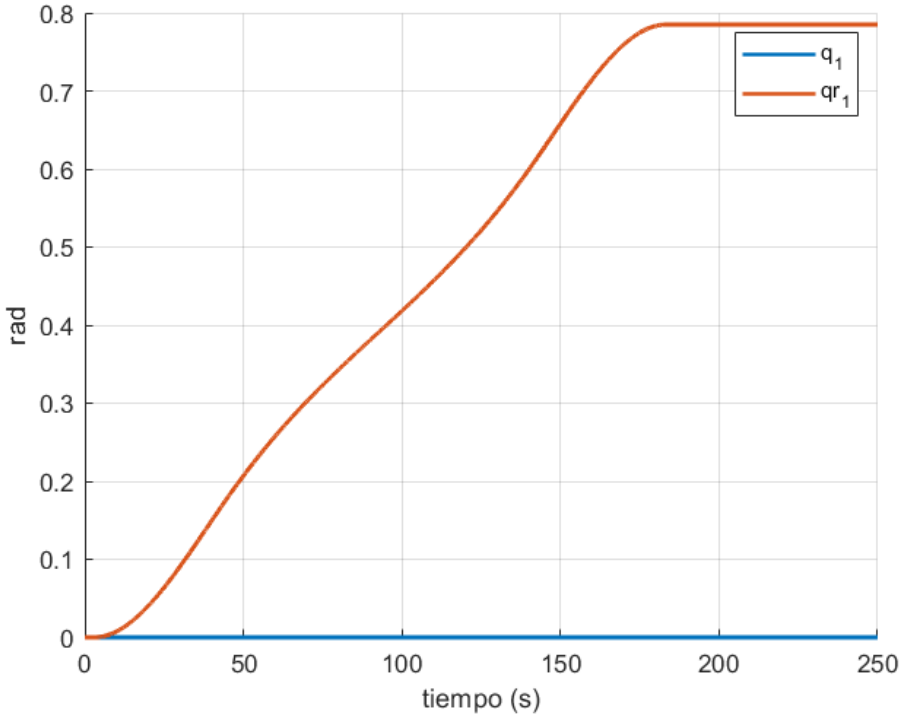


Figura 5.57 Comparación de q_1 con su referencia. MPC con Q y R como el LQR rápido.

Aunque en algún momento la señal llegara al mínimo, se movería pero con mucho retraso. Si se moviese en algún momento podría extenderse el horizonte de predicción para que el MPC pudiese contrarrestarlo, sin embargo esto hace las simulaciones más largas debido a que aumenta el cómputo y además no es la forma de solucionar este fallo.
Para evitar esto y no seguir sobredimensionando los parámetros utilizados, se realizará un escalado que incorpora matlab al objeto del MPC. De esta forma, se intenta que los valores de Q y R, o de los horizontes, se centren en la corrección de errores y no fallen por grandes diferencias entre las señales de control de las distintas articulaciones, tal y como ocurría en el LQR antes de usar Bryson.
Tras realizar una serie de pruebas, se llega a la siguiente configuración:

- Horizonte de predicción: 100 pasos (3 segundos).
- Horizonte de control: 4 pasos (0,12 segundos).
- Tiempo de discretización: 0,03 segundos.
- Restricciones sobre las variables manipuladas:
  - Motor de giro: $[-5 \times 10^6,\ 5 \times 10^6]$ Nm.
  - Motor del carro: $[-500,\ 500]$ N.
  - Motor de elevación: $[-1000,\ 1000]$ N.
- Valores de escalado de las variables manipuladas:
  - Motor de giro: 5×10^6
  - Motor del carro: 500
  - Motor de elevación: 1000
- Restricciones sobre las salidas:
  - Posición del carrito: $[-12,\ 14]$.
  - Longitud del cable: $[-14,\ 24]$.
  - Ángulos del péndulo: $[-\frac{\pi}{3},\ \frac{\pi}{3}]$.

- Matriz de pesos de las salidas (Q), considerada diagonal:

$$Q = \text{diag}(3000,\ 3000,\ 500,\ 500,\ 3000,\ 1000,\ 1000,\ 2000,\ 2000,\ 1000)$$

- Matriz de pesos de las variables manipuladas (R), considerada diagonal:

$$R = \text{diag}\,(0,1\ ,\ 0,1\ ,\ 0,1)$$

- Matriz de pesos sobre la variación de la señal de control (Δu), considerada diagonal:

$$R_\Delta = \text{diag}(150,\ 150,\ 150)$$

Los resultados son:

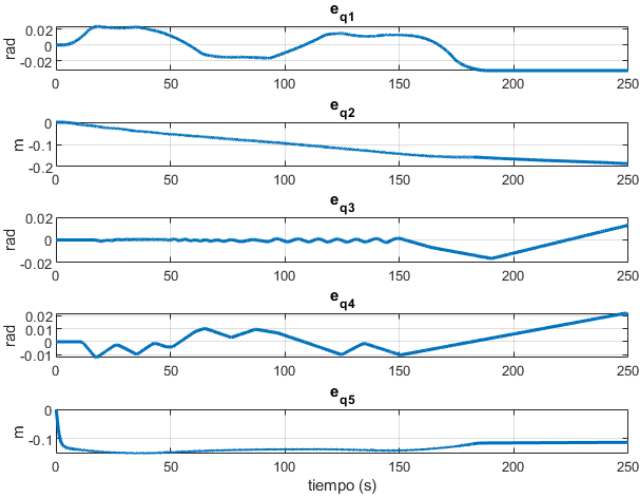


Figura 5.58 Errores en las articulaciones. MPC escalado.

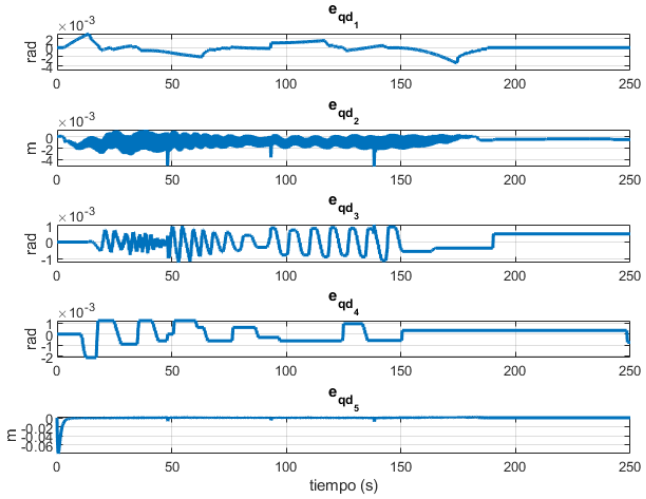


Figura 5.59 Errores en las velocidades articulaciones. MPC escalado.

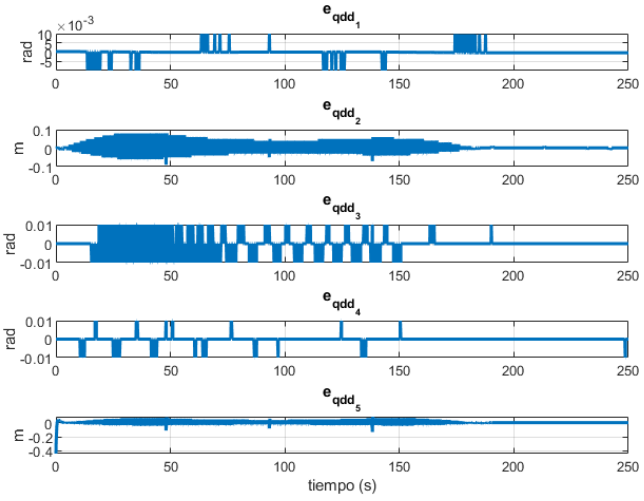


Figura 5.60 Errores en las aceleraciones articulaciones. MPC escalado.

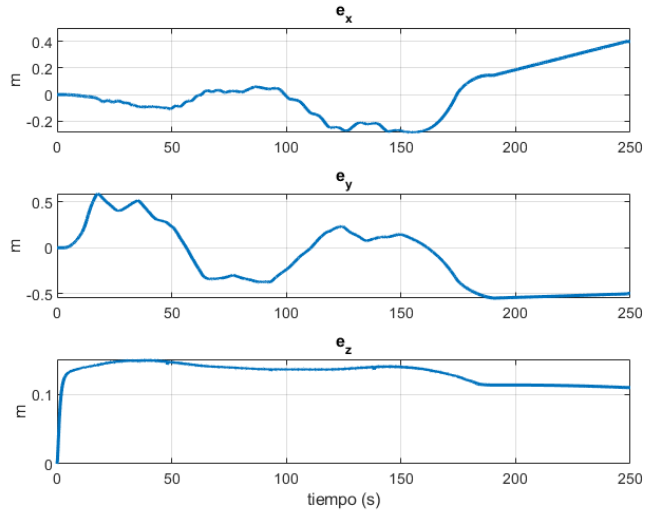


Figura 5.61 Errores cartesianos. MPC escalado.

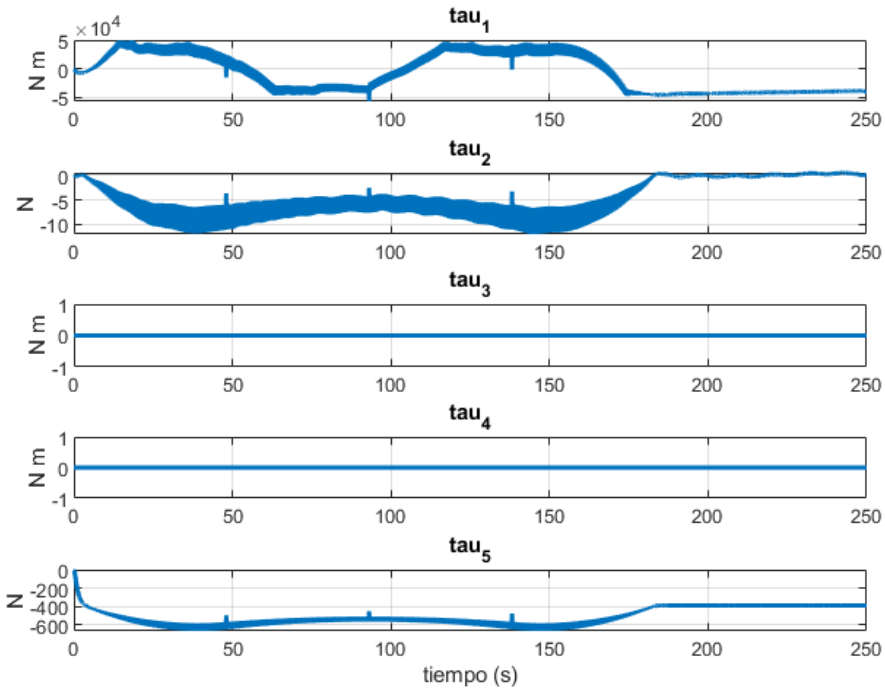


Figura 5.62 Señales de control. MPC escalado.

Los resultados son buenos, aunque se mantienen ciertos errores que parecen ser constantes. Después de muchas pruebas, se llega a la conclusión de que esto se debe a la diferencia entre el modelo real y el que se usa en el espacio de estados. Como el MPC utiliza el espacio de estados para predecir las reacciones futuras, al no ser exactamente igual al sistema, da lugar a errores que no es capaz de corregir. Aún así, mantiene el orden de las señales de control sin muchas oscilaciones con respecto a otros controles anteriores, con errores ligeramente mayores que en otros controles en general.

Si se quiere probar con otro tiempo de muestreo para comprobar si el control puede mejorar haciéndolo más lento y, de esta forma, intentar disminuir aún más las señales de control, se prueba con un tiempo de muestreo de 0,5 segundos. La sintonización completa es:

- Horizonte de predicción: 10 pasos (5 segundos).
- Horizonte de control: 3 pasos (1,5 segundos).
- Tiempo de discretización: 0,5 segundos.
- Restricciones sobre las variables manipuladas:
  - Motor de giro: $[-5 \times 10^6,\ 5 \times 10^6]$ Nm.
  - Motor del carro: $[-500,\ 500]$ N.
  - Motor de elevación: $[-1000,\ 1000]$ N.
- Valores de escalado de las variables manipuladas:
  - Motor de giro: 5×10^6
  - Motor del carro: 500
  - Motor de elevación: 1000
- Restricciones sobre las salidas:
  - Posición del carrito: $[-12,\ 14]$.
  - Longitud del cable: $[-14,\ 24]$.

- Ángulos del péndulo: $[-\frac{\pi}{3}, \frac{\pi}{3}]$.

- Matriz de pesos de las salidas (Q), considerada diagonal:

$$Q = \text{diag}(3000,\ 3000,\ 500,\ 500,\ 3000,\ 1000,\ 1000,\ 2000,\ 2000,\ 1000)$$

- Matriz de pesos de las variables manipuladas (R), considerada diagonal:

$$R = \text{diag}\,(100\ ,\ 100\ ,\ 100)$$

- Matriz de pesos sobre la variación de la señal de control (Δu), considerada diagonal:

$$R_\Delta = \text{diag}(1,\ 1,\ 1)$$

Para este caso, los errores y señales de control son:

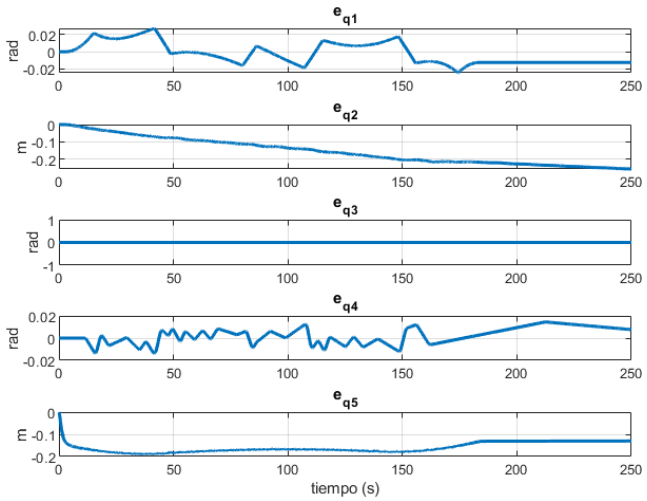


Figura 5.63 Errores en las articulaciones. MPC lento.

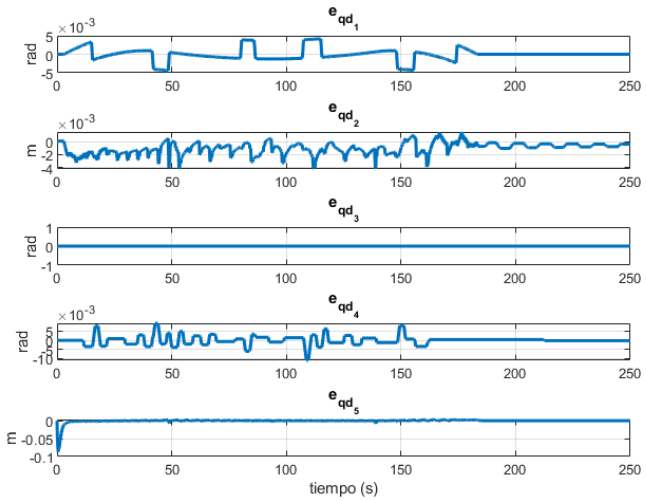


Figura 5.64 Errores en las velocidades articulaciones. MPC lento.

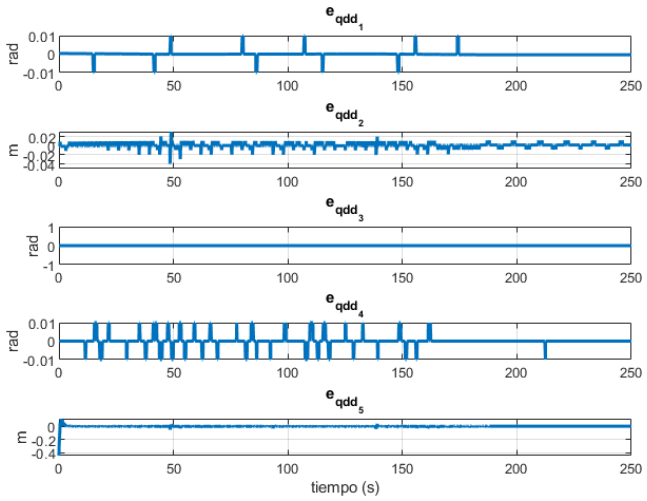


Figura 5.65 Errores en las aceleraciones articulaciones. MPC lento.

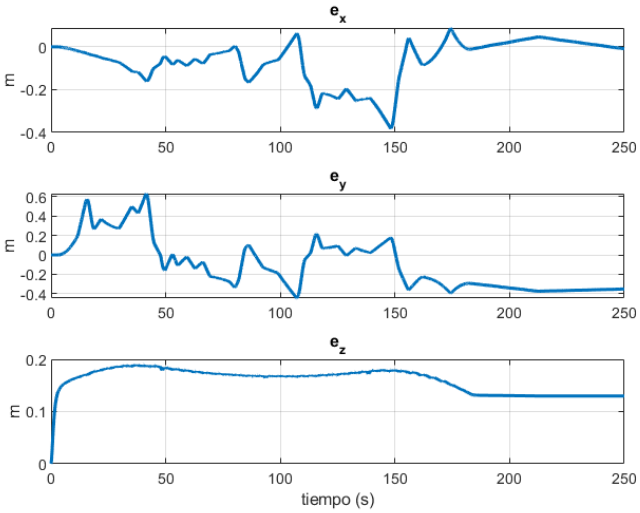


Figura 5.66 Errores cartesianos. MPC lento.

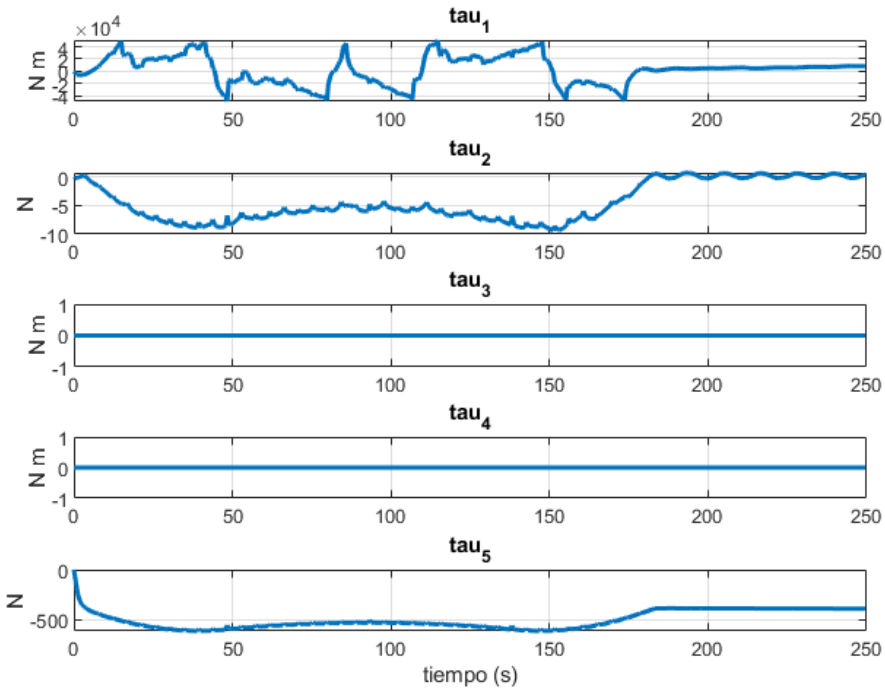


Figura 5.67 Señales de control. MPC lento.

Se eliminan por completo las oscilaciones, y aunque el control sea algo más lento, al igual que el LQR ralentizado, es bastante bueno, incluso reduciendo los errores del LQR y manteniendo los del MPC anterior.

5.4 Rechazo de perturbaciones

A diferencia de la expresión en bucle cerrado del seguimiento de referencias, la de las perturbaciones según el diagrama de control de la Figura 5.8 es la siguiente:

$$G_{BC}^{pert}(s) = \frac{y(s)}{p(s)} = \frac{G(s)}{1 + C(s)G(s)H(s)}$$

En el rechazo de perturbaciones, el sistema en sí mismo no logra corregir el error debido a que no se tienen en cuenta sus polos, sino los del controlador. Además, se combinan ambas dinámicas; por lo que, si se mantienen las perturbaciones en el tiempo, pueden inestabilizar el sistema si no tienen un buen control.
En cambio, es posible que las perturbaciones típicas que puede tener una grúa, como la fuerza del viento, no afecten prácticamente a las articulaciones actuadas, pero sí a las ondulaciones del péndulo.

Para saber exactamente cómo modelar la fuerza del viento, es necesario conocer su dinámica y saber cómo afecta a la grúa en cada uno de sus eslabones. La norma europea UNE-EN 13001-2, cuyo nombre es *Seguridad de las grúas. Requisitos generales de diseño. Parte 2: Acciones de la carga*, aborda este tema con todo detalle; sin embargo, dado que el acceso es restringido, para este trabajo de simulación, la perturbación del viento se va a modelar usando alguna referencia que haga uso de alguna norma parecida o equivalente.
En una tesis de la Universitat Jaume I [21], aunque el proyecto trate sobre una grúa pórtico, se hace uso de la norma UNE 58113:1985 *Grúas. Acción del viento.*, de donde se parte de la ecuación por la que se rige la fuerza del viento sobre un eslabón:

$$F = A \cdot p \cdot C_f$$

donde

- F es la fuerza debida al viento en kN

- A es la superficie neta del elemento considerado en m^2
- p es la presión del viento en $\frac{kN}{m^2}$
- C_f es el coeficiente de forma del elemento considerado

El desarrollo se encuentra en el punto 8.4.1.1 del documento y, además, incluye una serie de tablas para poder conocer de forma estimada la velocidad y presión del viento sobre el sistema en función del tipo de grúa que sea. Al decidir que este sistema a tratar se incluya dentro de todos los tipos normales de grúa que se instalen al aire libre, la velocidad del viento con la que se trabajará es de 20 m/s y la presión del viento será de 0,25 kPa/m^2. Además, esta velocidad, que es en servicio (movimiento), concuerda con el apartado 5.3.4. del Proyecto de Fin de Carrera [13] usado en el Capítulo 4.

Para el coeficiente de forma y siguiendo las líneas discutidas en capítulos anteriores, como el Capítulo 4; las dimensiones y masas se sacaban del Proyecto de Fin de Carrera mencionado varias veces 4, donde se apreciaba que los lados de los prismas que conformaban la torre, en general, eran celosías. Por lo tanto, el coeficiente de forma a elegir será de 1,7, ya que las caras son planas.

Para el área, si suponemos que el viento sopla paralelo al suelo, no va a afectar ni a q_2 ni a q_5, ya que el carrito está incrustado en la pluma, se considera que no se ve afectado, y el péndulo actúa totalmente perpendicular al suelo, por lo que tampoco se ve afectado.
q_1 sufrirá el momento que añadirá el viento sobre toda la pluma, por lo que, si la pluma mide 35 metros (Apartado 2.3) y el ancho de la sección es de 1,2 metros, el área será de 42 m^2.

La fuerza ejercida en toda la pluma es :

$$F_{w_{pluma}} = A \cdot p \cdot C_f = 42 \cdot 0,25 \cdot 1,7 = 17,85\ kN$$

Como esta fuerza se distribuye a lo largo de toda la pluma, se va a suponer que crea un momento desde el centroide, a la mitad de la pluma:

$$M_{w_{pluma}} = F_{w_{pluma}} \cdot 17,5 = 312,375\ kNm$$

Por tanto, esa será la fuerza más crítica a aplicar en q_1 para simular la perturbación del viento, ya que se supone que incide de forma totalmente perpendicular.

La fuerza que sufre la carga se rige por otra fórmula. Al considerar que la grúa puede ser clasificada dentro de todos los tipos normales de grúas que se instalan al aire libre, la fórmula es la siguiente:

$$f = 0,03 \cdot m \cdot g$$

donde

- f es la fuerza del viento sobre la carga en kN
- m es la masa de la carga móvil en t
- g es la aceleración de la gravedad en m/s^2

En este caso no hay carga; sin embargo, para aplicar alguna fuerza en el extremo, se tomará la masa del gancho que, aunque no ha sido considerada, en el Proyecto de Fin de Carrera de Abel Múñoz Ramírez de la UPC [13], el gancho seleccionado en el apartado 7.3.2 tiene un peso de 2,5 kg. Las unidades no están indicadas en el documento, pero se ha accedido al catálogo que utiliza e indica que se expresan en kilogramos.

$$f_{w_{extremo}} = 0,03 \cdot m \cdot g = 0,03 \cdot \frac{2,5}{1000} \cdot 9,81 = 0,00073575\ kN = 0,73575\ N$$

Esta fuerza provoca una fuerza lateral en el cable; para poder modelarla y dado que el cable en el modelo se ha supuesto rígido, se asume que se genera un momento, tanto en q_3 como en q_4, que equivaldrá a:

$$M_{w_{cable}} = F_{w_{extremo}} \cdot (15 + q_5) = 0,00073575 \cdot (15 + q_5)\ kNm$$

Para suponer el caso más crítico, se obtendrá el mayor valor de q_5:

$$M_{w_{cable}} = 0,00073575 \cdot (15 + q_{5_{mx}}) = 0,00073575 \cdot (15 + 24) = 0,02869425\ kNm = 28,7\ Nm$$

Dentro de Simulink, se realizará la inclusión de dichas perturbaciones como si durante 10 segundos crecieran hasta su máximo, se mantienen en ese punto otros 10 segundos y desaparecen posteriormente simulando las rachas de viento. Se permiten las perturbaciones a partir de los 10 segundos de simulación y se bloquean a los 150 segundos para que comience y termine en puntos de estabilidad. La implementación es la siguiente:

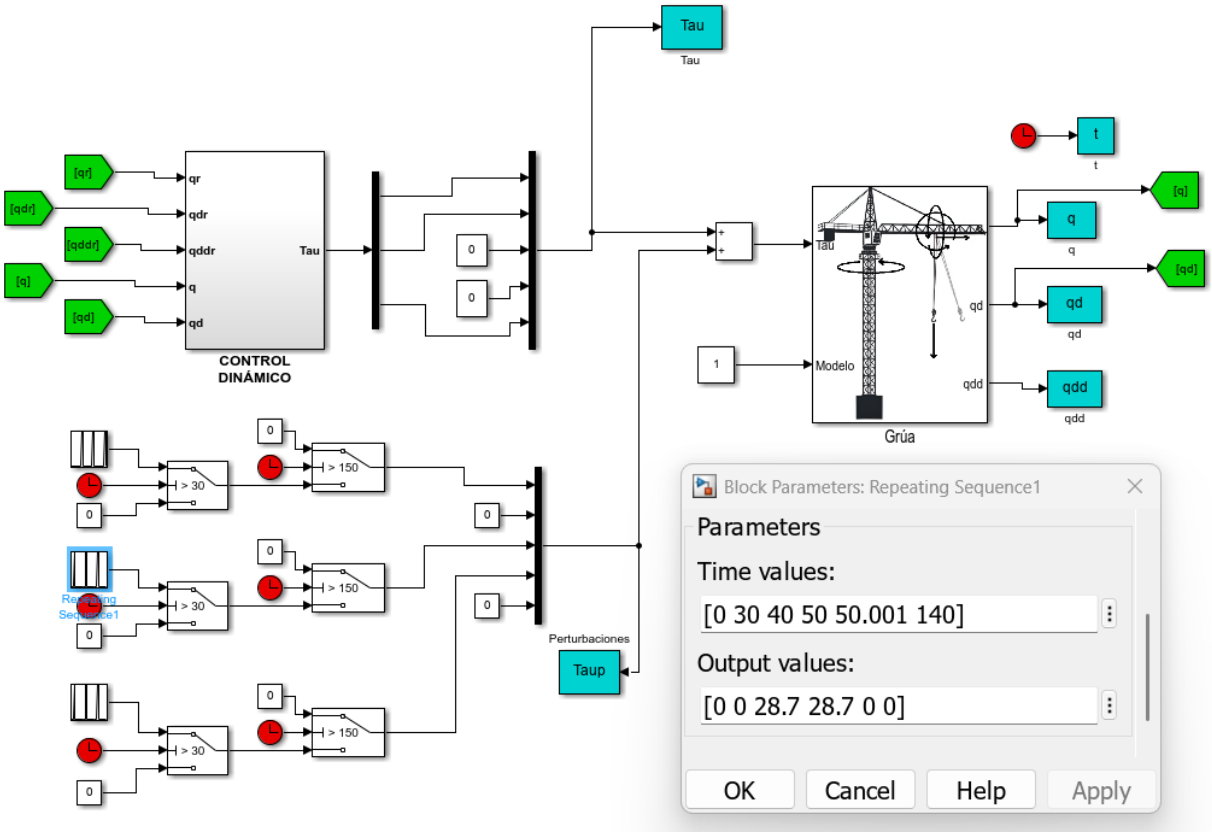


Figura 5.68 Implementación de las perturbaciones en Simulink.

Los resultados obtenidos tras la simulación son los siguientes:

- PD con tiempo de subida de 0,1 segundos:

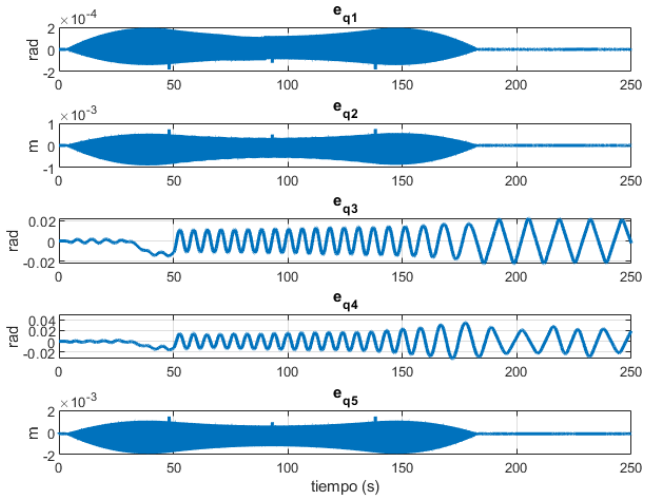


Figura 5.69 Errores en las articulaciones. PD con tiempo de subida de 0,1 segundos y perturbaciones.

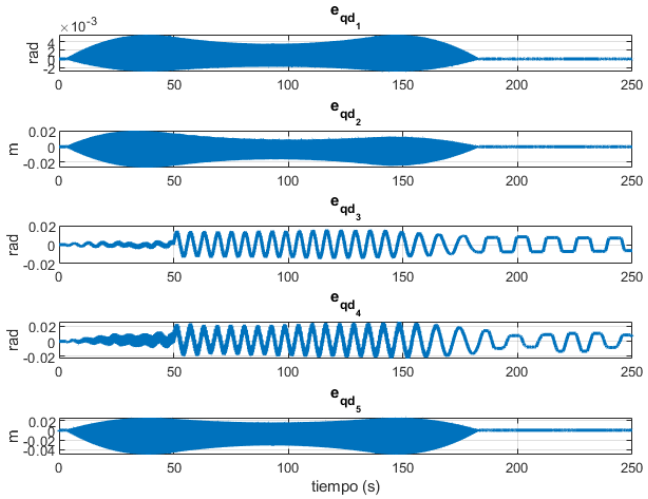


Figura 5.70 Errores en las velocidades articulaciones. PD con tiempo de subida de 0,1 segundos y perturbaciones.

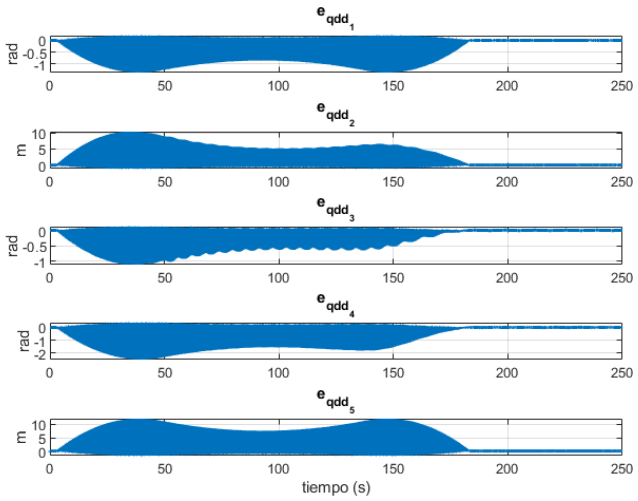


Figura 5.71 Errores en las aceleraciones articulaciones. PD con tiempo de subida de 0,1 segundos y perturbaciones.

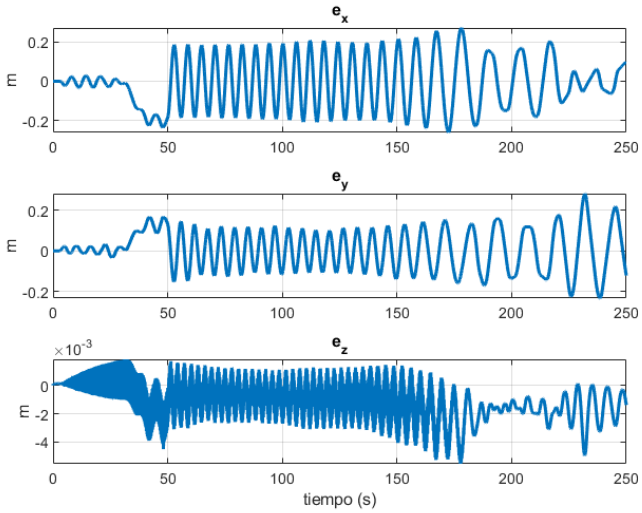


Figura 5.72 Errores cartesianos. PD con tiempo de subida de 0,1 segundos y perturbaciones.

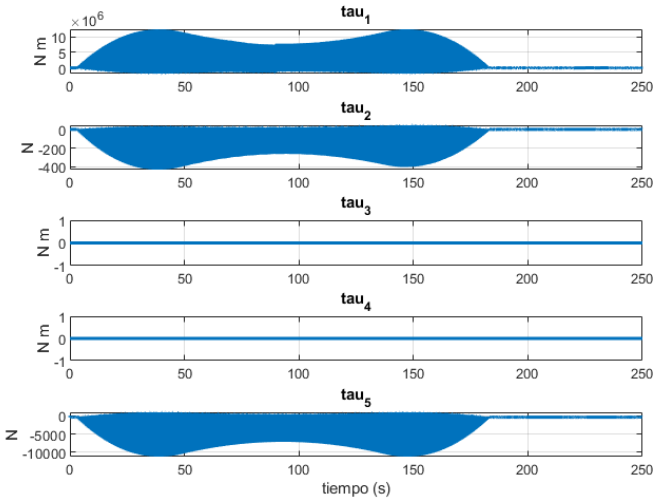


Figura 5.73 Señales de control. PD con tiempo de subida de 0,1 segundos y perturbaciones.

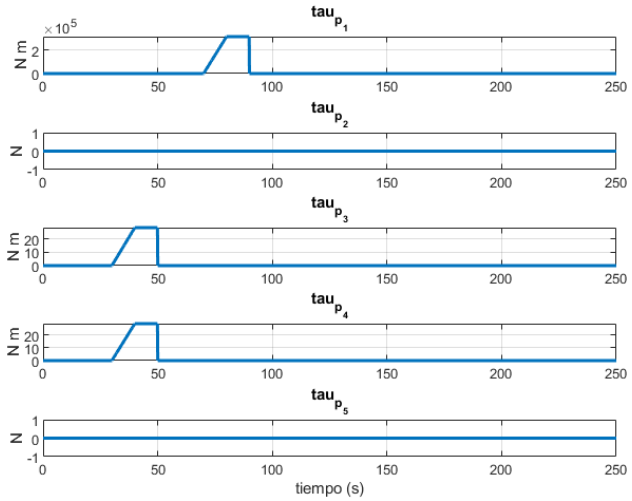


Figura 5.74 Perturbaciones. PD con tiempo de subida de 0,1 segundos y perturbaciones.

En este controlador sólo se aprecian las perturbaciones del péndulo debido a que la señal de control de q_1 es tan grande que la opaca por completo, por lo que prácticamente no tiene efecto sobre la articulación.

- PD con tiempo de subida de q_1 de 2 segundos y los demás a 0,5 segundos:

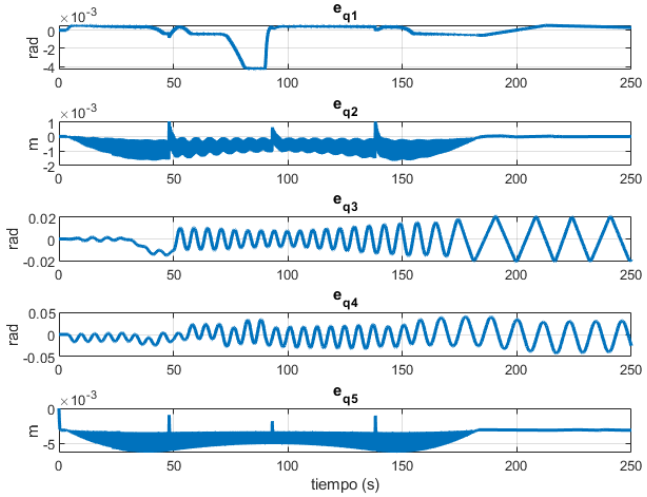


Figura 5.75 Errores en las articulaciones. PD con tiempos de subida diferentes y perturbaciones.

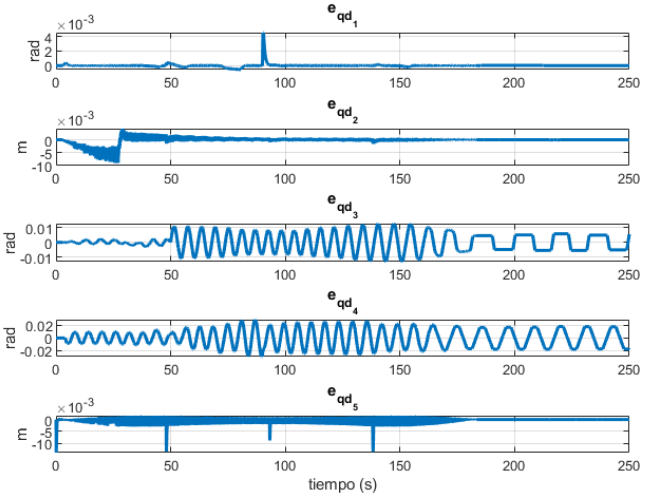


Figura 5.76 Errores en las velocidades articulaciones. PD con tiempos de subida diferentes y perturbaciones.

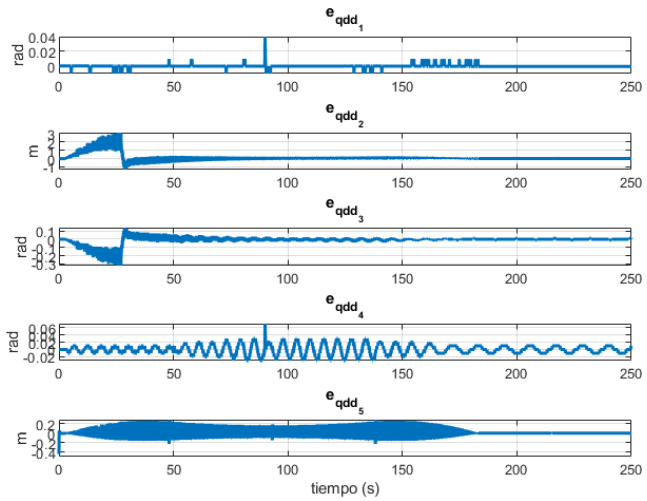


Figura 5.77 Errores en las aceleraciones articulaciones. PD con tiempos de subida diferentes y perturbaciones.

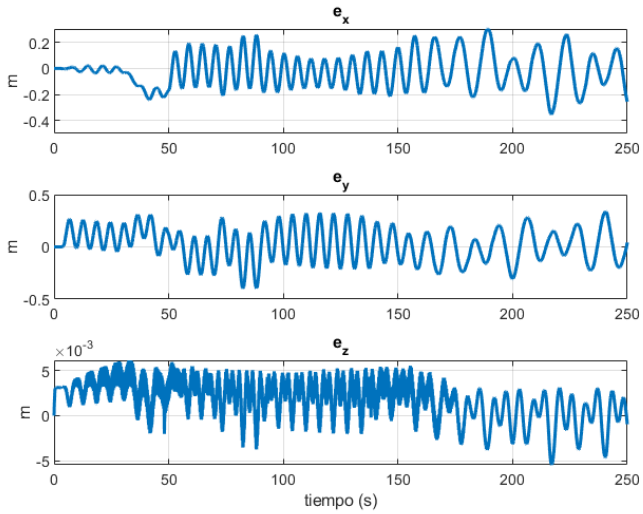


Figura 5.78 Errores cartesianos. PD con tiempos de subida diferentes y perturbaciones.

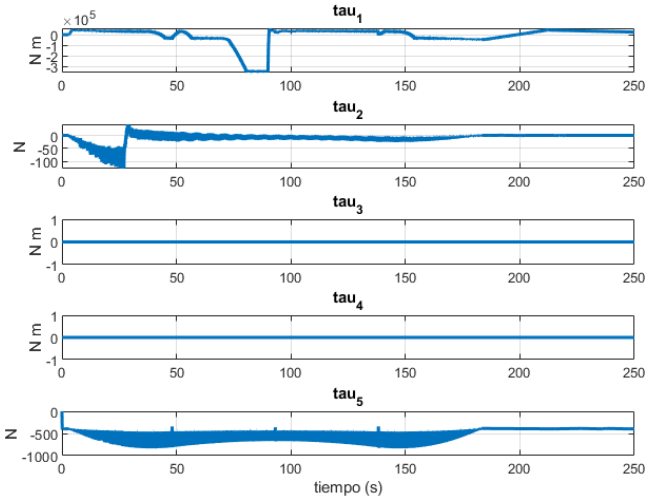


Figura 5.79 Señales de control. PD con tiempos de subida diferentes y perturbaciones.

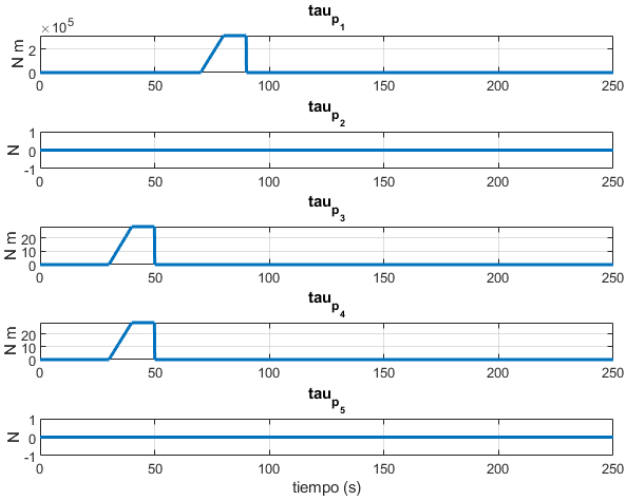


Figura 5.80 Perturbaciones. PD con tiempos de subida diferentes y perturbaciones.

Con el cambio de tiempos de subida del PD; es decir, las señales de control se reducen, se observa que efectivamente el control no es capaz de rechazar las perturbaciones y se aprecia un aumento del error en q_1. El péndulo también sufre error, pero en ningún momento se corrige ya que no se tiene en cuenta.

- PID con tiempo de subida de 0,1 segundos:

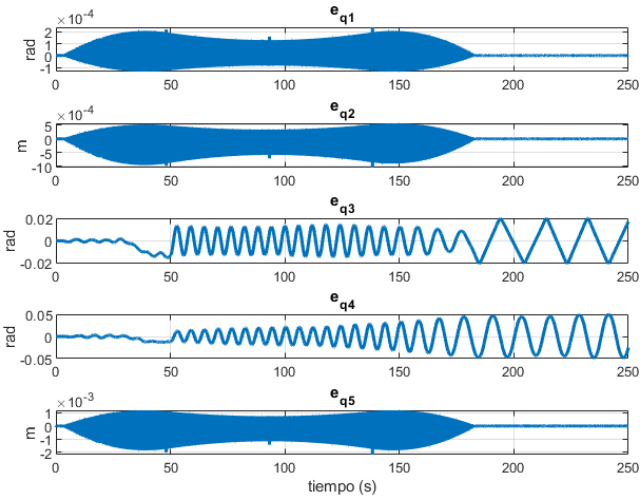


Figura 5.81 Errores en las articulaciones. PID con tiempo de subida de 0,1 segundos y perturbaciones.

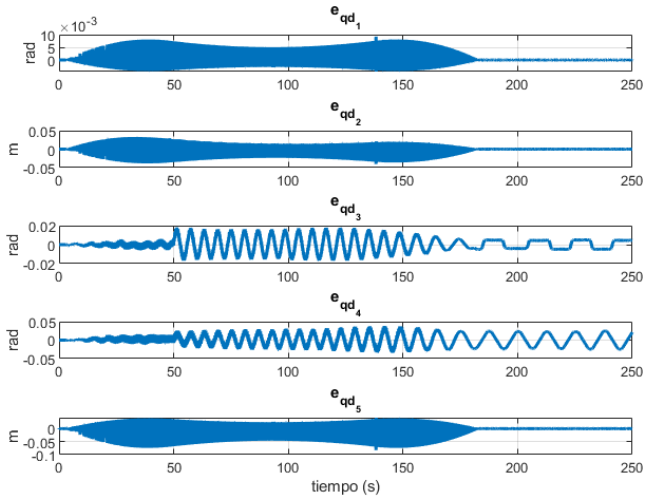


Figura 5.82 Errores en las velocidades articulaciones. PID con tiempo de subida de 0,1 segundos y perturbaciones.

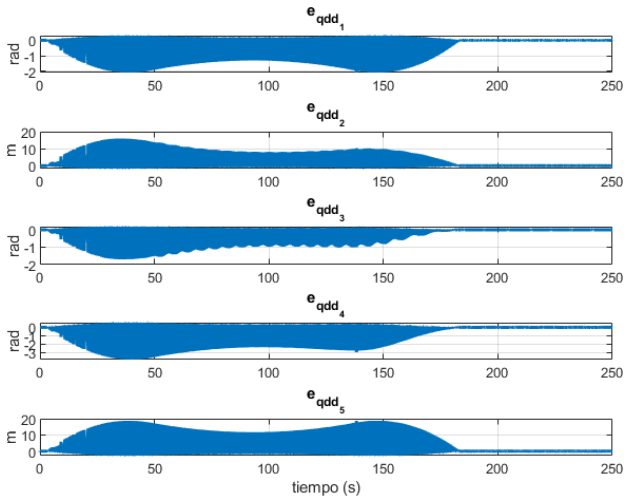


Figura 5.83 Errores en las aceleraciones articulaciones. PID con tiempo de subida de 0,1 segundos y perturbaciones.

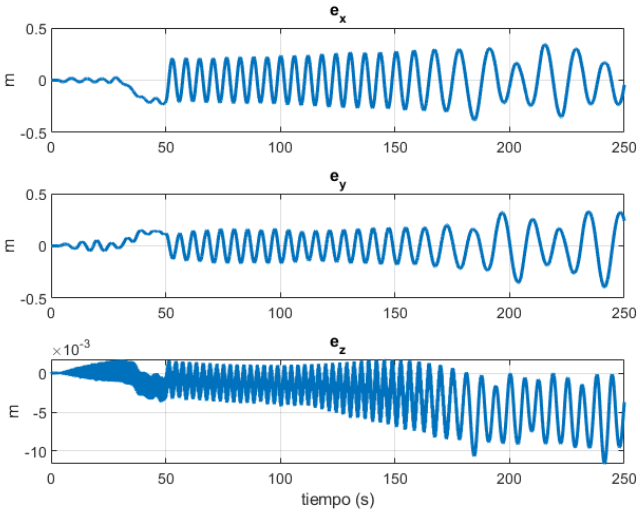


Figura 5.84 Errores cartesianos. PID con tiempo de subida de 0,1 segundos y perturbaciones.

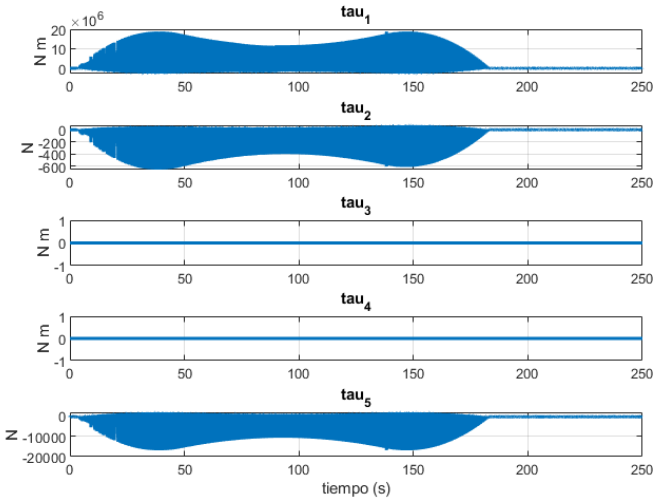


Figura 5.85 Señales de control. PID con tiempo de subida de 0,1 segundos y perturbaciones.

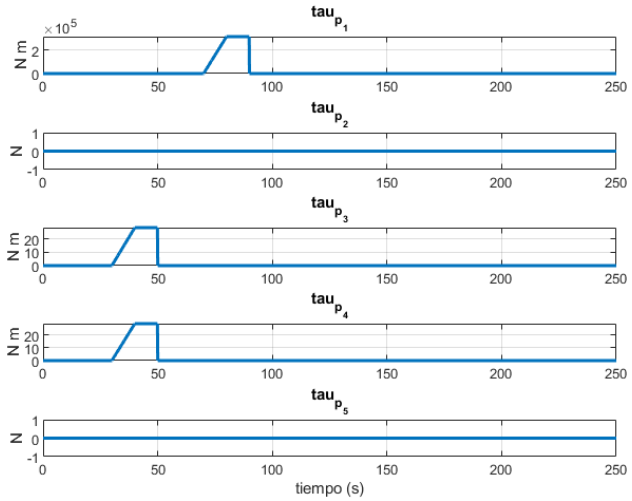


Figura 5.86 Perturbaciones. PID con tiempo de subida de 0,1 segundos y perturbaciones.

Al tener el efecto integral en el controlador, el error debido a las perturbaciones debería desaparecer; sin embargo, al igual que con el PD, la señal de control es mucho mayor y las opaca, anulando su efecto. Se sabe que se aplica, ya que el péndulo sufre las oscilaciones que introducen las perturbaciones.

- PID con tiempo de subida de q_1 de 2 segundos y los demás a 0,5 segundos:

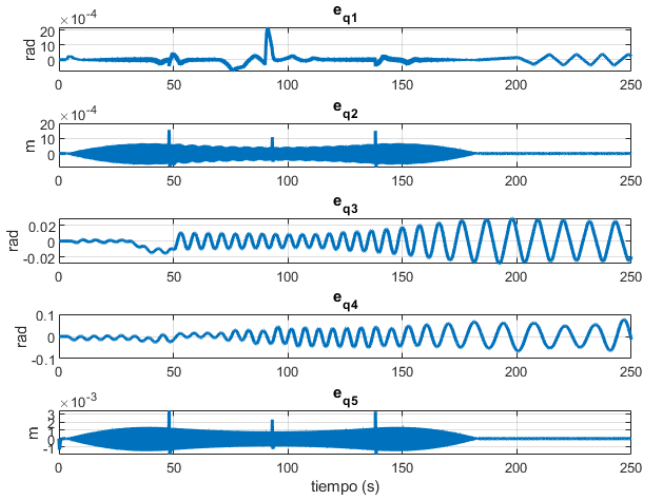


Figura 5.87 Errores en las articulaciones. PID con tiempos de subida diferentes y perturbaciones.

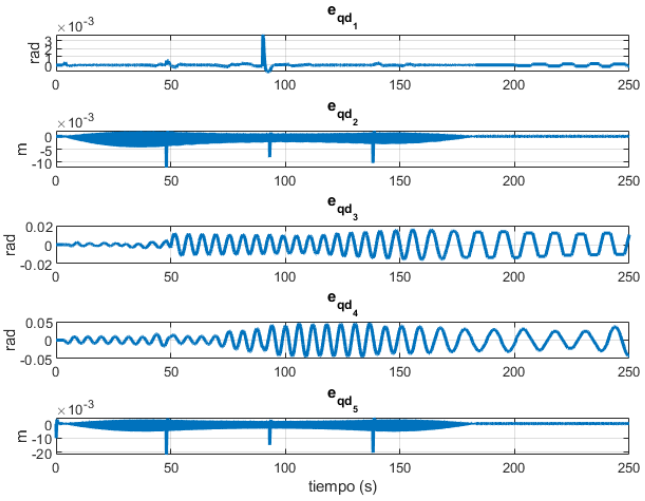


Figura 5.88 Errores en las velocidades articulaciones. PID con tiempos de subida diferentes y perturbaciones.

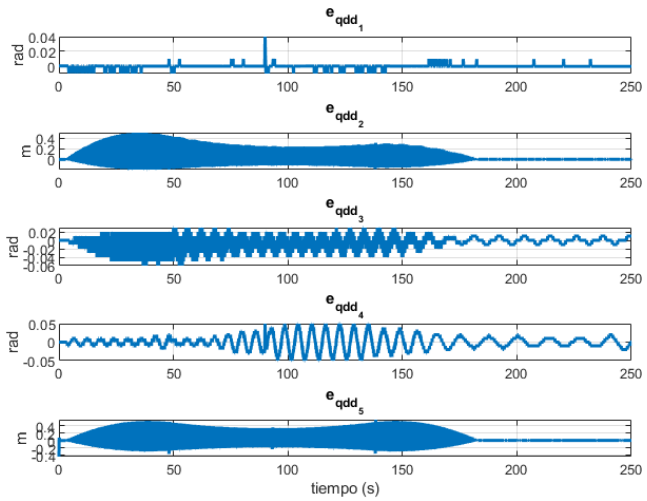


Figura 5.89 Errores en las aceleraciones articulaciones. PID con tiempos de subida diferentes y perturbaciones.

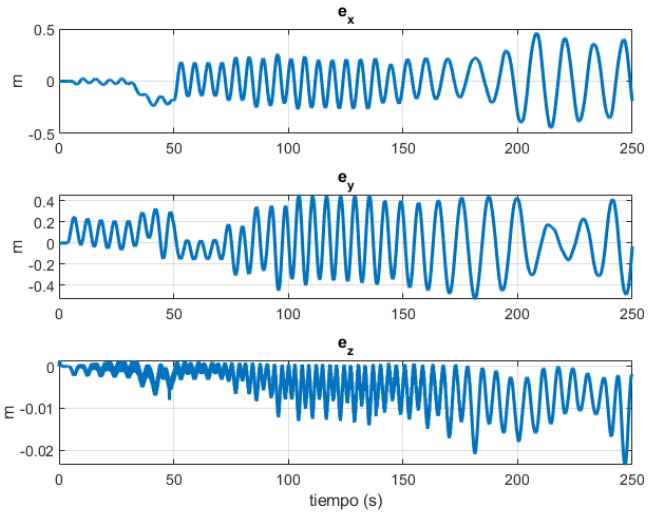


Figura 5.90 Errores cartesianos. PID con tiempos de subida diferentes y perturbaciones.

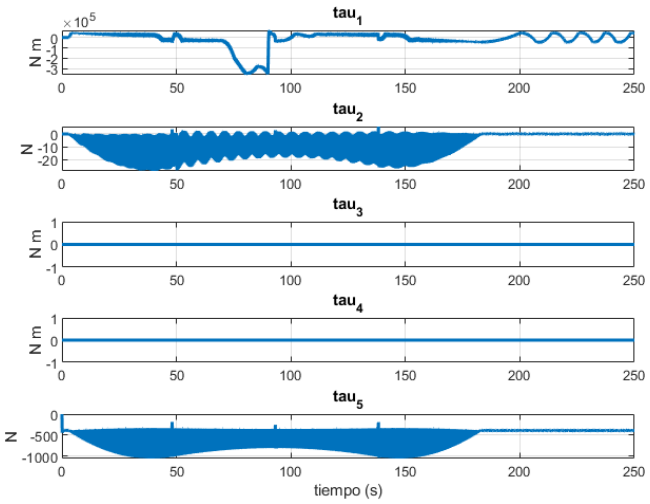


Figura 5.91 Señales de control. PID con tiempos de subida diferentes y perturbaciones.

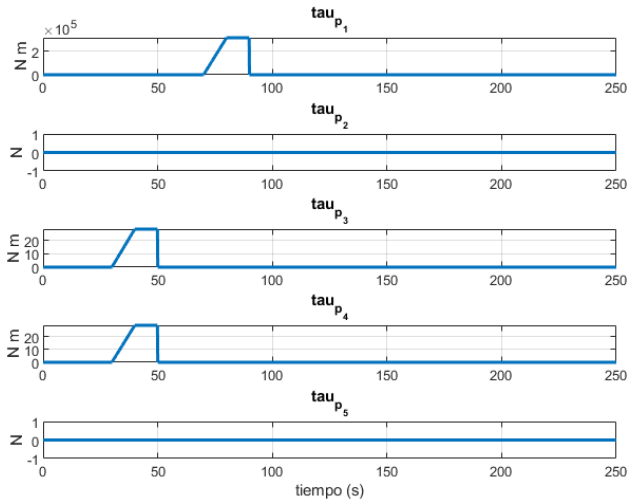


Figura 5.92 Perturbaciones. PID con tiempos de subida diferentes y perturbaciones.

Al igual que con el PD al reducir la señal de control sí se aprecian las perturbaciones de q_1; pero al contrario que en otros casos, para esta sintonización se aprecia el efecto integral ya que tras un pico de error, este se establece cerca del cero de nuevo, lo que va en sintonía con la reducción de la señal de control para compensar dicho error.
En cambio, al igual que en todos estos controladores que no tienen en cuenta las señales subactuadas, se acentúan los errores en el péndulo.

- LQR:

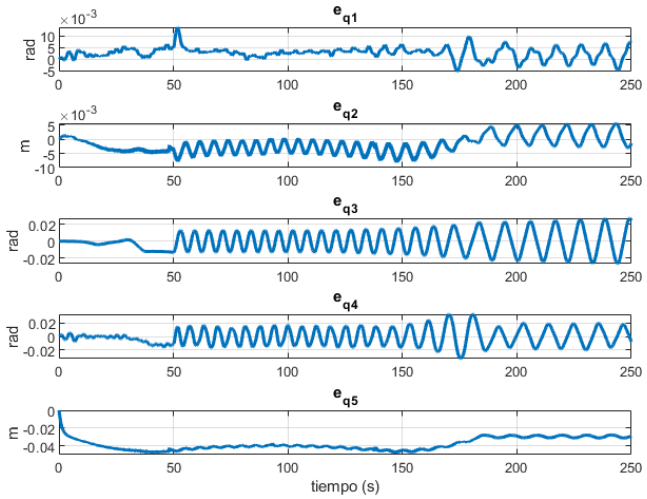


Figura 5.93 Errores en las articulaciones. LQR con perturbaciones.

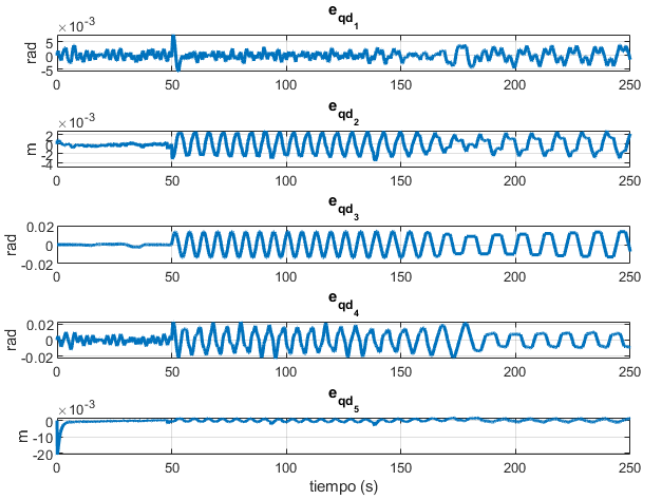


Figura 5.94 Errores en las velocidades articulaciones. LQR con perturbaciones.

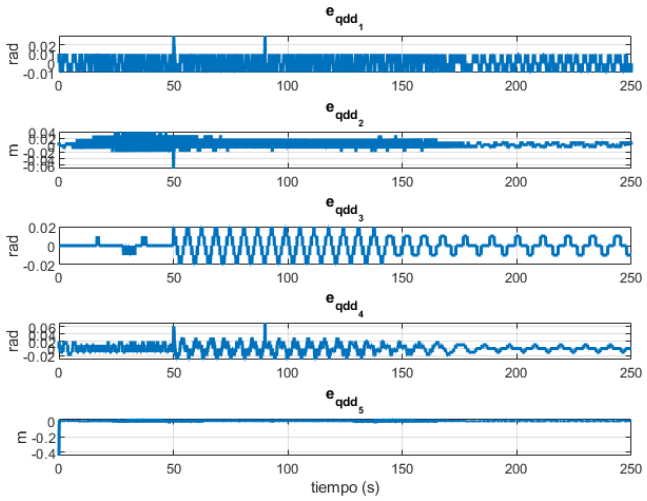


Figura 5.95 Errores en las aceleraciones articulaciones. LQR con perturbaciones.

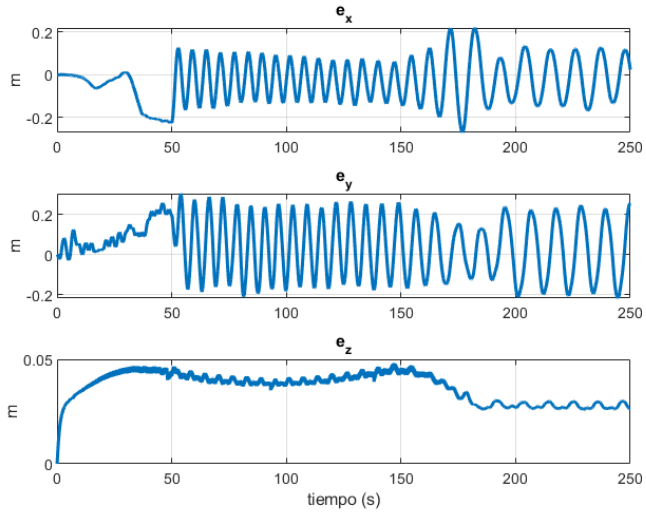


Figura 5.96 Errores cartesianos. LQR con perturbaciones.

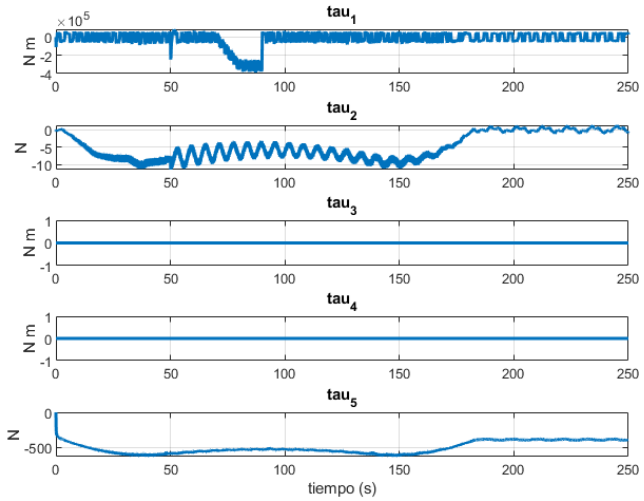


Figura 5.97 Señales de control. LQR con perturbaciones.

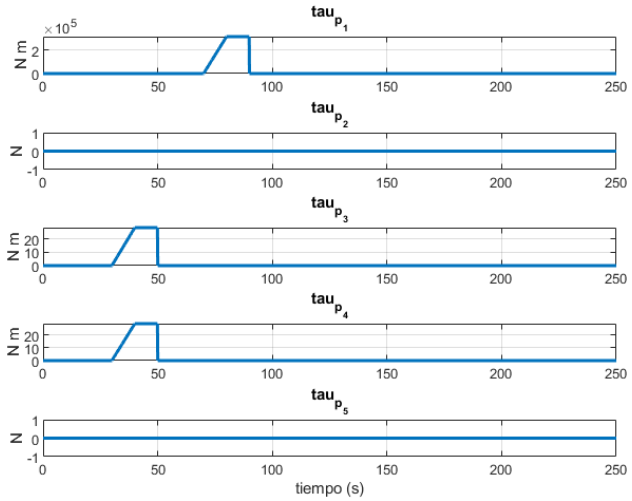


Figura 5.98 Perturbaciones. LQR con perturbaciones.

El LQR rechaza más rápidamente que los PID la perturbación de q_1; sin embargo, no es capaz de amortiguar las oscilaciones del péndulo, a pesar de ser un control MIMO que tiene en cuenta dichos errores.

- LQR lento:

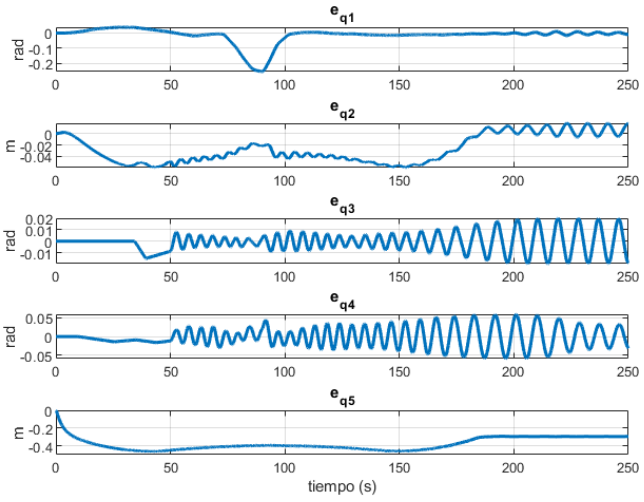


Figura 5.99 Errores en las articulaciones. LQR lento con perturbaciones.

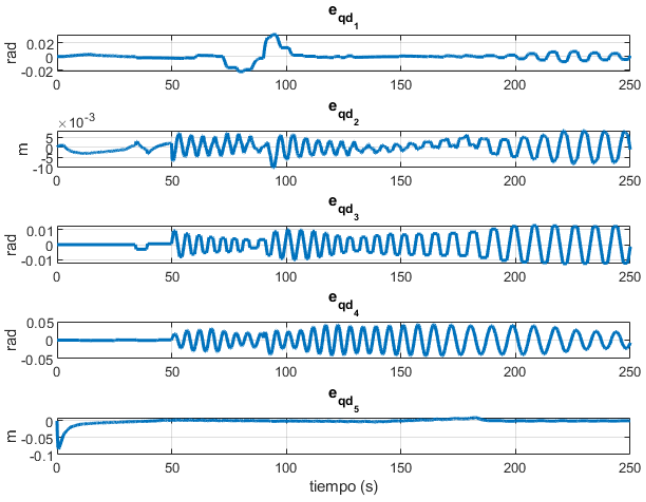


Figura 5.100 Errores en las velocidades articulaciones. LQR lento con perturbaciones.

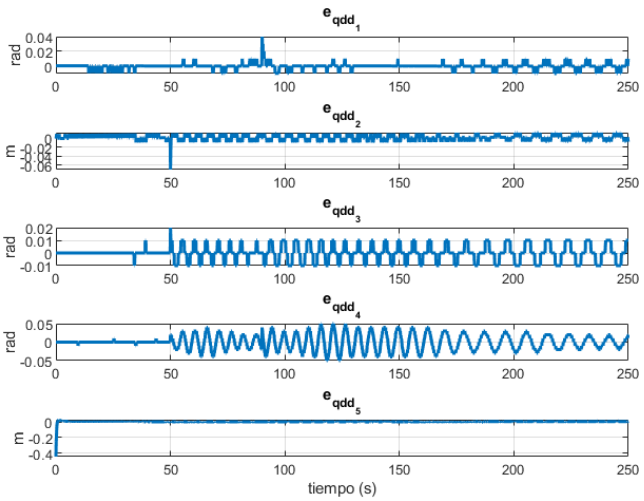


Figura 5.101 Errores en las aceleraciones articulaciones. LQR lento con perturbaciones.

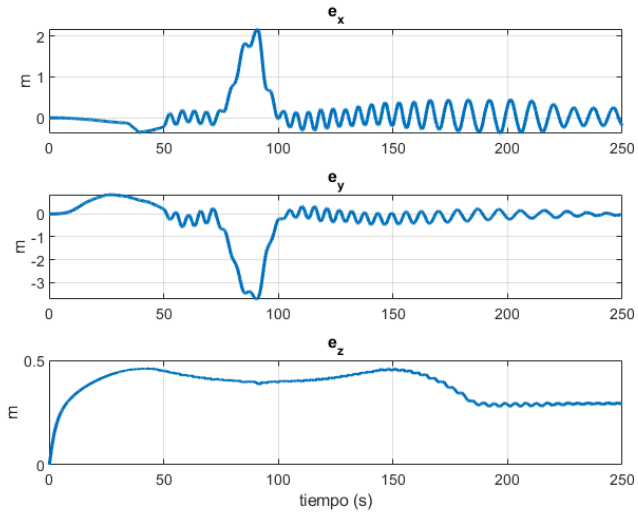


Figura 5.102 Errores cartesianos. LQR lento con perturbaciones.

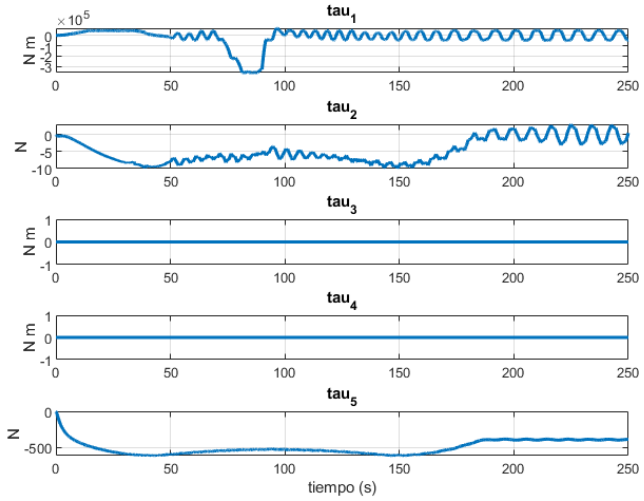


Figura 5.103 Señales de control. LQR lento con perturbaciones.

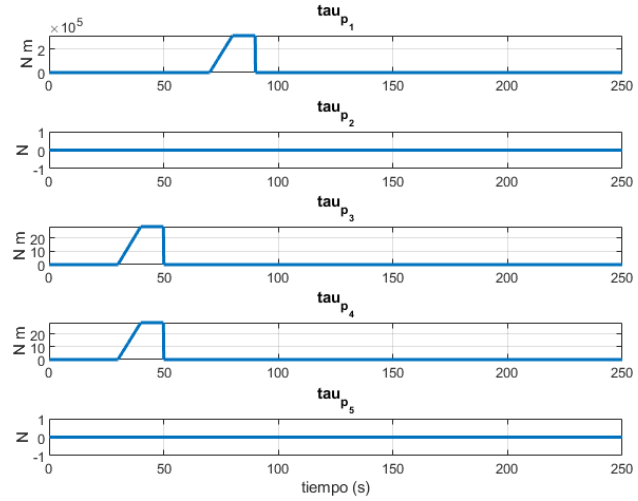


Figura 5.104 Perturbaciones. LQR lento con perturbaciones.

El LQR lento, en cambio, rechaza peor la perturbación de q_1, ya que no es tan agresivo, pero sí se puede apreciar que la estabilidad del control permite reducir en ciertos momentos las oscilaciones del péndulo haciendo uso de q_2.

- MPC:

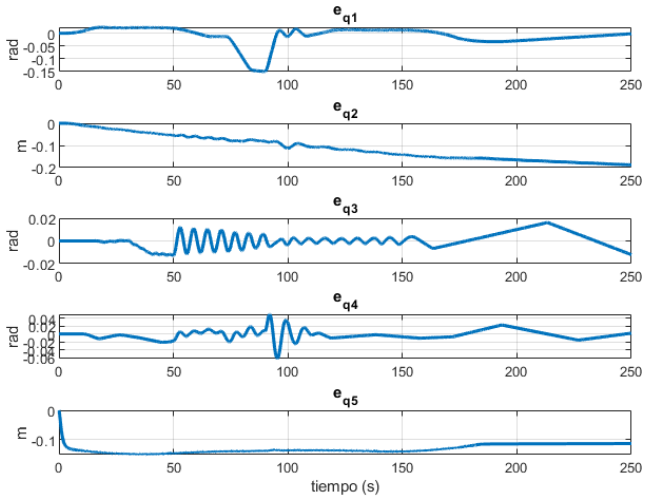


Figura 5.105 Errores en las articulaciones. MPC con perturbaciones.

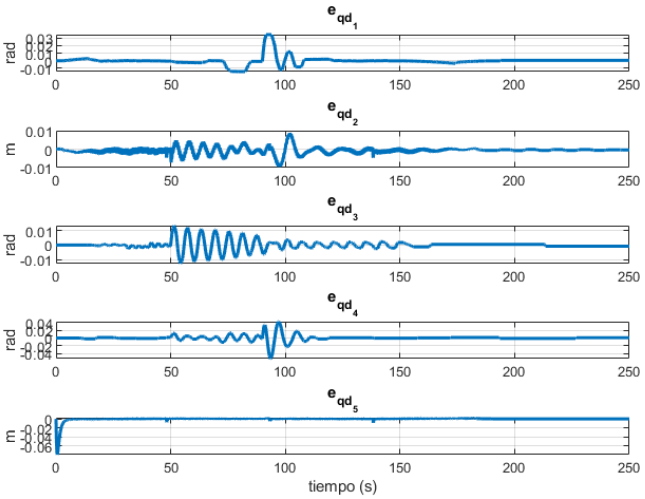


Figura 5.106 Errores en las velocidades articulaciones. MPC con perturbaciones.

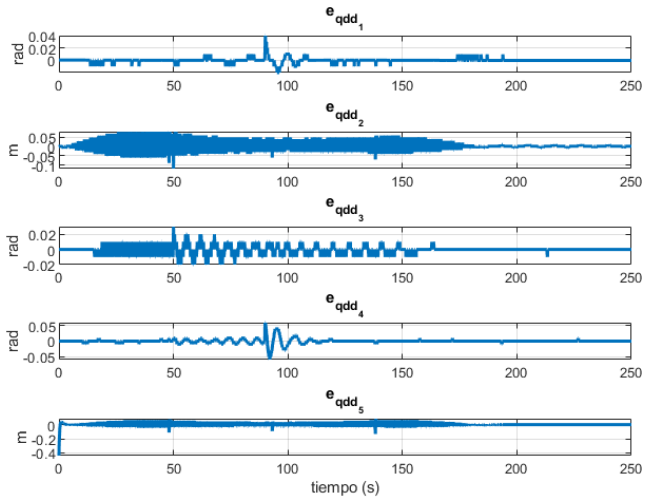


Figura 5.107 Errores en las aceleraciones articulaciones. MPC con perturbaciones.

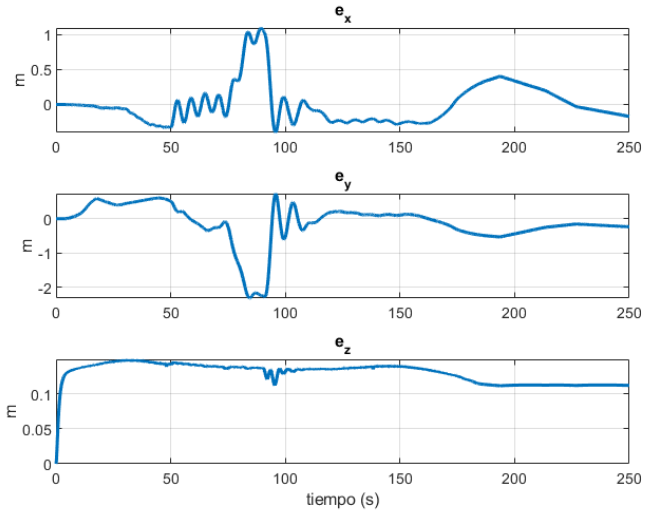


Figura 5.108 Errores cartesianos. MPC con perturbaciones.

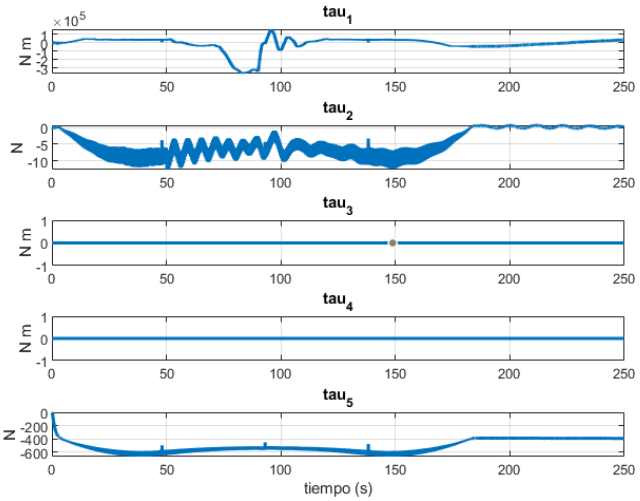


Figura 5.109 Señales de control. MPC con perturbaciones.

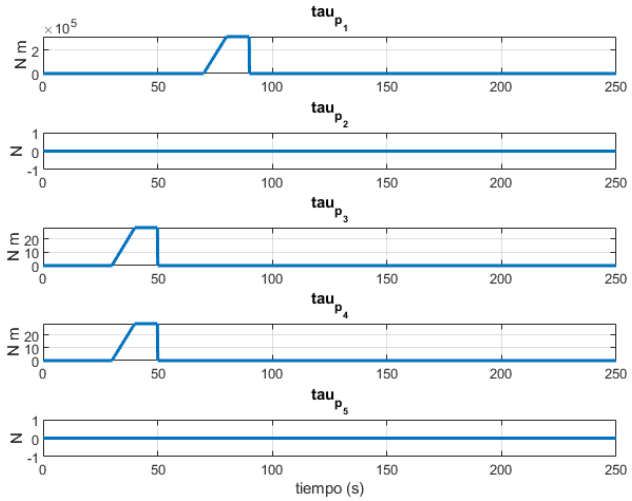


Figura 5.110 Perturbaciones. MPC con perturbaciones.

Con el MPC ocurre justo al contrario que con el LQR, no es capaz de rechazar la perturbación de q_1 principalmente debido a que no es capaz de predecirlo al no tener en cuenta las perturbaciones; sin embargo, cuando la racha de viento deja de afectar al péndulo puede definir las futuras oscilaciones y amortiguarlas reduciendo mucho el error en estas articulaciones.

- MPC lento:

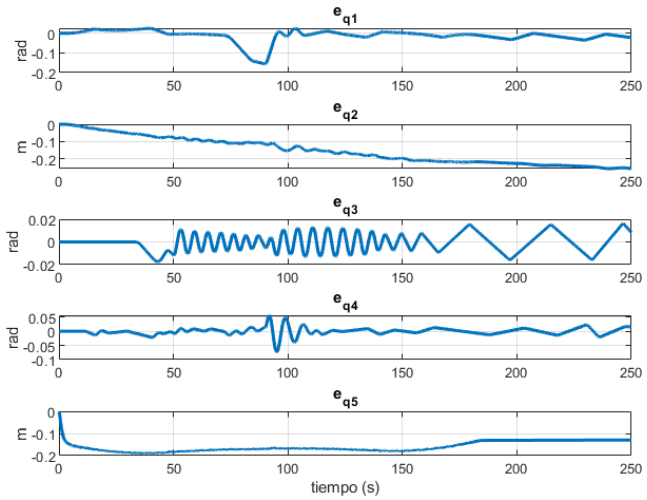


Figura 5.111 Errores en las articulaciones. MPC lento con perturbaciones.

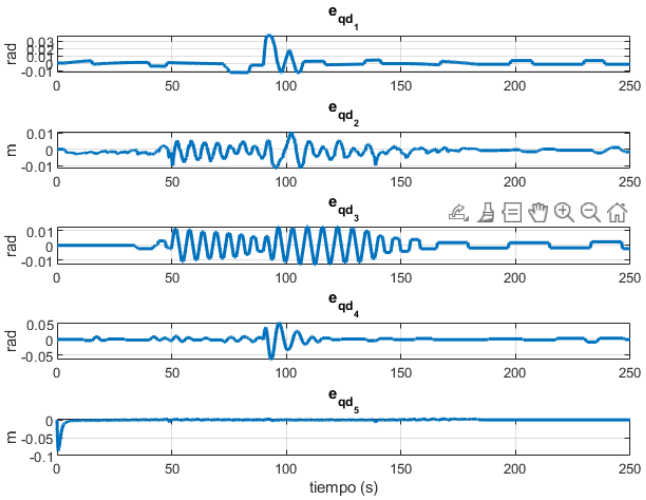


Figura 5.112 Errores en las velocidades articulaciones. MPC lento con perturbaciones.

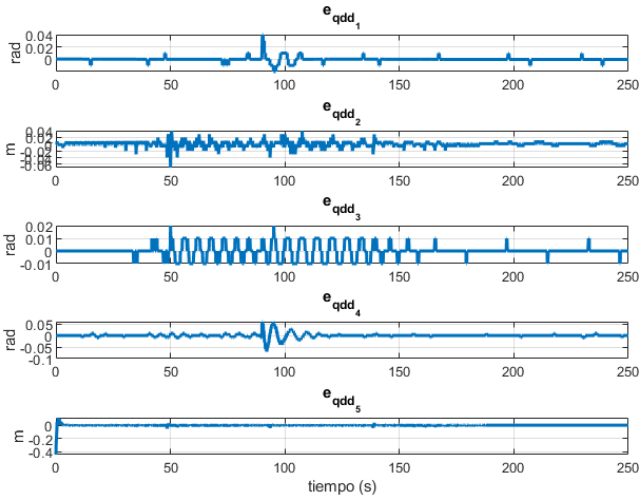


Figura 5.113 Errores en las aceleraciones articulaciones. MPC lento con perturbaciones.

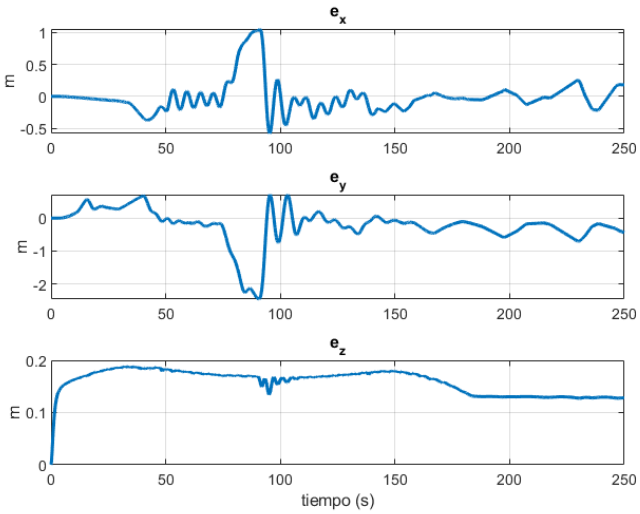


Figura 5.114 Errores cartesianos. MPC lento con perturbaciones.

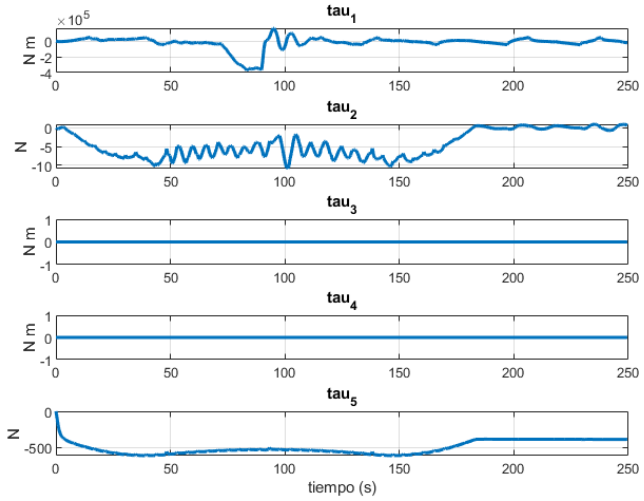


Figura 5.115 Señales de control. MPC lento con perturbaciones.

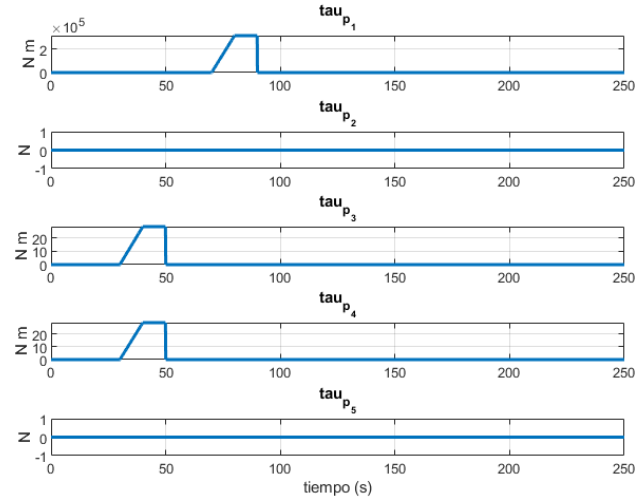


Figura 5.116 Perturbaciones. MPC lento con perturbaciones.

Al aumentar el tiempo de discretización y hacer el control más lento tarda más en amortiguar las oscilaciones pero la evolución de q_1 no varía significativamente. El gran cambio se observa en las señales de control de q_2 y q_5 que se vuelven más estables y menos agresivas.

Claramente, los controladores SISO son inviables, ya que no se tiene en cuenta la oscilación del cable, que es el punto más crítico del sistema y no tenerlo en cuenta puede tener consecuencias graves. Debido a esto, en el análisis de robustez se tendrá en cuenta únicamente el control LQR y MPC.

5.5 Análisis de robustez

Para el análisis de robustez, se parte del Código A.18, donde ya se tiene en cuenta una masa simbólica m que representa la carga que cuelga del cable.
La masa máxima que admite la grúa es de 2,5 t; sin embargo, esta masa sólo se soporta hasta cierto alcance y no en todo el rango posible de la grúa. La masa máxima permitida para cualquier punto de operación del sistema es de 2 t, por lo que será la masa que generará una fuerza adicional:

$$\vec{F}_{extremo} = m \cdot \vec{a} = 2000 \cdot 9,81 \cdot (-\vec{k}) = -19,62\,\vec{k}\ kN$$

Siendo $\vec{k}$ el vector unitario en el eje Z absoluto, apuntando hacia arriba.

Para implementarla, como realmente este cálculo se hará internamente al calcular las matrices simbólicas, únicamente es necesario sustituir la masa m por 2000 kg. Esto se realiza en el Código A.19, donde se añade un apartado nuevo con las mismas matrices, pero incluyendo esta masa que antes se establecía en cero. Tras modificar el código y ejecutarlo, se extraen las nuevas matrices ya sin ninguna variable simbólica y se incluirán en el Código A.20 dentro de las opciones de modelo como el número 3.

Las primeras pruebas se realizarán con los dos tipos de LQR:

- LQR:

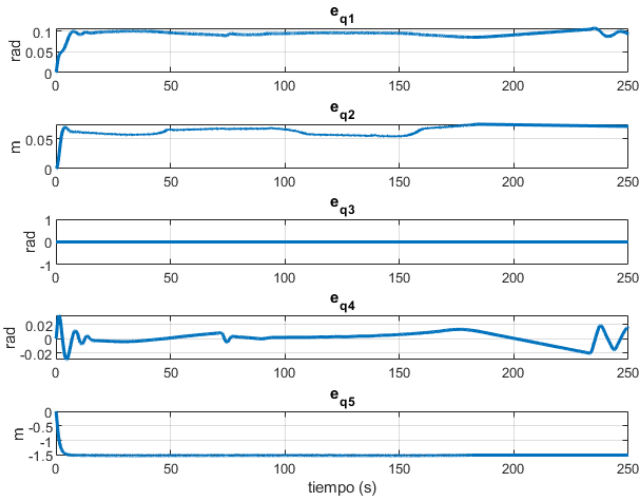


Figura 5.117 Errores en las articulaciones. Análisis de robustez LQR.

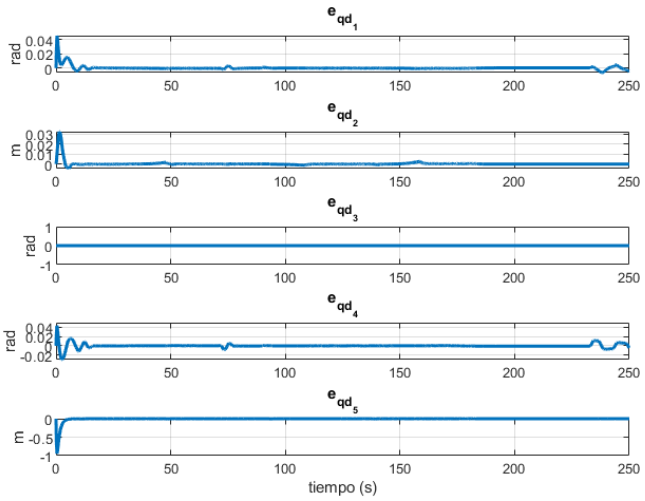


Figura 5.118 Errores en las velocidades articulaciones. Análisis de robustez LQR.

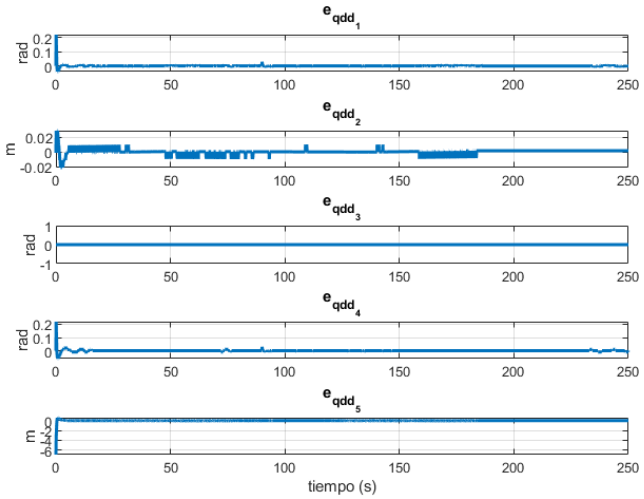


Figura 5.119 Errores en las aceleraciones articulaciones. Análisis de robustez LQR.

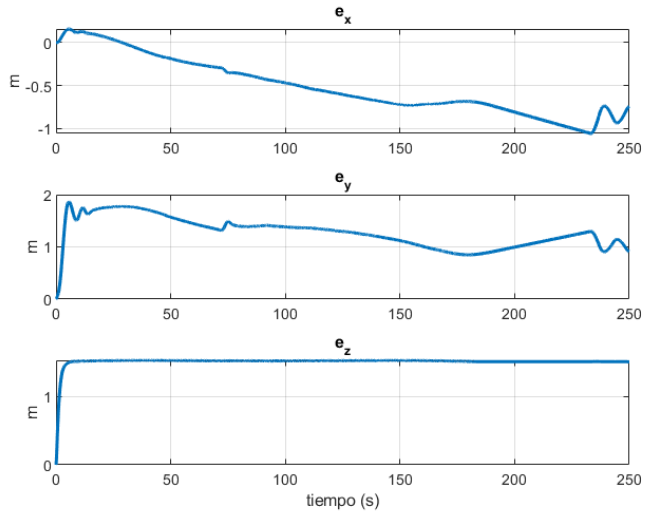


Figura 5.120 Errores cartesianos. Análisis de robustez LQR.

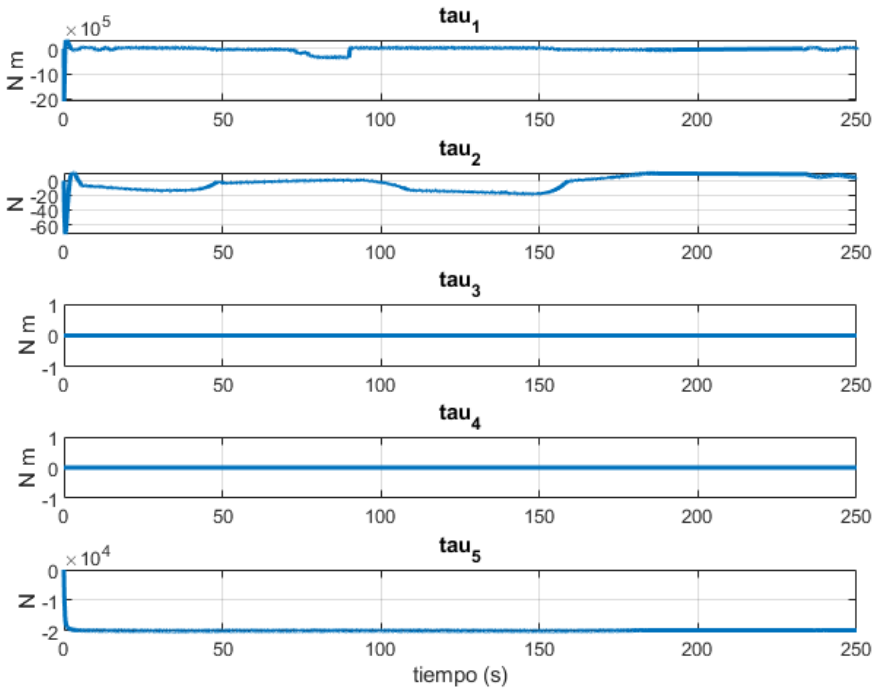


Figura 5.121 Señales de control. Análisis de robustez LQR.

Aunque este tipo de control responde bastante bien a las masas añadidas en el extremo final, presenta errores considerables. Esto se debe principalmente a que el modelo elegido no refleja con exactitud los cambios dinámicos que sufre la grúa al soportar diferentes cargas.

- LQR lento:

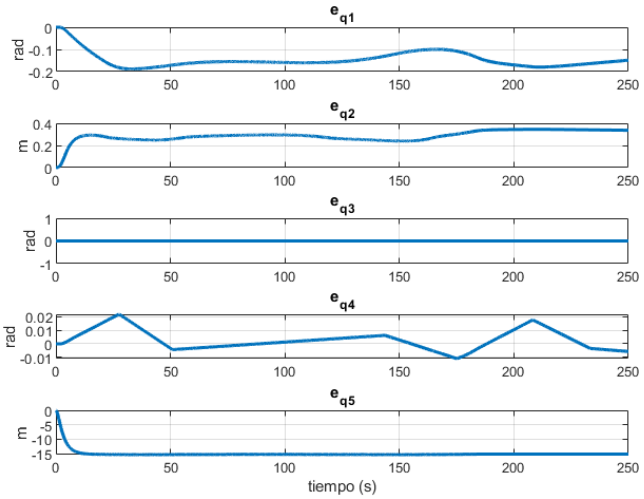


Figura 5.122 Errores en las articulaciones. Análisis de robustez LQR lento.

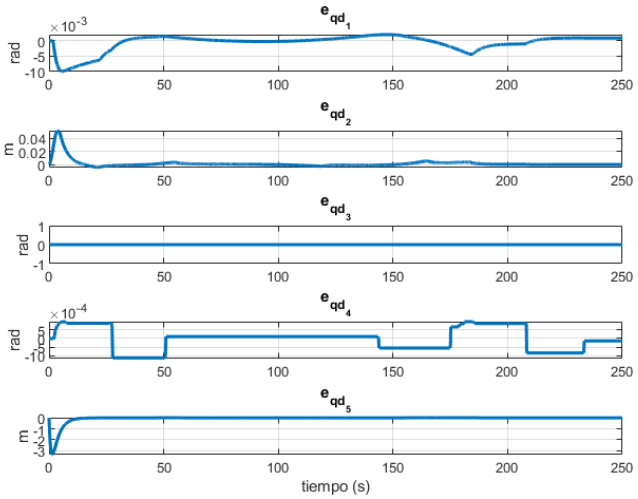


Figura 5.123 Errores en las velocidades articulaciones. Análisis de robustez LQR lento.

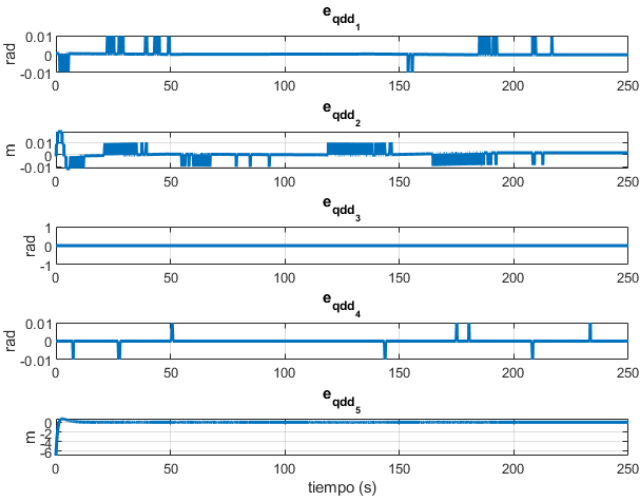


Figura 5.124 Errores en las aceleraciones articulaciones. Análisis de robustez LQR lento.

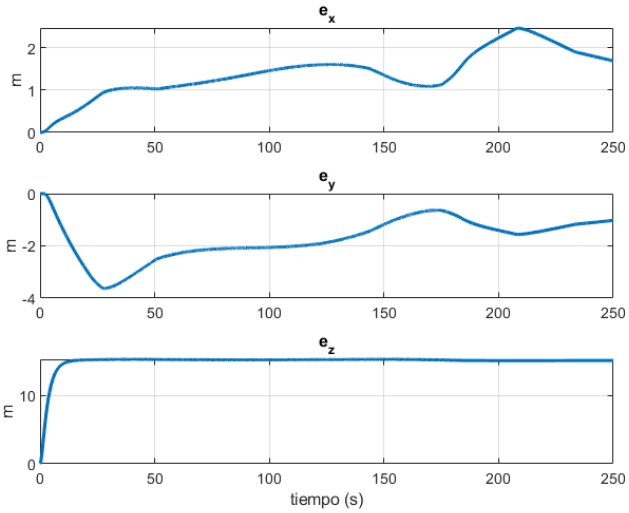


Figura 5.125 Errores cartesianos. Análisis de robustez LQR lento.

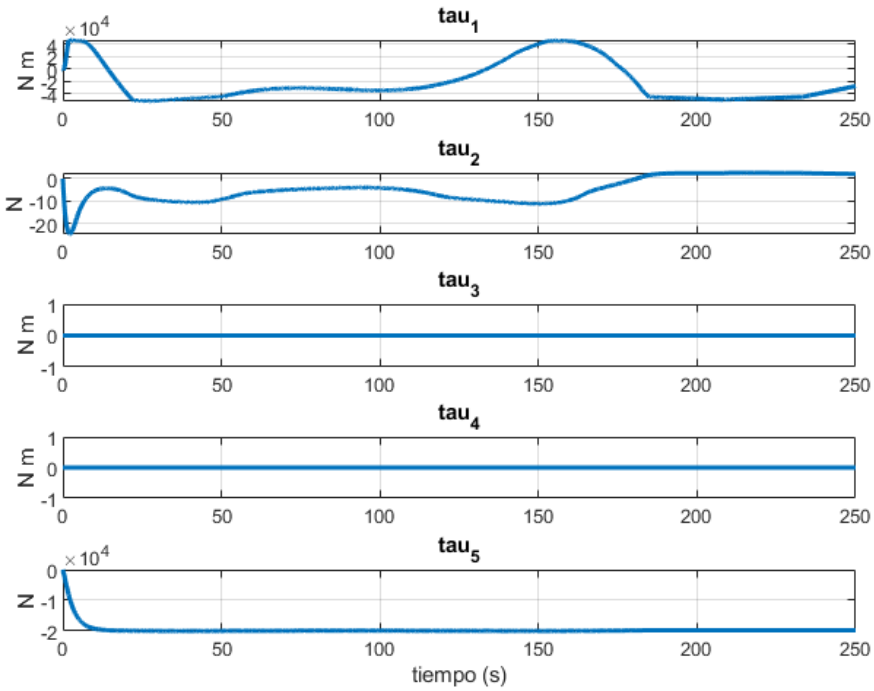


Figura 5.126 Señales de control. Análisis de robustez LQR lento.

Al hacer el control más lento, se pronuncian aún más los errores, ya que no tiene la capacidad de reacción que tiene el controlador anterior. Sobre todo, la variable q_5 aumenta demasiado su error en comparación con el orden de los otros errores.

Tras analizar la respuesta del LQR se pasa a ver si el MPC responde mejor o peor:

- MPC:

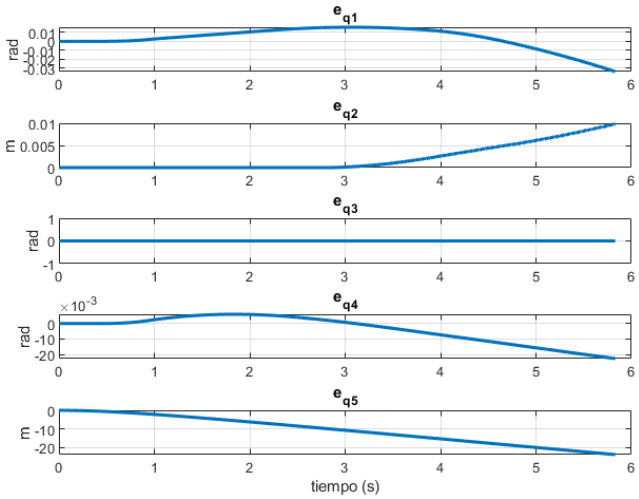


Figura 5.127 Errores en las articulaciones. Análisis de robustez MPC.

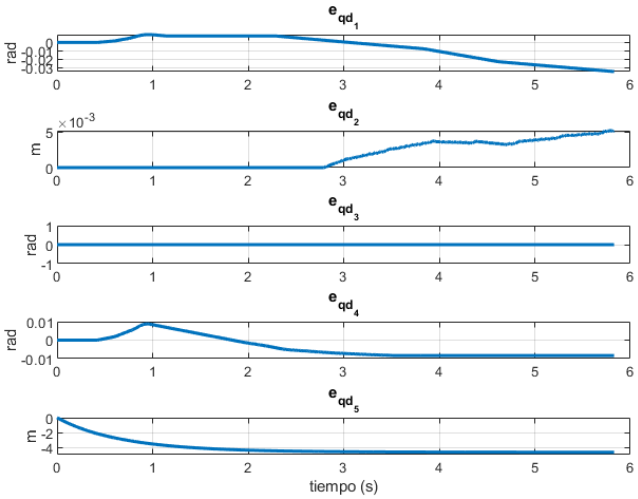


Figura 5.128 Errores en las velocidades articulaciones. Análisis de robustez MPC.

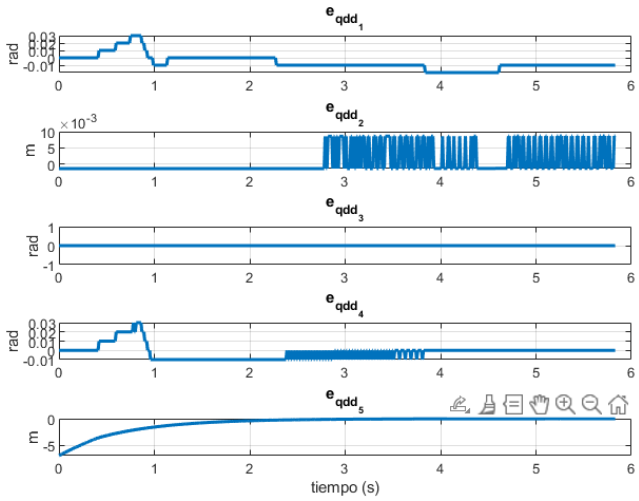


Figura 5.129 Errores en las aceleraciones articulaciones. Análisis de robustez MPC.

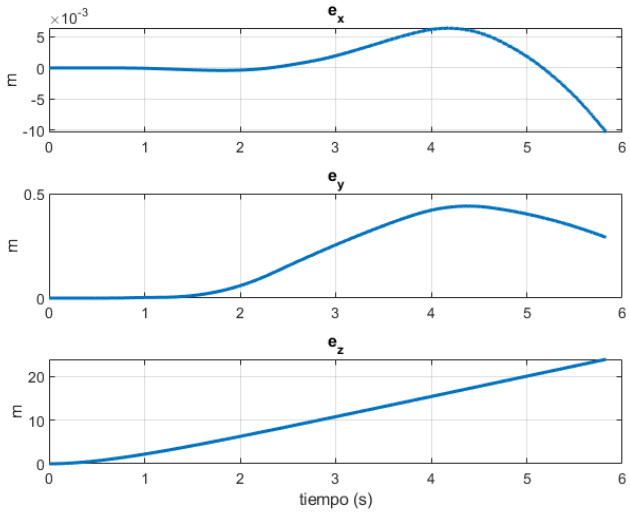


Figura 5.130 Errores cartesianos. Análisis de robustez MPC.

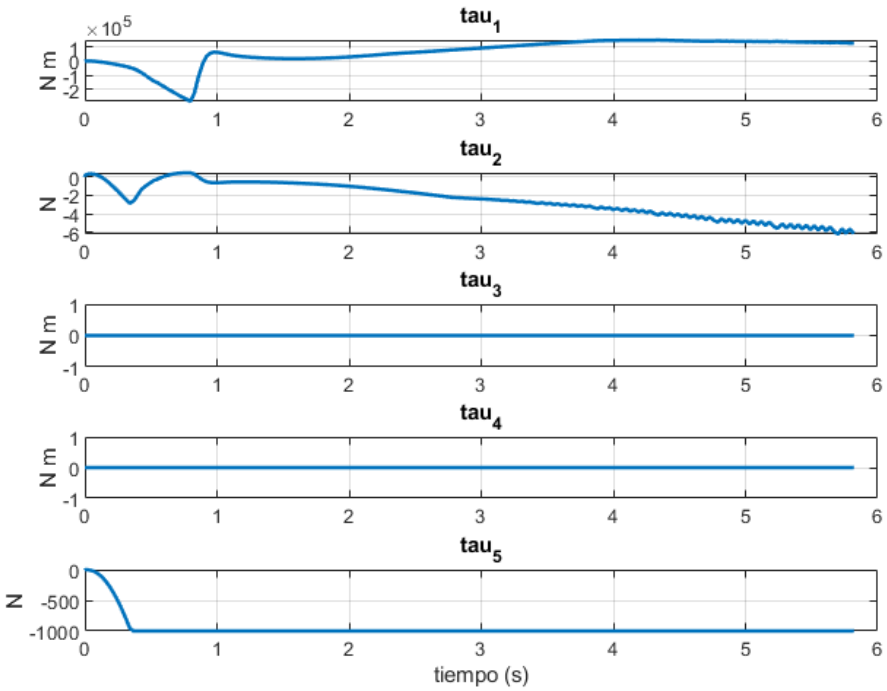


Figura 5.131 Señales de control. Análisis de robustez MPC.

Se observa que la simulación dura apenas cinco segundos debido a que la longitud del cable se vuelve demasiado larga, excediendo sus límites. En la señal de control de q_5, que es la articulación que más sufre el cambio de masa en el extremo, se aprecia que no supera el límite impuesto, ± 1000, lo cual no tiene ningún sentido, ya que previamente se calculó que esta masa ejercería una fuerza de 19,62 kN.
Es imposible que se compense la fuerza que ejerce la masa hacia abajo con ese límite. Por tanto, en los MPC a evaluar se va a modificar el escalado y los límites de la variable q_5:

- Restricciones sobre las variables manipuladas:
  - Motor de giro: $[-5 \times 10^6,\ 5 \times 10^6]$ Nm.
  - Motor del carro: $[-500,\ 500]$ N.
  - Motor de elevación: $[-50000,\ 50000]$ N.
- Valores de escalado de las variables manipuladas:
  - Motor de giro: 5×10^6
  - Motor del carro: 500
  - Motor de elevación: 50000

Al volver a realizar la simulación los resultados son:

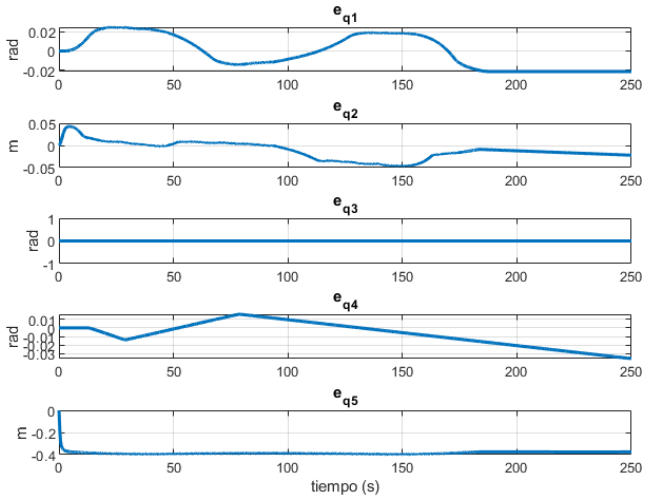


Figura 5.132 Errores en las articulaciones. Análisis de robustez MPC con nuevo escalado.

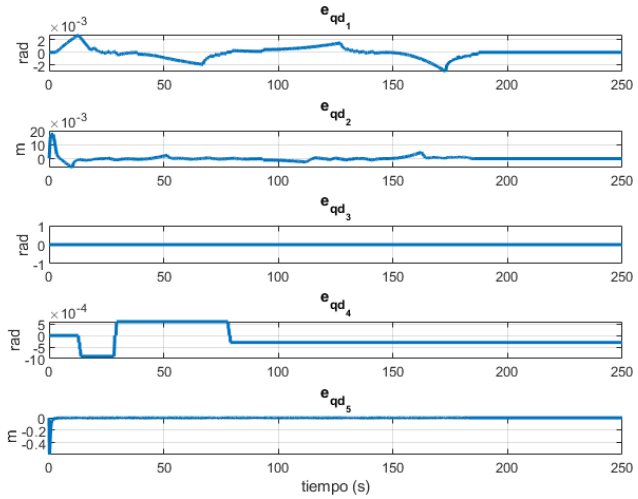


Figura 5.133 Errores en las velocidades articulaciones. Análisis de robustez MPC con nuevo escalado.

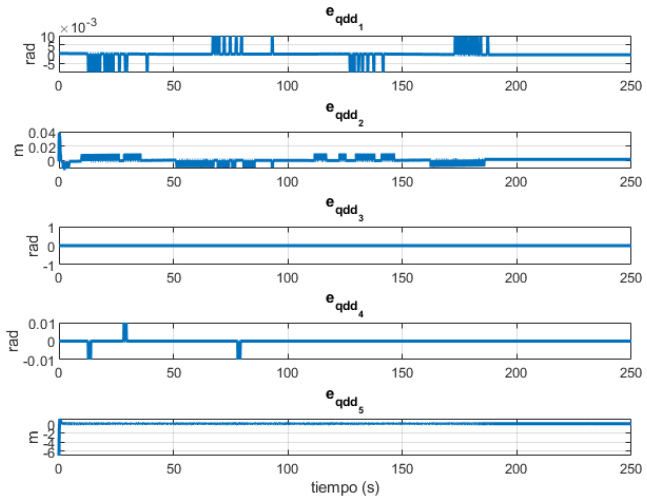


Figura 5.134 Errores en las aceleraciones articulaciones. Análisis de robustez MPC con nuevo escalado.

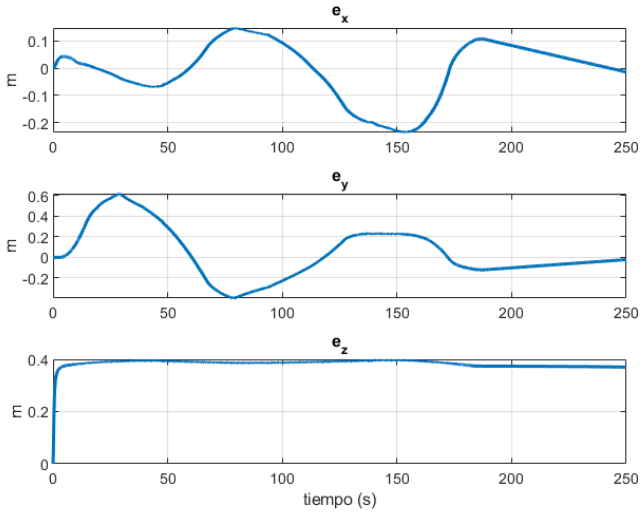


Figura 5.135 Errores cartesianos. Análisis de robustez MPC con nuevo escalado.

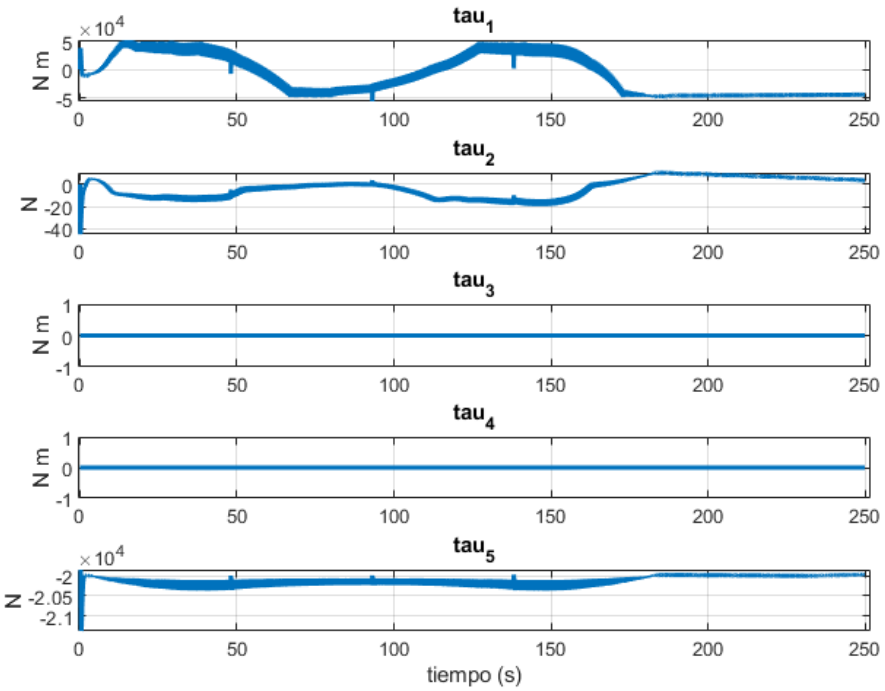


Figura 5.136 Señales de control. Análisis de robustez MPC con nuevo escalado.

Se observa que efectivamente aumenta notablemente la señal de control de q_5, y los errores se acentúan ligeramente en esta articulación, aunque siguen siendo bastante pequeños.

Se comprueba si la respuesta del MCP más lento se asemeja a este control haciendo uso también del escalado previo:

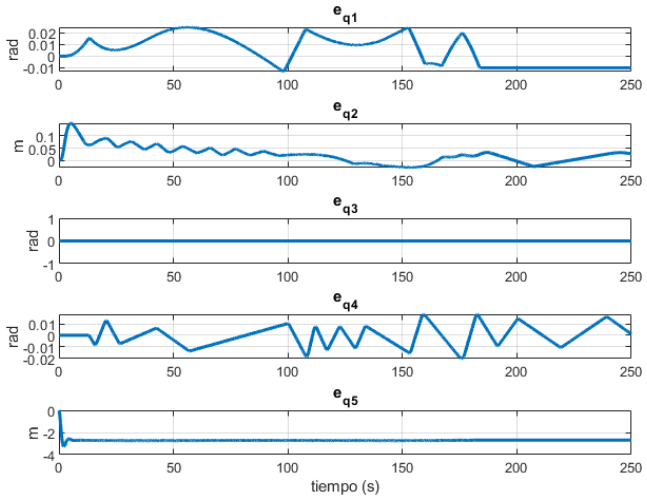


Figura 5.137 Errores en las articulaciones. Análisis de robustez MPC lento con nuevo escalado.

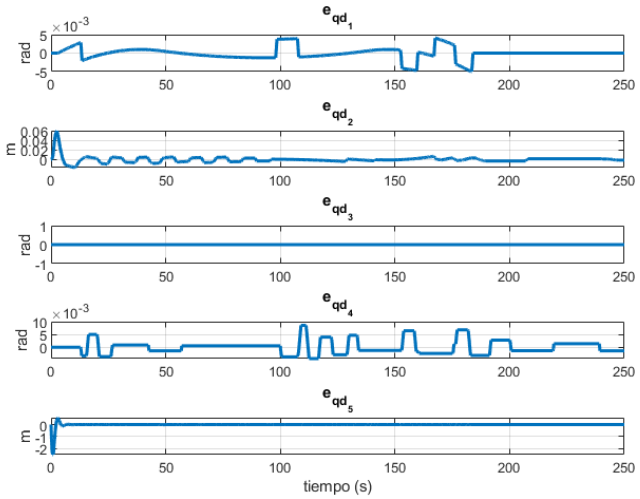


Figura 5.138 Errores en las velocidades articulaciones. Análisis de robustez MPC lento con nuevo escalado.

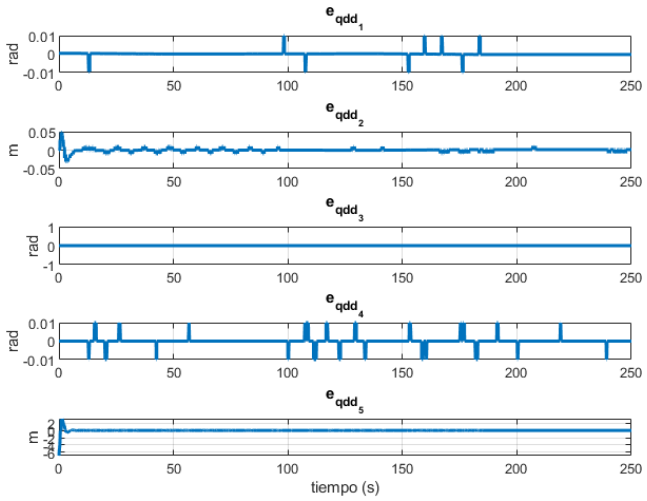


Figura 5.139 Errores en las aceleraciones articulaciones. Análisis de robustez MPC lento con nuevo escalado.

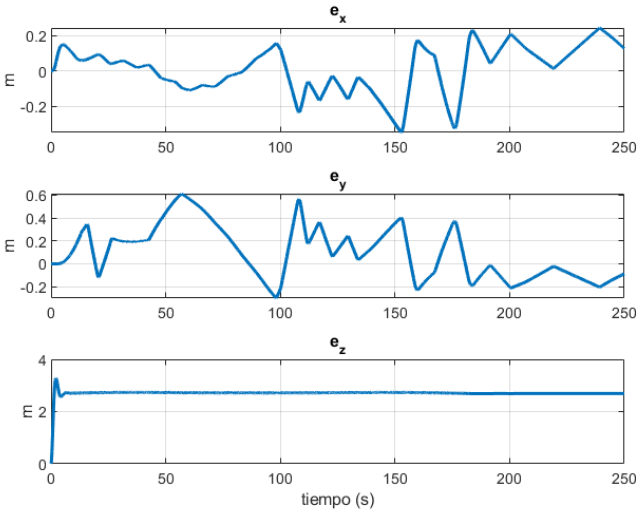


Figura 5.140 Errores cartesianos. Análisis de robustez MPC lento con nuevo escalado.

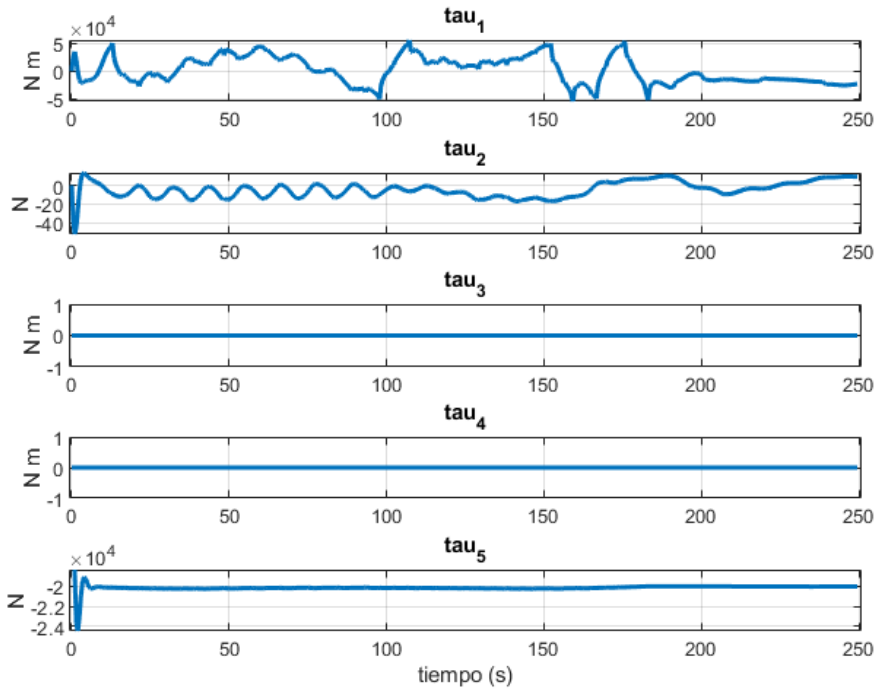


Figura 5.141 Señales de control. Análisis de robustez MPC lento con nuevo escalado.

Se observa que el problema de ser tan lento es que no es capaz de recuperarse del momento inicial y se mantiene un error prácticamente constante en la variable q_5.

Robustez al cambio de trayectoria

A pesar de la robustez ante cambios en la dinámica del sistema, todas las pruebas se han considerado siguiendo la misma trayectoria.

Los análisis de robustez y de perturbaciones permanecen invariantes ante modificaciones en la trayectoria; no obstante, el trabajo del control de seguimiento de referencia sí puede verse afectado, ya que la dinámica del péndulo varía según el sentido del movimiento (ascenso o descenso). A diferencia del método anterior en la generación de la trayectoria, basado en la interpolación de variables articulares a través de cinco puntos definidos, en esta propuesta se optará por una trayectoria rectilínea descendente.

La trayectoria en este caso partirá del punto (17,5 0 30) hacia el (-10 -15 10), cuyo movimiento comenzará a los 5 segundos, durará 200 segundos y debe pasar exactamente por 8 puntos.
Para esta casuística, se probarán los controladores LQR y MPC. Para el MPC se usará el que fue modificado en el análisis de robustez.

- LQR

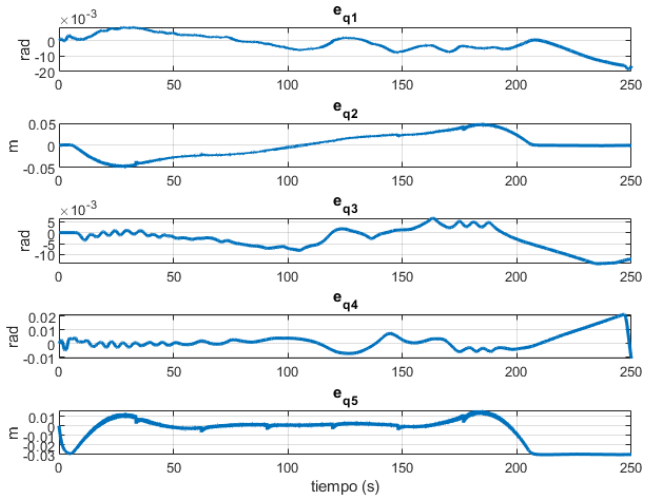


Figura 5.142 Errores en las articulaciones. Cambio de trayectoria LQR.

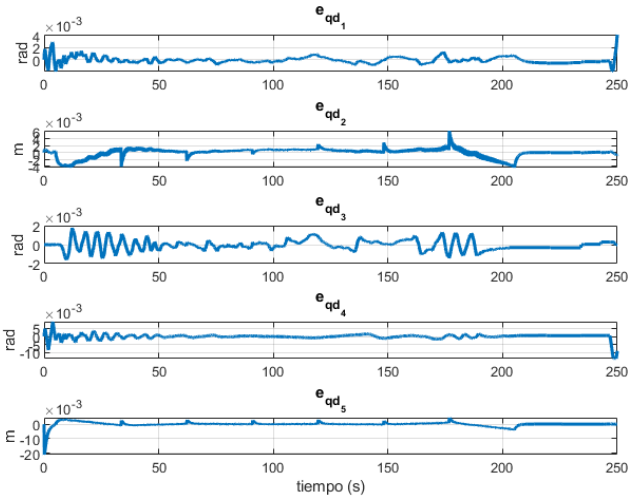


Figura 5.143 Errores en las velocidades articulaciones. Cambio de trayectoria LQR.

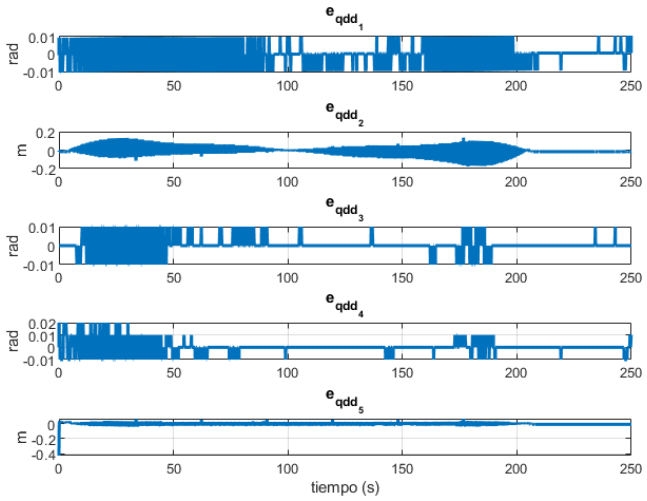


Figura 5.144 Errores en las aceleraciones articulaciones. Cambio de trayectoria LQR.

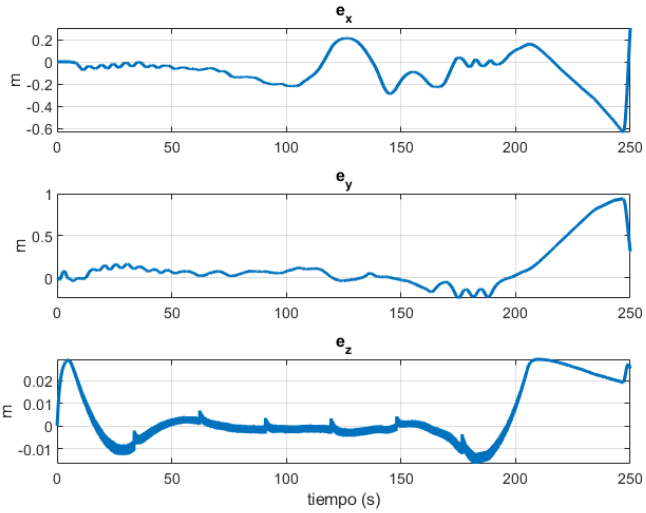


Figura 5.145 Errores cartesianos. Cambio de trayectoria LQR.

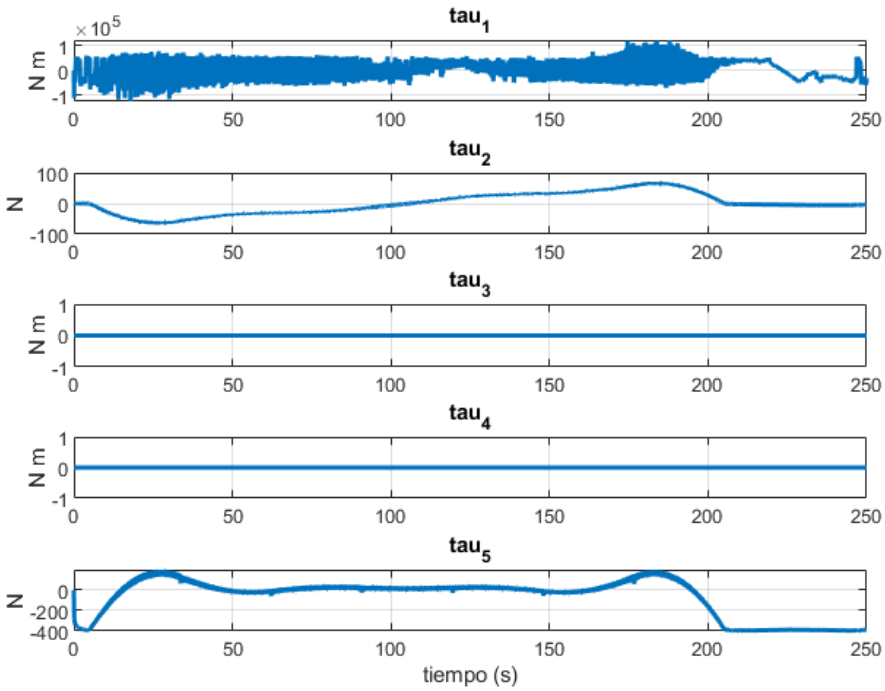


Figura 5.146 Señales de control. Cambio de trayectoria LQR.

Se observa que la trayectoria la sigue perfectamente con errores muy bajos y muy pocas oscilaciones en el péndulo, incluso mejor que el movimiento de subida. La señal de control de q_1, sin embargo, sigue oscilando mucho al igual que en la otra trayectoria.

- LQR lento

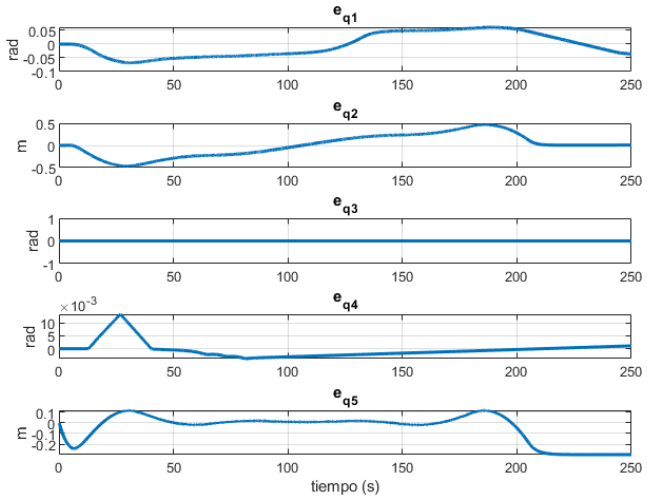


Figura 5.147 Errores en las articulaciones. Cambio de trayectoria LQR lento.

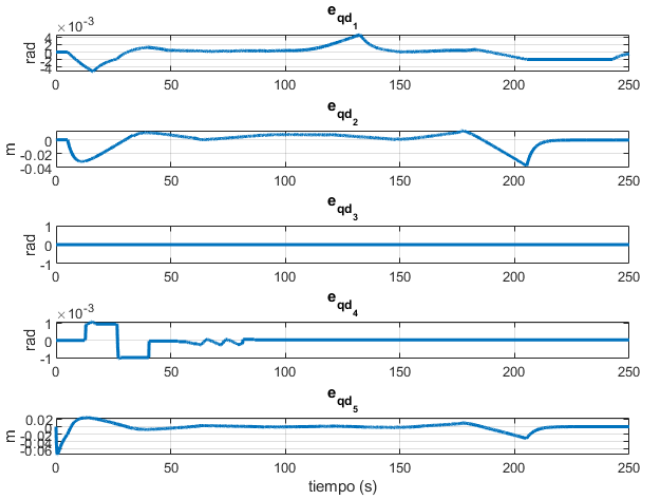


Figura 5.148 Errores en las velocidades articulaciones. Cambio de trayectoria LQR lento.

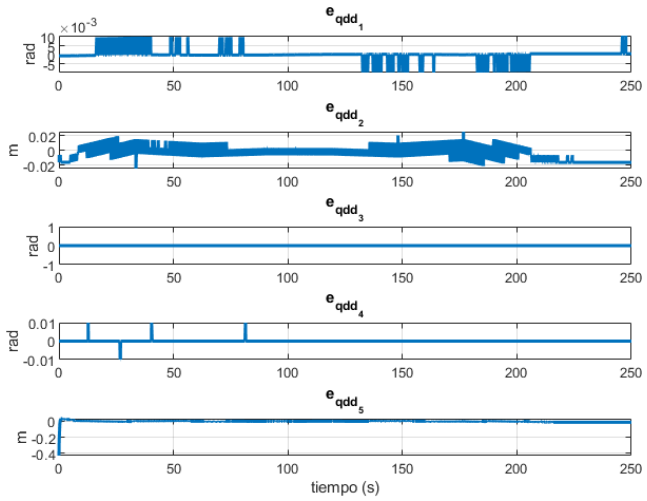


Figura 5.149 Errores en las aceleraciones articulaciones. Cambio de trayectoria LQR lento.

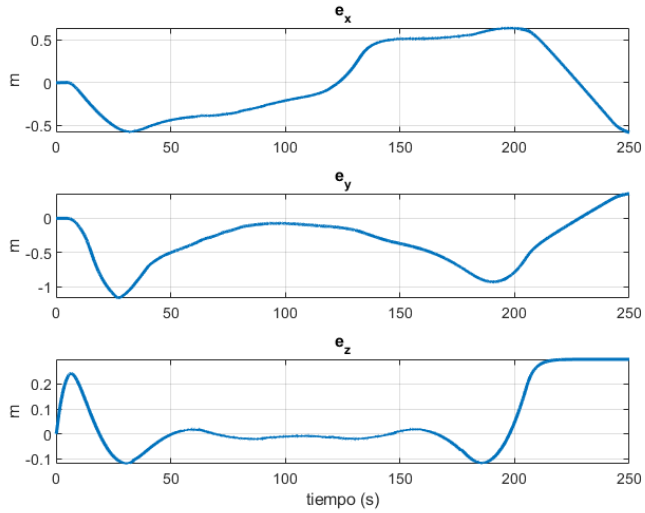


Figura 5.150 Errores cartesianos. Cambio de trayectoria LQR lento.

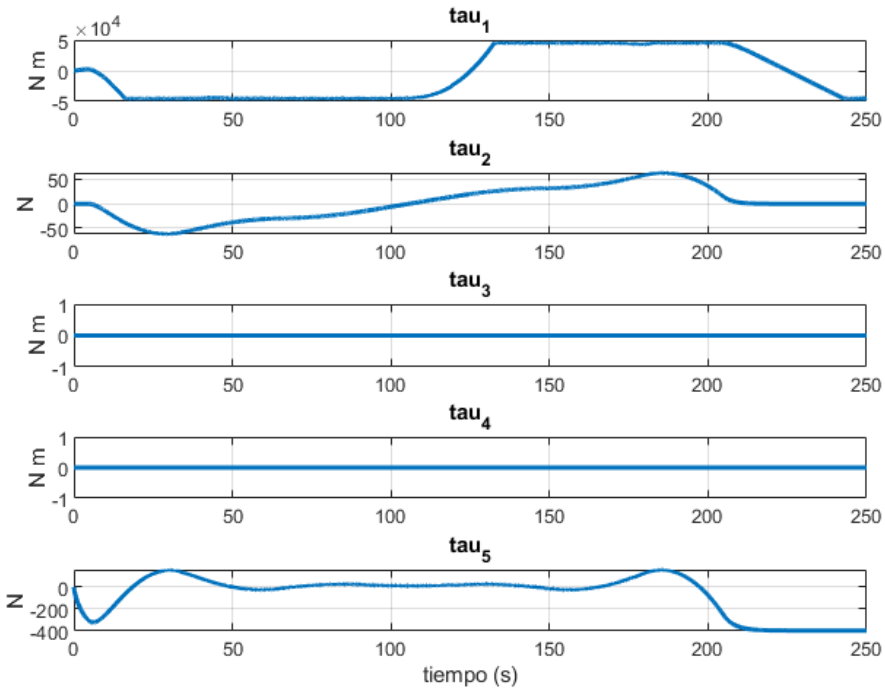


Figura 5.151 Señales de control. Cambio de trayectoria LQR lento.

Los resultados son buenos pero ligeramente de menor calidad que los del LQR previo. La articulación q_1 sufre una saturación debido a Bryson que hace que su error y el de las variables cartesianas aumente en el último tercio del movimiento.

- MPC robusto

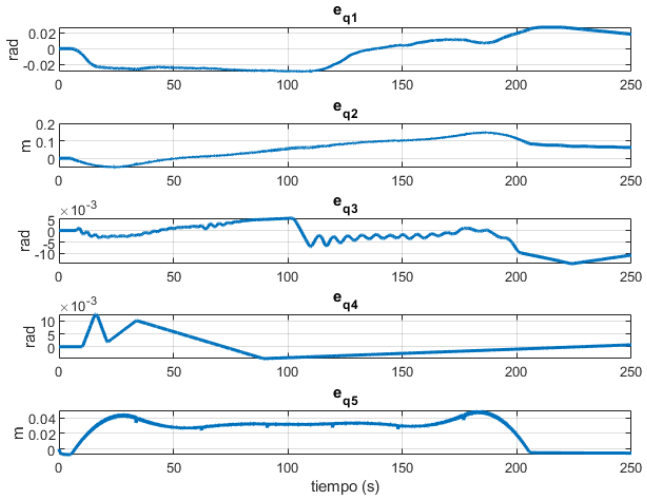


Figura 5.152 Errores en las articulaciones. Cambio de trayectoria MPC.

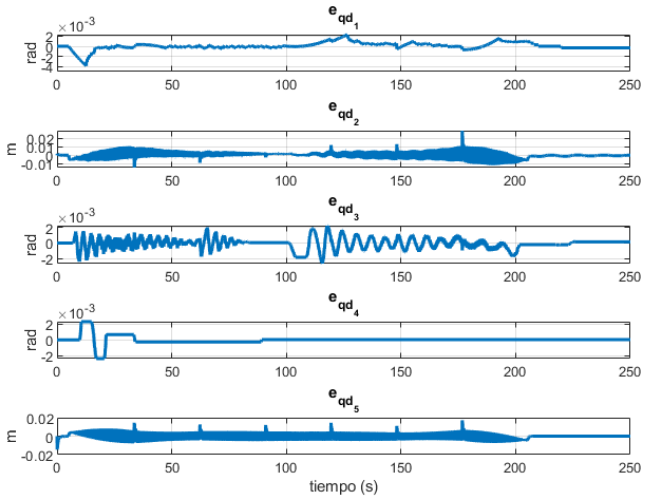


Figura 5.153 Errores en las velocidades articulaciones. Cambio de trayectoria MPC.

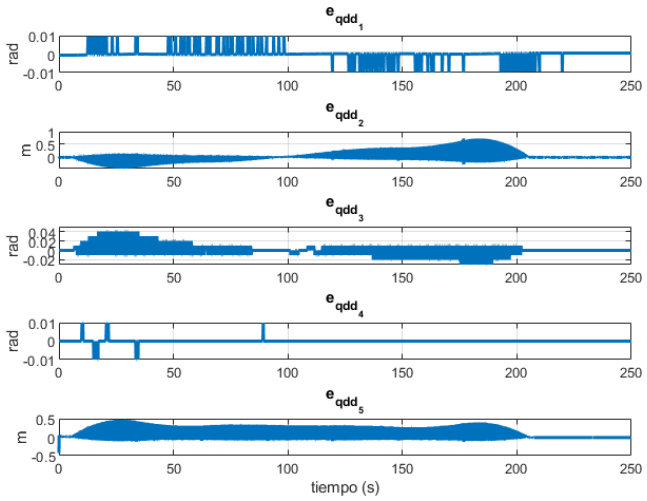


Figura 5.154 Errores en las aceleraciones articulaciones. Cambio de trayectoria MPC.

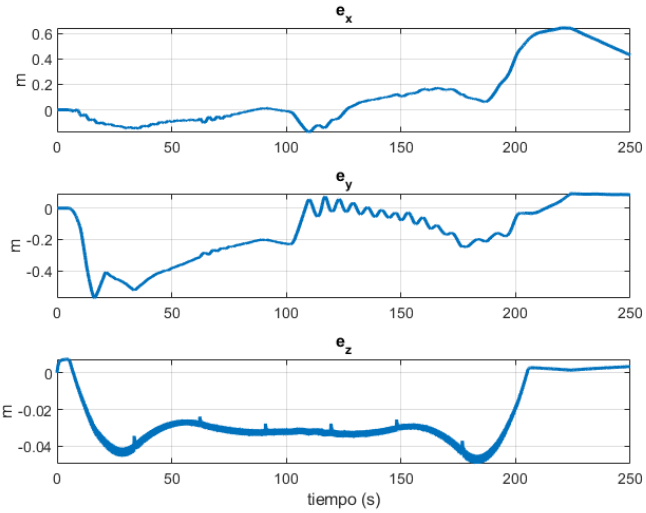


Figura 5.155 Errores cartesianos. Cambio de trayectoria MPC.

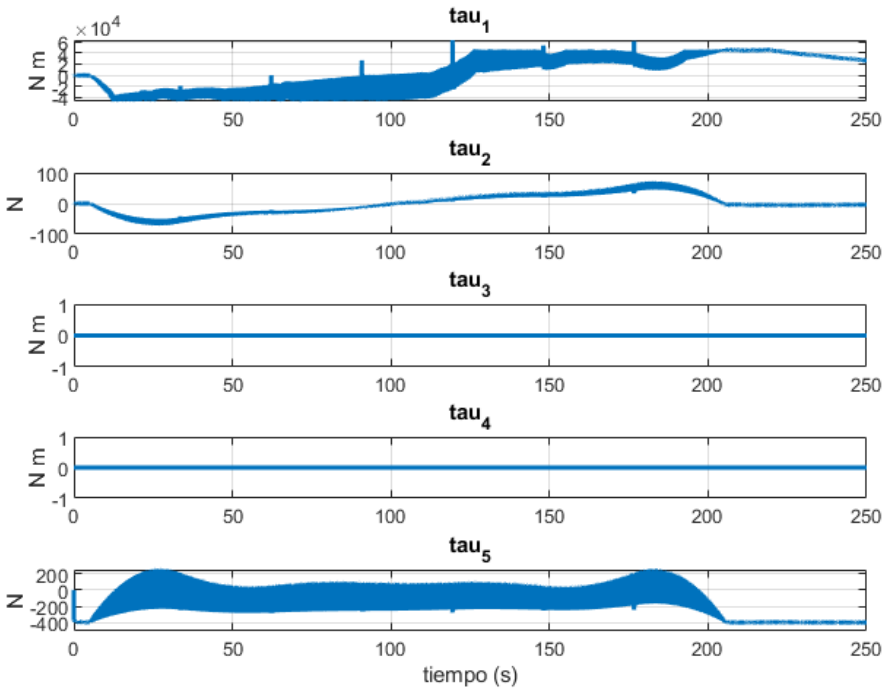


Figura 5.156 Señales de control. Cambio de trayectoria MPC.

Los errores en el péndulo se reducen ligeramente, mientras que los cartesianos y los de las variables articulares se mantienen semejantes, todo en comparación al LQR. Sin embargo, la señal de q_1 también oscila, aunque en menor medida ya que también es de menor orden y la de q_5 sí oscila mucho más.

- MPC lento robusto

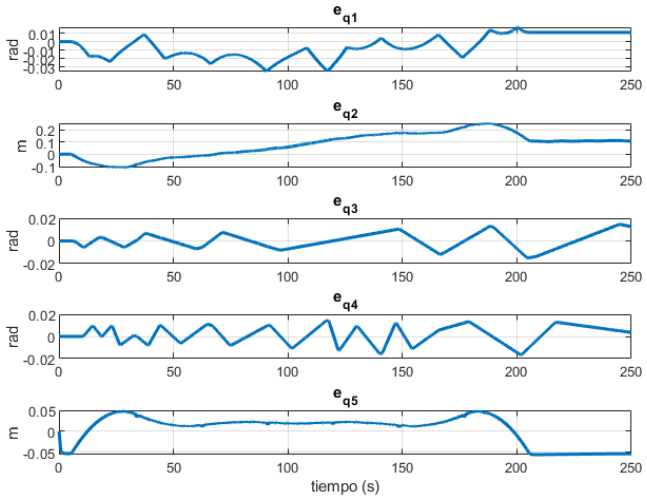


Figura 5.157 Errores en las articulaciones. Cambio de trayectoria MPC lento.

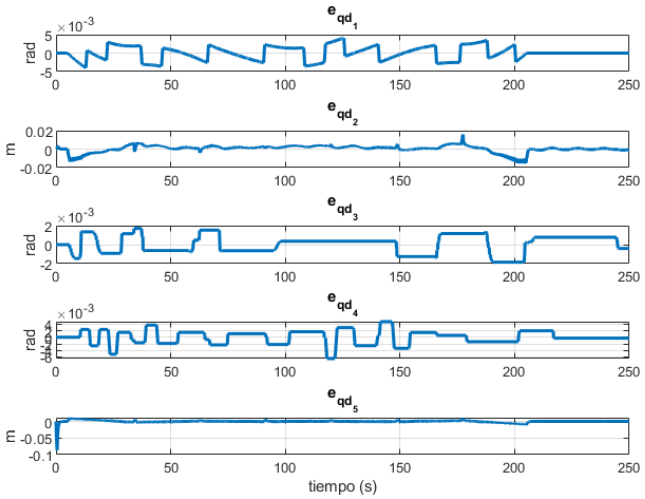


Figura 5.158 Errores en las velocidades articulaciones. Cambio de trayectoria MPC lento.

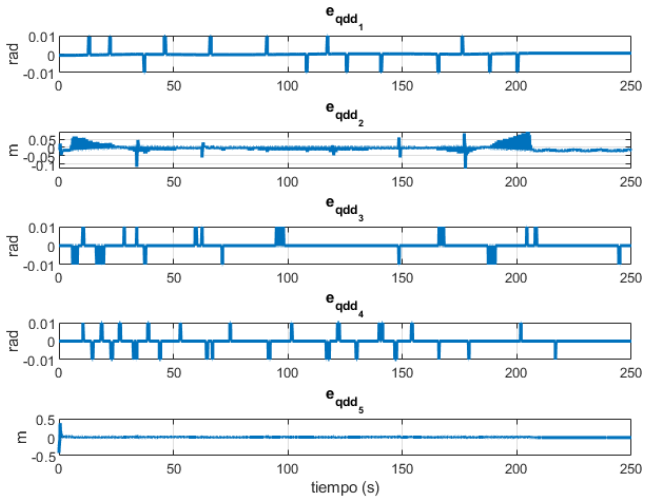


Figura 5.159 Errores en las aceleraciones articulaciones. Cambio de trayectoria MPC lento.

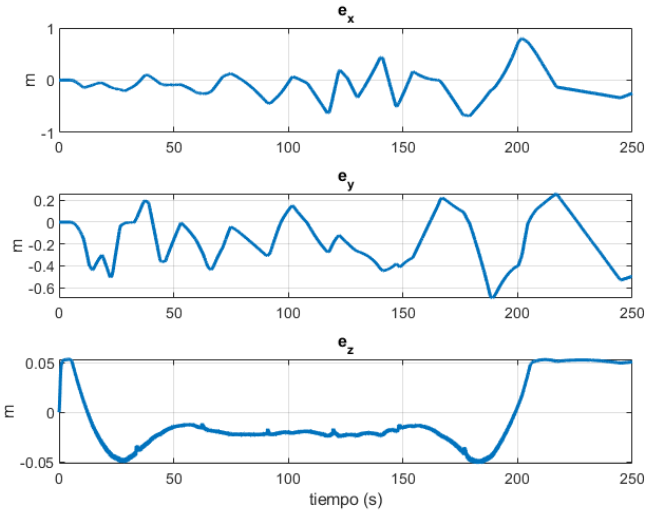


Figura 5.160 Errores cartesianos. Cambio de trayectoria MPC lento.

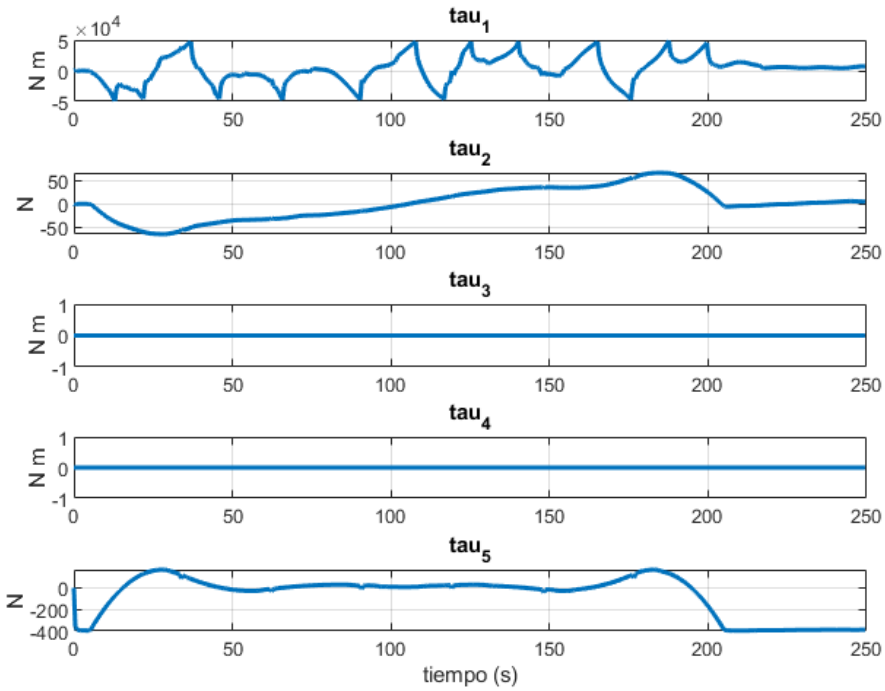


Figura 5.161 Señales de control. Cambio de trayectoria MPC lento.

Finalmente, este control tal y como en otras ocasiones, aumenta ligeramente los errores en las articulaciones y en las coordenadas cartesianas, pero a cambio reduce cualquier oscilación en las señales de control haciéndolas mucho más estables.

6 Conclusión

A modo de conclusión, se realizará un recorrido por cada uno de los apartados tratados para evaluarlos, sobre todo la parte de control dinámico, ya que deben definirse los puntos fuertes y debilidades de cada una de las estrategias empleadas.

La cinemática directa del sistema no tiene mucha complicación, gracias a que desde la tabla de Denavit-Hartenberg se extraen directamente las matrices de transformación desde la base hasta el extremo final; sin embargo, la cinemática inversa es más compleja, ya que pueden existir varias disposiciones y no existe un método mecánico para su extracción.
El modelo diferencial se obtiene de forma directa y la verdadera complicación aparece a la hora de las configuraciones de los puntos singulares. Al encontrarse en planos de varias variables no es posible extraer matemáticamente la mayoría de las soluciones.

Referente al modelo dinámico, el modelo se considera bueno a falta de probar los controladores en un sistemas real. Se puede observar en las diferentes simulaciones que el modelo dinámico actúa de forma lógica en general, principalmente describiendo las oscilaciones del péndulo bien en función de la longitud del mismo y del movimiento de las demás articulaciones.
Las matrices del modelo han salido bastante grandes y a simple vista no se puede analizar pero es consecuencia de realizar un modelo relativamente fiel con acoplamiento entre las diferentes variables.

Finalmente, se han utilizado los controladores SISO (PD y PID) principalmente para comprobar que las oscilaciones del péndulo no se tienen en cuenta en ningún momento y, por tanto, aunque se siga la referencia de las variables actuadas de forma precisa, el acoplamiento hace que estas oscilaciones aparezcan. Aún así, si se controla la trayectoria y la velocidad de dichas variables, se pueden minimizar las oscilaciones y emplearlas.
En cambio, si hay perturbaciones externas, como las del viento, que han sido utilizadas en este proyecto, estos controles no son capaces de rechazar las que afectan al péndulo. Esto los hace inservibles en este tipo de sistemas

En cuanto a los controladores MIMO, si se sintonizan adecuadamente, el seguimiento de trayectorias lo realizan con errores ligeramente mayores que los SISO. El mayor problema en este sistema es claramente la diferencia de orden entre las señales de control, lo que conlleva un escalado, ya sea a través de la regla de Bryson en el LQR o dentro de la propia función de Matlab en el MPC. Esto se debe a que, aunque se conozca el modelo, los pesos para las variables suelen ser del mismo orden, lo que genera conflictos con las articulaciones que necesitan mayor fuerza para su movimiento, como es el caso de q_1, el giro de la grúa.

Las perturbaciones del péndulo el MPC las ha sabido corregir de forma bastante buena gracias a su horizonte de predicción, cosa que el LQR no ha sido capaz ya que no es capaz de preveer el movimiento. Aún así las perturbaciones de las variables actuadas el LQR si las rechaza de forma rápida mientras que el MPC al no poder predecirlo no logra eliminarlas.

Por último, la robustez de los sistemas MIMO es bastante buena, aunque para el MPC ha habido que reajustar el escalado de la variable q_5, aunque se utilice el modelo que no tiene en cuenta que exista carga colgada.

Principalmente, se debe a los límites de fuerza que se le habían impuesto a la articulación, los cuales impedían que pudiese ejercer más fuerza y levantar la carga añadida. El LQR, a pesar de que la ganancia calculada no tenía en cuenta unos límites tan grandes en dicha variable, a la hora de usar Bryson sí se ha adaptado bien.

Como conclusión, el LQR es más robusto y realiza mejor el seguimiento; sin embargo, el hecho de que no sea capaz de rechazar las oscilaciones del péndulo es un punto muy en su contra, por lo que se debería apostar por perfeccionar el MPC. Además, tras todas las pruebas realizadas, se pueden fijar saturaciones en los actuadores que sirvan para cualquier masa colgada del cable que soporte la grúa. tau_1 se limitará entre $\pm 5 \times 10^5$ o $\pm 5 \times 10^6$ si se requiere un control más rápido, mientras que tau_2 saturará entre ± 50 y tau_5 en ± 50000.
Como trabajo futuro con el que seguir con este proyecto, se plantea realizar un control basado en NMPC, es decir, un MPC adaptativo en función del punto de operación para que sea algo más preciso e intentar que rechace las perturbaciones con más rapidez.

Apéndice A
Apéndice de códigos

A.1 Códigos de funciones auxiliares en Matlab

Código A.1 Giro en el eje Z.

```
function Rz= rotz(alfa)

Rz=[cos(alfa) -sin(alfa) 0     0
    sin(alfa)  cos(alfa) 0     0
       0          0      1     0
       0          0      0     1];
end
```

Código A.2 Traslación del sistema de referencia.

```
function D= trast(x,y,z)

D=[1   0   0     x
   0   1   0     y
   0   0   1     z
   0   0   0     1];
end
```

Código A.3 Giro en el eje X.

```
function Rx= rotx(alfa)

Rx=[1    0          0     0
    0  cos(alfa) -sin(alfa) 0
    0  sin(alfa) cos(alfa) 0
    0    0          0     1];
end
```

Código A.4 Función MDH.

```
function A= MDH(theta,d,a,alfa)

A=rotz(theta)*trast(0,0,d)*trast(a,0,0)*rotx(alfa);
```

```
end
```

Código A.5 Giro en el eje Y.

```
function Ry= roty(alfa)

Ry=[cos(alfa) 0 sin(alfa) 0
      0     1    0     0
   -sin(alfa) 0 cos(alfa) 0
   0    0         0     1];
end
```

Código A.6 Implementación del Teorema de Steiner.

```
function I=Steiner(vg,Io,m)
% vg debe ser un vector fila de 3 componentes indicando la traslación del
% punto donde se ha calculado Io hacia el punto donde se quiere saber I. Io
% será el tensor de inercia 3x3 calculado en un punto conocido. m será la
% masa correspondiente al elemento en el q se ha calculado Io.

xg=vg(1,1);
yg=vg(1,2);
zg=vg(1,3);

A=[yg^2+zg^2   -xg*yg    -xg*zg;
   -xg*yg     xg^2+zg^2   -yg*zg;
   -xg*zg     -yg*zg    xg^2+yg^2];

I= Io + m* A;

end
```

A.2 Códigos de la cinemática del sistema

Código A.7 Cálculo de la Cinemática directa simbólica del sistema.

```
function [x, y, z, phi, theta, psi, TB6]= CinematicaDirSim()

syms L0 L1A L2 q1 q2 q3 q4 q5 real       % Declaración de las variables simbó
    licas

PI=sym(pi);

[theta0, d0, a0, alpha0]=deal(0, L0, 0, 0);

[theta1, d1, a1, alpha1]=deal(q1+PI/2, 0, 0, PI/2);

[theta2, d2, a2, alpha2]=deal(-PI/2, L1A+q2, 0, PI/2);

[theta3, d3, a3, alpha3]=deal(q3, 0, 0, PI/2);

[theta4, d4, a4, alpha4]=deal(q4+PI/2, 0, 0, PI/2);
```

```
[theta5, d5, a5, alpha5]=deal(0, q5, 0, 0);

[theta6, d6, a6, alpha6]=deal(0, L2, 0, 0);

TB0= MDH(theta0, d0, a0, alpha0);

%{
 [1, 0, 0, 0]
 [0, 1, 0, 0]
 [0, 0, 1, L0]
 [0, 0, 0, 1]
%}

T01= MDH(theta1, d1, a1, alpha1);

%{
 [cos(q1 + pi/2), 0, sin(q1 + pi/2), 0]
 [sin(q1 + pi/2), 0, -cos(q1 + pi/2), 0]
 [            0, 1,              0, 0]
 [            0, 0,              0, 1]
%}

T12= MDH(theta2, d2, a2, alpha2);

%{
 [ 0, 0, -1,     0]
 [-1, 0, 0,      0]
 [ 0, 1, 0, L1A + q2]
 [ 0, 0, 0,      1]
%}


T23= MDH(theta3, d3, a3, alpha3);

%{
 [cos(q3), 0, sin(q3), 0]
 [sin(q3), 0, -cos(q3), 0]
 [     0, 1,       0, 0]
 [     0, 0,       0, 1]
%}

T34= MDH(theta4, d4, a4, alpha4);

%{
 [cos(q4 + pi/2), 0, sin(q4 + pi/2), 0]
 [sin(q4 + pi/2), 0, -cos(q4 + pi/2), 0]
 [            0, 1,              0, 0]
 [            0, 0,              0, 1]
%}

T45= MDH(theta5, d5, a5, alpha5);

%{
 [1, 0, 0, 0]
 [0, 1, 0, 0]
 [0, 0, 1, q5]
```

```
 [0, 0, 0, 1]
%}

T56= MDH(theta6, d6, a6, alpha6);

%{
 [1, 0, 0, 0]
 [0, 1, 0, 0]
 [0, 0, 1, L2]
 [0, 0, 0, 1]
%}

TB6= simplify(TB0*T01*T12*T23*T34*T45*T56);

%{
[  cos(q4)*sin(q1) - cos(q1)*sin(q3)*sin(q4), -cos(q1)*cos(q3), sin(q1)*sin(q4)
     + cos(q1)*cos(q4)*sin(q3), cos(q1)*(L1A + q2) + (L2 + q5)*(sin(q1)*sin(q4)
     + cos(q1)*cos(q4)*sin(q3))]
[- cos(q1)*cos(q4) - sin(q1)*sin(q3)*sin(q4), -cos(q3)*sin(q1), cos(q4)*sin(q1)
    *sin(q3) - cos(q1)*sin(q4), sin(q1)*(L1A
 + q2) - (L2 + q5)*(cos(q1)*sin(q4) - cos(q4)*sin(q1)*sin(q3))]
[                          cos(q3)*sin(q4),       -sin(q3),
                          -cos(q3)*cos(q4),                                 L0 -
    cos(q3)*cos(q4)*(L2 + q5)]
[                                        0,              0,
                                         0,
                                                          1]
%}


x= cos(q1)*(L1A + q2) + (L2 + q5)*(sin(q1)*sin(q4) + cos(q1)*cos(q4)*sin(q3));

y= sin(q1)*(L1A + q2) - (L2 + q5)*(cos(q1)*sin(q4) - cos(q4)*sin(q1)*sin(q3));

z= L0 - cos(q3)*cos(q4)*(L2 + q5);

%% Ángulos de Euler

% MRPY=rotz(phi)*roty(theta)*rotx(psi);

% [cos(phi)*cos(theta), cos(phi)*sin(psi)*sin(theta) - cos(psi)*sin(phi), sin(
    phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta), 0]
% [cos(theta)*sin(phi), cos(phi)*cos(psi) + sin(phi)*sin(psi)*sin(theta), cos(
    psi)*sin(phi)*sin(theta) - cos(phi)*sin(psi), 0]
% [       -sin(theta),                          cos(theta)*sin(psi),
                           cos(psi)*cos(theta), 0]
% [                 0,                                            0,
                                       0, 1]

% Se compara con TB6

r11= TB6(1,1);    r12= TB6(1,2);    r13= TB6(1,3);
r21= TB6(2,1);    r22= TB6(2,2);    r23= TB6(2,3);
r31= TB6(3,1);    r32= TB6(3,2);    r33= TB6(3,3);

theta = atan2(-r31,sqrt(r11^2+r21^2));
```

```
phi = atan2(r21,r11);

psi = atan2(r32,r33);

end
```

Código A.8 Cálculo de la Cinemática inversa simbólica del sistema.

```
function [q1, q2, q3, q4, q5]= CinematicaInvSim()

syms L0 L1A L2 x y z theta psi_sym phi real       % Declaración de las
    variables simbólicas

PI=sym(pi);

% Cinemática inversa

% Se van a utilizar las ecuaciones obtenidas a la hora de calcular los
% ángulos de Euler para conseguir la definición de las variables
% articulares. Se seguirá un procedimiento parecido al anterior.

MRPY=rotz(phi)*roty(theta)*rotx(psi_sym);

% [cos(phi)*cos(theta), cos(phi)*sin(psi_sym)*sin(theta) - cos(psi_sym)*sin(phi
    ), sin(phi)*sin(psi_sym) + cos(phi)*cos(psi_sym)*sin(theta), 0]
% [cos(theta)*sin(phi), cos(phi)*cos(psi_sym) + sin(phi)*sin(psi_sym)*sin(theta
    ), cos(psi_sym)*sin(phi)*sin(theta) - cos(phi)*sin(psi_sym), 0]
% [       -sin(theta),                           cos(theta)*sin(psi_sym),
                          cos(psi_sym)*cos(theta), 0]
% [                 0,                                                 0,
                                       0, 1]

% Se compara con la matriz TB6 obtenida en la cinemática directa.

r11= MRPY(1,1);    r12= MRPY(1,2);    r13= MRPY(1,3);
r21= MRPY(2,1);    r22= MRPY(2,2);    r23= MRPY(2,3);
r31= MRPY(3,1);    r32= MRPY(3,2);    r33= MRPY(3,3);

%%%%% q3 %%%%%
% -sin(q3)= r32
% sin(q3)= -r32

% cos(q3)*sin(q4)= r31
% -cos(q3)*cos(q4)= r33

% r31^2 + r33^2 = 2*cos(q3)^2
% cos(q3) = sqrt((r31^2 + r33^2)/2)
% cos(q3) = -sqrt((r31^2 + r33^2)/2)

q3 = atan2(-r32, sqrt((r31^2 + r33^2)));

%%%%% q4 %%%%%
% cos(q3)*sin(q4)= r31
% sin(q4)*cos(q3) = r31

% -cos(q3)*cos(q4)= r33
```

```
% cos(q4)*cos(q3) = -r33

% (sin(q4)*cos(q3)) / (cos(q4)*cos(q3)) = r31/-r33
% sin(q4)/cos(q4) = r31/-r33

q4 = atan2(r31,-r33);

%%%% q1 %%%%
% -cos(q1)*cos(q3) = r12
% cos(q1)*cos(q3) = -r12

% -cos(q3)*sin(q1) = r22
% sin(q1)*cos(q3) = -r22

% (sin(q1)*cos(q3)) / (cos(q1)*cos(q3)) = -r22/-r12
% sin(q1)/cos(q1) = -r22/-r12

q1 = atan2(-r22,-r12);

% Una vez se ha solucionado la parte de la orientación gracias al desacoplo
% que se puede hacer del robot pasaremos a obtener las variales restantes
% utilizando las ecuaciones de posición

%%%% q5 %%%%
% z= L0 - cos(q3)*cos(q4)*(L2 + q5);
% -(z-L0)/ cos(q3)*cos(q4) = L2+q5
% q5 = (L0-z)/cos(q3)*cos(q4)

q5 = (L0-z)/(cos(q3)*cos(q4)) -L2;

%%%% q2 %%%%
% x= cos(q1)*(L1A + q2) + (L2 + q5)*(sin(q1)*sin(q4) + cos(q1)*cos(q4)*sin(q3))
    ;
% x= cos(q1)*(L1A + q2) + (L2 + q5)*r13;
% x - (L2 + q5)*r13 = cos(q1)*(L1A + q2);
% (x - (L2 + q5)*r13)*cos(q1) = cos(q1)^2*(L1A + q2);

% y= sin(q1)*(L1A + q2) - (L2 + q5)*(cos(q1)*sin(q4) - cos(q4)*sin(q1)*sin(q3))
    ;
% y= sin(q1)*(L1A + q2) + (L2 + q5)*r23;
% (y - (L2 + q5)*r23)*sin(q1) = sin(q1)^2*(L1A + q2);

% sin(q1)^2*(L1A + q2) + cos(q1)^2*(L1A + q2) = (x - (L2 + q5)*r13)*cos(q1) + (
    y - (L2 + q5)*r23)*sin(q1)
% (L1A + q2) = (x - (L2 + q5)*r13)*cos(q1) + (y - (L2 + q5)*r23)*sin(q1)
% q2 = (x - (L2 + q5)*r13)*cos(q1) + (y - (L2 + q5)*r23)*sin(q1) -L1A

q2 = (x - (L2 + q5)*r13)*cos(q1) + (y - (L2 + q5)*r23)*sin(q1) -L1A;

end
```

Código A.9 Cálculo de la Cinemática directa numérica del sistema.

```
function [xyzangEuler] = CinematicaDir(in)
  q1      = in(1);          % Posiciones articulares
  q2      = in(2);
```

```
  q3      = in(3);
  q4      = in(4);
  q5      = in(5);

L0= 45; % m
L1B = 35; % m
L1A= L1B/2; % m
L2= 15; % m

x= cos(q1)*(L1A + q2) + (L2 + q5)*(sin(q1)*sin(q4) + cos(q1)*cos(q4)*sin(q3));

y= sin(q1)*(L1A + q2) - (L2 + q5)*(cos(q1)*sin(q4) - cos(q4)*sin(q1)*sin(q3));

z= L0 - cos(q3)*cos(q4)*(L2 + q5);

xyz=[x;y;z];

TB6= [ cos(q4)*sin(q1) - cos(q1)*sin(q3)*sin(q4), -cos(q1)*cos(q3), sin(q1)*
    sin(q4) + cos(q1)*cos(q4)*sin(q3), L1A*cos(q1) + q2*cos(q1) + L2*sin(q1)*
    sin(q4) + q5*sin(q1)*sin(q4) + L2*cos(q1)*cos(q4)*sin(q3) + q5*cos(q1)*cos(
    q4)*sin(q3);
      - cos(q1)*cos(q4) - sin(q1)*sin(q3)*sin(q4), -cos(q3)*sin(q1), cos(q4)*
          sin(q1)*sin(q3) - cos(q1)*sin(q4), L1A*sin(q1) + q2*sin(q1) - L2*cos(
          q1)*sin(q4) - q5*cos(q1)*sin(q4) + L2*cos(q4)*sin(q1)*sin(q3) + q5*
          cos(q4)*sin(q1)*sin(q3);
                                cos(q3)*sin(q4),       -sin(q3),
                                                   -cos(q3)*cos(q4),

                                    L0 - L2*cos(q3)*cos(q4) - q5*cos(q3)*cos(q4)
                                    ;
                                             0,              0,

                                                 0,

                                                 1];

r11= TB6(1,1);    r12= TB6(1,2);    r13= TB6(1,3);
r21= TB6(2,1);    r22= TB6(2,2);    r23= TB6(2,3);
r31= TB6(3,1);    r32= TB6(3,2);    r33= TB6(3,3);

theta = atan2(-r31,sqrt(r11^2+r21^2));

phi = atan2(r21,r11);

psi = atan2(r32,r33);

angEuler = [phi;theta;psi];

xyzangEuler= [xyz;angEuler];

end
```

Código A.10 Cálculo de la Cinemática inversa numérica del sistema.

```
function [q,fueraRango] = CinematicaInv(pose)
x = pose(1);          % Posición cartesianas
```

```
y = pose(2);          %
z = pose(3);          %
phi = pose(4);        % Ángulos RPY
theta = pose(5);      %
psi = pose(6); %
fueraRango = false;  % Suponemos inicialmente que existe solución

    L0= 45; % m
    L1B = 35; % m
    L1A= L1B/2; % m
    L2= 15; % m

MRPY=rotz(phi)*roty(theta)*rotx(psi);

% Se compara con la matriz TB6 obtenida en la cinemática directa.

r11= MRPY(1,1);    r12= MRPY(1,2);    r13= MRPY(1,3);
r21= MRPY(2,1);    r22= MRPY(2,2);    r23= MRPY(2,3);
r31= MRPY(3,1);    r32= MRPY(3,2);    r33= MRPY(3,3);

q3 = atan2(-r32, sqrt(r31^2 + r33^2));
q4 = atan2(r31,-r33);
q1 = atan2(-r22,-r12);

% Analizamos los límites de las articulaciones

if (abs(q3)>= pi/3)
    fueraRango = true;
end

if (abs(q4)>= pi/3)
    fueraRango = true;
end

% Se desplaza q1 para que opte por valores entre
% pi y -pi sin modificar la posición del extremo

 if q1<-pi
    q1= 2*pi+q1;

 elseif q1>pi
    q1= q1-2*pi;
 end

% Será necesario evitar la indeterminación del denominador de q5, sin
% embargo, como q3 y q4 se acotan entre +-pi/3 esto no ocurrirá nunca

if fueraRango == false
    q5 = (L0-z)/(cos(q3)*cos(q4)) -L2;

% Analizamos los límites de la articulación

    if (q5>= 4*(L0-L2)/5 || q5<= -4*L2/5)
        fueraRango = true;
    end

end
```

```
if fueraRango == false

    inter1_q2= x-(L2+q5)*r13;
    inter2_q2= y-(L2+q5)*r23;

    q2= inter1_q2*cos(q1) + inter2_q2*sin(q1) - L1A;

% Analizamos los límites de las articulaciones

    if (q2>= 4*(L1B-L1A)/5 || q2<= -4*L1A/5)
        fueraRango = true;
    end
end

 if (fueraRango == false)
    q = [q1; q2; q3; q4; q5];
    q = round(q,3);
 else
    q = zeros(5,1);
 end

end
```

Código A.11 Cálculo de trayectorias y comprobación del modelo cinemático.

```
function [Q_res]=planificador_grua_torre(opcion)

    animar = opcion;      % Elegir entre visualización de la trayectoria (
        cualquier otro valor) o animación (1)

    %% Configuración de la Trayectoria
    tipo = 'recta'; % Opciones: 'recta' o 'circulo'

    % Intervalos
    paso = 0.1;
    total = 20;
    intervalos = 0:paso:total;

    % Altura de operación constante para la prueba
    z_prueba = 20;

    %% 2. Generación de posición

    if strcmp(tipo, 'recta')
        x_ini = 5; y_ini = 20;
        x_f = 20;  y_f = 5;

        % Interpolación lineal
        x = linspace(x_ini, x_f, length(intervalos));
        y = linspace(y_ini, y_f, length(intervalos));
        z = ones(size(intervalos)) * z_prueba;

    elseif strcmp(tipo, 'circulo')
        % Trayectoria Circular
        radio = 10;
```

```
    omega = 0.3; % rad/s
    centro_x = 15;
    centro_y = 0;

    x = centro_x + radio * cos(omega * intervalos);
    y = centro_y + radio * sin(omega * intervalos);
    z = ones(size(intervalos)) * z_prueba;
end

%% 3. Cálculo de Orientación

Q_res = zeros(5, length(intervalos)); % resultado de las var. articulares
Pos_res = zeros(3, length(intervalos)); % Para comprobar con la directa
valid = true(1, length(intervalos)); % Indica si válido o no

% Calculamos la orientación
% se va a suponer que q3 y q4 no se mueven, solo cambia la orientación
% del yaw (phi). Inicialmente, (todas las articulaciones a 0), la
% orintación es phi=-pi/2, theta=0 y psi= -pi

phi = atan2(y, x)-pi/2;
theta = 0;
psi = -pi;

for i = 1:length(intervalos)

    pose_deseada = [x(i); y(i); z(i); phi(i); theta; psi];

    [q_sol, fueraRango] = CinematicaInv(pose_deseada);

    if fueraRango
        valid(i) = false;
        % Si falla, se pone todo a cero
            Q_res(:, i) = [0;0;0;0;0];
    else
        Q_res(:, i) = q_sol;
    end

    % Verificación con Cinemática Directa
    [xyz] = CinematicaDir(Q_res(:, i));
    Pos_res(:, i) = xyz(1:3);
end

%% 4 Visualización
% Trayectoria

if animar==1

    plot3(x, y, z, 'k--', 'LineWidth', 1.5); hold on;
    plot3(Pos_res(1, valid), Pos_res(2, valid), Pos_res(3, valid), 'ro', '
        MarkerSize', 1);

else
    figure('Name', 'Trayectoria de la Grúa', 'Color', 'w');
    plot3(x, y, z, 'k--', 'LineWidth', 1.5); hold on;
    plot3(Pos_res(1, valid), Pos_res(2, valid), Pos_res(3, valid), 'ro', '
        MarkerSize', 1);
```

```
        grid on; axis equal;
        xlabel('X [m]'); ylabel('Y [m]'); zlabel('Z [m]');
        legend('Trayectoria deseada', 'Trayectoria seguida');
        title(['Seguimiento de ' tipo]);
        view(45, 30);

        margen = 5;

        % límites de datos
        min_x = min(x); max_x = max(x);
        min_y = min(y); max_y = max(y);

        % límites + margen
        axis([min_x - margen, max_x + margen, ...
              min_y - margen, max_y + margen, ...
              0,              z_prueba + margen]);


        % Evolución de q
        figure('Name', 'Estados Articulares', 'Color', 'w');
        nombres_q = {'q1', 'q2', 'q3', 'q4', 'q5'};
        for k = 1:5
            subplot(5, 1, k);
            plot(intervalos, Q_res(k, :), 'LineWidth', 1.5);
            ylabel(nombres_q{k});
            grid on;
            if k==1, title('Evolución de las Articulaciones'); end
        end
        xlabel('Intervalos');

        if any(~valid)
            disp('ADVERTENCIA: Algunos puntos estuvieron fuera de rango (ver grá
                fica)');
        end


    end

end
```

Código A.12 Cálculo simbólico del Jacobiano directo e inverso.

```
function [Ja Jpi] = Modelo_Diferencial_simb
syms L0 L1A L2 q1 q2 q3 q4 q5 x y z phi theta psi_sym real     % Declaración de
     las variables simbólicas

PI=sym(pi);

 [x_1, y_1, z_1, phi_1, theta_1, psi_1, TB6]= CinematicaDirSim();


Ja=simplify([diff(x_1,q1) diff(x_1,q2) diff(x_1,q3) diff(x_1,q4) diff(x_1,q5);
    diff(y_1,q1) diff(y_1,q2) diff(y_1,q3) diff(y_1,q4) diff(y_1,q5);
    diff(z_1,q1) diff(z_1,q2) diff(z_1,q3) diff(z_1,q4) diff(z_1,q5);
    diff(phi_1,q1) diff(phi_1,q2) diff(phi_1,q3) diff(phi_1,q4) diff(phi_1,q5);
```

```
    diff(theta_1,q1) diff(theta_1,q2) diff(theta_1,q3) diff(theta_1,q4) diff(
        theta_1,q5);
    diff(psi_1,q1) diff(psi_1,q2) diff(psi_1,q3) diff(psi_1,q4) diff(psi_1,q5)
        ]);


[q1_1, q2_1, q3_1, q4_1, q5_1]= CinematicaInvSim();

Jpi= simplify([diff(q1_1,x) diff(q1_1,y) diff(q1_1,z) diff(q1_1,phi) diff(q1_1,
    theta) diff(q1_1,psi_sym); ...
              diff(q2_1,x) diff(q2_1,y) diff(q2_1,z) diff(q2_1,phi) diff(q2_1,
                  theta) diff(q2_1,psi_sym); ...
              diff(q3_1,x) diff(q3_1,y) diff(q3_1,z) diff(q3_1,phi) diff(q3_1,
                  theta) diff(q3_1,psi_sym); ...
              diff(q4_1,x) diff(q4_1,y) diff(q4_1,z) diff(q4_1,phi) diff(q4_1,
                  theta) diff(q4_1,psi_sym); ...
              diff(q5_1,x) diff(q5_1,y) diff(q5_1,z) diff(q5_1,phi) diff(q5_1,
                  theta) diff(q5_1,psi_sym)]);

end
```

Código A.13 Cálculo simbólico del determinante de la matriz Jacobiana y extracción de puntos singulares.

```
function [Deter_Ja,Sol] = Cal_conf_sing()
syms q1 q2 q3 q4 q5 real
[Ja Jpi] = Modelo_Diferencial_simb;

Ja_cuadrada=Ja'*Ja;

Deter_Ja= simplify(det(Ja_cuadrada), 'IgnoreAnalyticConstraints', true);

Sol= solve(Deter_Ja == 0, [q1 q2 q3 q4 q5]);

end
```

A.3 Código del modelo para la representación visual

Código A.14 Modelo cinemático de la grúa usando Robotics System Toolbox.

```
clc; close all;

% PARÁMETROS
L0  = 45;
L1B = 35;
L1A = L1B/2;
L2  = 15;

robot = rigidBodyTree('DataFormat','column');

% BASE    TB0 (Fija)
body0 = rigidBody('Base');

jnt0 = rigidBodyJoint('jnt0','fixed');
```

```
TB0 = [1 0 0 0; 0 1 0 0; 0 0 1 L0; 0 0 0 1];
setFixedTransform(jnt0, TB0);

addVisual(body0,"Cylinder",[0.6 L0], ...
         trvec2tform([0 0 -L0/2]));

body0.Joint = jnt0;
addBody(robot, body0,'base');



% T01    q1
body1 = rigidBody('PlumaRot');

jnt1 = rigidBodyJoint('jnt1','revolute');
jnt1.JointAxis = [0 1 0]; % eje Y

% Parte fija = rotación /2
T01_fixed = [cos(pi/2) 0 sin(pi/2) 0; sin(pi/2) 0 -cos(pi/2) 0; 0 1 0 0; 0 0 0
    1];
setFixedTransform(jnt1, T01_fixed);

jnt1.PositionLimits = [-pi pi];

body1.Joint = jnt1;

addVisual(body1,"Cylinder",[0.6 L1B], ...
         trvec2tform([0 0 L1B/2]));

addBody(robot, body1,'Base');

% T12    q2
body2 = rigidBody('Carrito');

jnt2 = rigidBodyJoint('jnt2','prismatic');
jnt2.JointAxis = [0 1 0];

T12_f = [ 0 0 -1 0;
        -1 0  0 0;
         0 1  0 L1A;
         0 0  0 1];

setFixedTransform(jnt2,T12_f);

jnt2.PositionLimits = [ -4*L1A/5 , 4*(L1B-L1A)/5 ];

body2.Joint = jnt2;

addVisual(body2,"Box",[2 2 1]);

addBody(robot, body2,'PlumaRot');

% T23    q3
body3 = rigidBody('CableRot1');

jnt3 = rigidBodyJoint('jnt3','revolute');
jnt3.JointAxis = [0 1 0];
```

```
T23_fixed = [1 0 0 0; 0 0 -1 0; 0 1 0 0; 0 0 0 1]; % T23 con q3=0
setFixedTransform(jnt3, T23_fixed);

jnt3.PositionLimits = [-pi/3 pi/3];

body3.Joint = jnt3;

addVisual(body3,"Sphere",0.6);

addBody(robot, body3,'Carrito');

% T34    q4
body4 = rigidBody('CableRot2');

jnt4 = rigidBodyJoint('jnt4','revolute');
jnt4.JointAxis = [0 1 0];

T34_fixed = [cos(pi/2) 0 sin(pi/2) 0; sin(pi/2) 0 -cos(pi/2) 0; 0 1 0 0; 0 0 0
    1];
setFixedTransform(jnt4, T34_fixed);

jnt4.PositionLimits = [-pi/3 pi/3];

body4.Joint = jnt4;

addVisual(body4,"Sphere",0.6, ...
         trvec2tform([0 0 1]));

addBody(robot, body4,'CableRot1');

% T45    q5
body5 = rigidBody('CableExt');

jnt5 = rigidBodyJoint('jnt5','prismatic');
jnt5.JointAxis = [0 0 1];

setFixedTransform(jnt5, eye(4));

jnt5.PositionLimits = [ -4*L2/5 , 4*(L0-L2)/5 ];

body5.Joint = jnt5;

L_cable = L2;
% addVisual(body5,"Cylinder",[0.2 L_cable], ...
%          trvec2tform([0 0 L_cable/2]));

addBody(robot, body5,'CableRot2');

% T56    GANCHO
body6 = rigidBody('Gancho');

jnt6 = rigidBodyJoint('jnt6','fixed');
setFixedTransform(jnt6, trvec2tform([0 0 L2]));

body6.Joint = jnt6;
```

```
addVisual(body6,"Box",[1 1 0.5]);

addBody(robot, body6,'CableExt');

%% CONFIGURACIÓN INICIAL
q_home = [0; 0; 0; 0; 0];

%% VISUALIZACIÓN CORRECTA
figure('Color','w');

show(robot,q_home,...
    'Frames','on', ...
    'Visuals','on');
patches = findobj(gca,'Type','Patch');

% Asignar colores
for k = 1:length(patches)
    patches(k).EdgeColor = 'none';
end

patches(1).FaceColor = [0.8 0.8 0.8];
patches(2).FaceColor = [0.8 0.8 0.8];
patches(3).FaceColor = [0.8 0.8 0.8];

patches(4).FaceColor = [1 0.9 0];
patches(5).FaceColor = [1 0.9 0];
patches(6).FaceColor = [1 0.9 0];
patches(7).FaceColor = [1 0.9 0];
patches(8).FaceColor = [1 0.9 0];
patches(9).FaceColor = [1 0.9 0];


hold on;
[X,Y] = meshgrid(-30:5:30);
Z = zeros(size(X));
surf(X,Y,Z,'FaceColor',[0.9 0.9 0.9],'EdgeColor','none')
axis equal
axis auto
grid on
view(140,25)
light('Position',[50 50 100],'Style','infinite')
material metal
lighting phong
title('Grúa torre 5GDL subactuada');
hold on;
cable_plot = plot3([0 0],[0 0],[0 0],'Color',[0.5 0.5 0.5],'LineWidth',3);

animar_cin=0;

if animar_cin==1
    % Animacion control cinemático grúa

    Q_res=planificador_grua_torre(1);

    for k = 1:length(Q_res)

        qk = Q_res(:,k); % columna
```

```
        % actualizar robot
        show(robot,qk,'PreservePlot',false,'Frames','off');

        % Posición del punto superior del cable
        T_top = getTransform(robot,qk,'CableRot2');
        pos_top = tform2trvec(T_top);

        % obtener posición del efector final
        T_ee = getTransform(robot,qk,'Gancho');
        pos = tform2trvec(T_ee);

        % Actualizar cable
        set(cable_plot,...
            'XData',[pos_top(1) pos(1)],...
            'YData',[pos_top(2) pos(2)],...
            'ZData',[pos_top(3) pos(3)]);

        drawnow;
    end
elseif animar_cin==0

    % Animacion control dinámico grúa

    plot3(xr, yr, zr,'--b','LineWidth',1.5)
    hold on;

    punto_ref = plot3(xr(1), yr(1), zr(1), 'ob', 'MarkerFaceColor', 'r', '
        MarkerSize', 7);

    trayectoria = animatedline('Color','r','LineWidth',2);

    salto = 50;

    for k = 1:salto:length(t)

        qk = q(k,:)'; % columna

        % actualizar robot
        show(robot,qk,'PreservePlot',false,'Frames','off');

        % Posición del punto superior del cable
        T_top = getTransform(robot,qk,'CableRot2');
        pos_top = tform2trvec(T_top);

        % obtener posición del efector final
        T_ee = getTransform(robot,qk,'Gancho');
        pos = tform2trvec(T_ee);

        % Actualizar cable
        set(cable_plot,...
            'XData',[pos_top(1) pos(1)],...
            'YData',[pos_top(2) pos(2)],...
            'ZData',[pos_top(3) pos(3)]);


        title(sprintf('Grúa torre 5GDL subactuada | Tiempo: %.2f s', t(k)));
```

```
        % dibujar trayectoria
        addpoints(trayectoria,pos(1),pos(2),pos(3));

        % Punto de cada instante
        set(punto_ref, 'XData', xr(k), 'YData', yr(k), 'ZData', zr(k));

        drawnow;
    end
end
```

A.4 Códigos para la generación de trayectorias

Código A.15 Código para cálculo de puntos de paso y paso de referencias al control.

```
function [out]=GTCL_R5GDL(in)

posini=[in(1), in(2), in(3)];
posfin=[in(4), in(5), in(6)];
n=in(7);
inicio=in(8);
intervalo=in(9);
tiempo=in(10);

persistent PL_q1 PL_q2 PL_q3 PL_q4 PL_q5 Ts
if isempty(PL_q1)

    %% definición de variables
    xyz= ones(n,3);
    phi= ones(n);
    theta= ones(n);
    psi_= ones(n);

    % inicialización

    xyz(1,:) = posini;
    xyz(n,:) = posfin;

    %% Tiempo
    t= inicio:intervalo/(n-1):intervalo+inicio;

    %% cálculo de la orientación

    phi(1) = atan2(posini(2), posini(1))-pi/2;
    phi(n) = atan2(posfin(2), posfin(1))-pi/2;
    theta = 0*theta;
    psi_ = -pi*psi_;

    %% Posición inicial y final a var articulares

    posedeseada_ini=[xyz(1,1); xyz(1,2); xyz(1,3); phi(1); theta(1); psi_(1)];

    posedeseada_fin=[xyz(n,1); xyz(n,2); xyz(n,3); phi(n); theta(n); psi_(n)];

    [q_ini, val_ini] = CinematicaInv(posedeseada_ini);
```

```
[q_fin, val_fin] = CinematicaInv(posedeseada_fin);

%% Si no da error se interpola

tipo_tray=0;
if(val_ini==1 || val_fin==1)
       error('ERROR en posición inicial o final: Una de las dos posiciones
           no es alcanzable')
else
    if tipo_tray == 0
        % --- TRAYECTORIA EN LÍNEA RECTA ---
        xyz(:,1) = linspace(posini(1), posfin(1), n);
        xyz(:,2) = linspace(posini(2), posfin(2), n);
        xyz(:,3) = linspace(posini(3), posfin(3), n);

        for i = 1:n
            phi(i)=atan2(xyz(i,2), xyz(i,1))-pi/2;

            pose_i = [xyz(i,1); xyz(i,2); xyz(i,3); phi(i); theta(i); psi_(i
                )];
            [q_sol, val] = CinematicaInv(pose_i);

            % Comprobación de error punto por punto
            if val == 1
                error(['ERROR: La posición cartesiana en el paso ', num2str(i
                    ), ' no es alcanzable.']);
            end

            Q(:, i) = q_sol;
        end

    elseif tipo_tray == 1
        % --- TRAYECTORIA EN CURVA LATERAL (PLANO XY) ---
        % Mantenemos Z como línea recta pura
        s = linspace(0, 1, n); % Vector de progreso normalizado (0 a 1)
        xyz(:,3) = posini(3) + (posfin(3) - posini(3)) * s;

        % Vector de dirección en el plano XY
        dx = posfin(1) - posini(1);
        dy = posfin(2) - posini(2);
        longitud_xy = norm([dx, dy]);

        % Vector perpendicular normalizado en XY (-dy, dx)
        if longitud_xy > 1e-4 % Evitar división por 0 si el movimiento es
            vertical
            nx = -dy / longitud_xy;
            ny = dx / longitud_xy;
        else
            nx = 0; ny = 0;
        end

        % Definimos cuánto queremos que sea el max de la curva hacia los
            lados (m)
        desviacion_maxima = 5;

        % Perfil parabólico que vale 0 al inicio/fin y el máximo en el
            centro
```

```
            curva_lateral = desviacion_maxima * 4 * (s .* (1 - s));

            % Sumamos a la línea recta la desviación perpendicular
            xyz(:,1) = (posini(1) + dx * s) + curva_lateral * nx;
            xyz(:,2) = (posini(2) + dy * s) + curva_lateral * ny;
            for i = 1:n
                phi(i)=atan2(xyz(i,2), xyz(i,1))-pi/2;

                pose_i = [xyz(i,1); xyz(i,2); xyz(i,3); phi(i); theta(i); psi_(i
                    )];
                [q_sol, val] = CinematicaInv(pose_i);

                % Comprobación de error punto por punto
                if val == 1
                    error(['ERROR: La posición cartesiana en el paso ', num2str(i
                        ), ' no es alcanzable.']);
                end

                Q(:, i) = q_sol;
            end

        else
            % --- TRAYECTORIA PTO A PTO (INTERPOLADO EN Q) ---

            q1 = linspace(q_ini(1), q_fin(1), n);
            q2 = linspace(q_ini(2), q_fin(2), n);
            q3 = linspace(q_ini(3), q_fin(3), n);
            q4 = linspace(q_ini(4), q_fin(4), n);
            q5 = linspace(q_ini(5), q_fin(5), n);

            Q=[q1;q2;q3;q4;q5];
        end

    end

    %% Ver las posiciones a las que corresponden la orientación y se sacan las
    % trayectorias de cada articulación

    for i= 1:n
        [result] = CinematicaDir(Q(:, i));
        xyz(i,:)=result(1:3);
    end

    PL_q1= p_cubico(Q(1,:)',t',2);
    PL_q2= p_cubico(Q(2,:)',t',2);
    PL_q3= p_cubico(Q(3,:)',t',2);
    PL_q4= p_cubico(Q(4,:)',t',2);
    PL_q5= p_cubico(Q(5,:)',t',2);

    Ts= PL_q1(2,1) - PL_q1(1,1)

    %% Comprobación de toda la trayectoria
    if any(abs(PL_q2(:,2)) > 14 | PL_q5(:,2) > 24 | PL_q5(:,2) < -12)
        error('La trayectoria calculada sale fuera de los límites de las
            variables articulares')
    end
end
```

```
%% Mandar referencias correspondientemente al tiempo

idx = floor((tiempo - inicio)/Ts) + 1;

% Saturación
if idx < 1
    idx = 1;
elseif idx > length(PL_q1(:,1))
    idx = length(PL_q1(:,1));
end

if isempty(Ts) || tiempo < inicio
    q = [
        PL_q1(1,2);
        PL_q2(1,2);
        PL_q3(1,2);
        PL_q4(1,2);
        PL_q5(1,2);
    ];

    qd = [
        PL_q1(1,3);
        PL_q2(1,3);
        PL_q3(1,3);
        PL_q4(1,3);
        PL_q5(1,3);
    ];

    qdd = [
        PL_q1(1,4);
        PL_q2(1,4);
        PL_q3(1,4);
        PL_q4(1,4);
        PL_q5(1,4);
    ];
else
    q = [
        PL_q1(idx,2);
        PL_q2(idx,2);
        PL_q3(idx,2);
        PL_q4(idx,2);
        PL_q5(idx,2);
    ];

    qd = [
        PL_q1(idx,3);
        PL_q2(idx,3);
        PL_q3(idx,3);
        PL_q4(idx,3);
        PL_q5(idx,3);
    ];

    qdd = [
        PL_q1(idx,4);
        PL_q2(idx,4);
        PL_q3(idx,4);
        PL_q4(idx,4);
```

```
        PL_q5(idx,4);
    ];

end

out=[q;qd;qdd];
end
```

Código A.16 Código para cálculo de variables articulares según los coeficientes del polinomio cúbico y graficado de la trayectoria.

```
function PL=p_cubico(q,t,met,qd)
% Dibuja el resultado de interpolar mediante spline cubico los valores q en
% los instantes tt con las velocidades de paso qd.
% Devuelve un vector con 1 fila por punto con los
% resultados (t,q,qd,qdd)de muestrear el polinomio interpolador
% Utiliza la funcion i_cubico que obtiene los valores de los coeficientes
% Las velocidades de paso pueden ser expecificadas o en caso contrario se
% utiliza la expresión [6.8]
% Ejercicio 6.2
%   (c) FUNDAMENTOS DE ROBOTICA (A. Barrientos) McGrawHill 2007
%-----------------------------------------------------------------


npuntos=800; %numero de puntos a pintar por intervalo
n=length(q);   % numero de intervalos

PL=[];
% figure;
% hold on

% Obtiene los coeficientes de los splines cubicos
% [ti,tf,a,b,c,d] para cada intervalo
if nargin==4
    P=i_cubico(q,t,qd,met);
else
    P=i_cubico(q,t,met);
end


for intervalo=1:n-1
    ti =P(intervalo,1);
    tf =P(intervalo,2);
    a  =P(intervalo,3);
    b  =P(intervalo,4);
    c  =P(intervalo,5);
    d  =P(intervalo,6);

    inc=(tf-ti)/npuntos;
    for tt=ti:inc:tf
        qt=a+b*(tt-ti)+c*(tt-ti)^2+d*(tt-ti)^3;
        qdt=b+2*c*(tt-ti)+3*d*(tt-ti)^2;
        qddt=2*c+6*d*(tt-ti);
        % plot(tt,qt,'k');
        PL=vertcat(PL,[tt,qt,qdt,qddt]);
    end
```

```
% subplot(3,1,1);
% plot(t, q, 'o'); % Puntos de paso originales
% hold on;
% plot(PL(:,1), PL(:,2), 'r'); % Trayectoria interpolada (Posición)
% grid on;
% ylabel('Posición (q)');
% title('Análisis de la Trayectoria');
% legend('Puntos via', 'Interpolación');
%
% % Subplot 2: Velocidad
% subplot(3,1,2);
% plot(PL(:,1), PL(:,3), 'b');
% grid on;
% ylabel('Velocidad (qd)');
%
% % Subplot 3: Aceleración
% subplot(3,1,3);
% plot(PL(:,1), PL(:,4), 'g');
% grid on;
% ylabel('Aceleración (qdd)');
% xlabel('Tiempo (t)');
%
% hold off;
end
```

Código A.17 Código para cálculo de coeficientes del polinomio cúbico y las velocidades de paso.

```
function P=i_cubico(q,t,met,qd)
% Obtiene los coeficientes de los splines cubicos
% que interpolan los valores q en los instantes t
% con las velocidades de paso qd
% Las velocidades de paso pueden ser especificadas
% o en caso contrario se utiliza la expresión [6.8]
% Retorna un vector con una fila por cada tramo
% con [ti,tf,a,b,c,d] siendo el polinomio:
% q(t)=a+b(t-ti)+c(t-ti)^2+d(t-ti)^3 para ti<t<tf
% Ejercicio 6.2
%   (c) FUNDAMENTOS DE ROBOTICA (A. Barrientos) McGrawHill 2007
%-------------------------------------------------------------------

n=length(q);
if n~=length(t)
    error('ERROR en i_cubico: Las dimensiones de q y t deben ser iguales')
end

if (met ~= 1 && met ~= 2)
    error('ERROR en i_cubico: El método seleccionado debe ser o 1 o 2')
end

if nargin~= 4  % qd no definida. La obtiene segun [6.8]
   qd(1)=0;
   qd(n)=0;
   if met==1
      for i=2:n-1
          if (sign(q(i)-q(i-1))==sign(q(i+1)-q(i)))...
```

```
                |q(i)==q(i+1)...
                |q(i-1)==q(i)
            qd(i)=0.5*((q(i+1)-q(i))/(t(i+1)-t(i))+ ...
           +(q(i)-q(i-1))/(t(i)-t(i-1)));
        else
            qd(i)=0;
        end
    end

elseif met == 2
    h = diff(t); %incremento entre intervalos

    num_int = n-2;
    A = zeros(num_int, num_int);
    B = zeros(num_int, 1);

    %velocidades en extremos
    qd_inicio = 0;
    qd_final = 0;

    for i = 1:num_int
        idx = i + 1; % indice del punto real en q y t

        %A%
        if i > 1
            A(i, i-1) = h(idx);
        end

        A(i, i) = 2*(h(idx-1) + h(idx));

        if i < num_int
            A(i, i+1) = h(idx-1);
        end

        %B%
        term1 = (h(idx-1)^2) * (q(idx+1) - q(idx));
        term2 = (h(idx)^2) * (q(idx) - q(idx-1));
        denominador = h(idx-1) * h(idx);

        B(i) = (3 / denominador) * (term1 + term2);

        % Por si las v en los extremos ~ de 0
        if i == 1
            B(i) = B(i) - h(idx) * qd_inicio;
        end
        if i == num_int
            B(i) = B(i) - h(idx-1) * qd_final;
        end
    end

    vel_paso = A \ B;

    % 4. Componemos el vector final de velocidades de paso
    qd = [qd_inicio; vel_paso; qd_final];
end
end
```

```
% obtiene los coeficientes de los polinomios
    for i=1:n-1
        ti=t(i);
        tii=t(i+1);
        if tii<=ti
            error ('ERROR en i_cubico. Los tiempos deben estar ordenados: t(i)
                debe ser < t(i+1)')
        end
        T=tii-ti;
        TT(:,i)=[ti;tii];
        a(i)=q(i);
        b(i)=qd(i);
        c(i)= 3/T^2*(q(i+1)-q(i)) - 1/T *(qd(i+1)+2*qd(i));
        d(i)=-2/T^3*(q(i+1)-q(i)) + 1/T^2*(qd(i+1) +qd(i));
    end

    P=[TT;a; b; c; d]';

end
```

A.5 Códigos de la dinámica del sistema

Código A.18 Cálculo del modelo dinámico inverso simbólico del sistema.

```
% Ejemplo de la utilización del algoritmo de Newton Euler para la dinámica
% de un robot de 3 DGL
% M.G. Ortega (2020)
% Modificado C. Vivas (2023)
% Modificado Alejandro Noci (2025-2026)

clear
clc
% Elegir entre R (rotación) y P (prismática)
Tipo_Q1 = 'R';
Tipo_Q2 = 'P';
Tipo_Q3 = 'R';
Tipo_Q4 = 'R';
Tipo_Q5 = 'P';

if ( (Tipo_Q1 ~= 'R') & (Tipo_Q1 ~='P')); error('Elegir R o P para Tipo_Q1');
    end
if ( (Tipo_Q2 ~= 'R') & (Tipo_Q2 ~='P')); error('Elegir R o P para Tipo_Q2');
    end
if ( (Tipo_Q3 ~= 'R') & (Tipo_Q3 ~='P')); error('Elegir R o P para Tipo_Q3');
    end
if ( (Tipo_Q4 ~= 'R') & (Tipo_Q4 ~='P')); error('Elegir R o P para Tipo_Q4');
    end
if ( (Tipo_Q5 ~= 'R') & (Tipo_Q5 ~='P')); error('Elegir R o P para Tipo_Q5');
    end


% Definición de variables simbólicas
syms T1 T2 T3 T4 T5 q1 qd1 qdd1 q2 qd2 qdd2 q3 qd3 qdd3 q4 qd4 qdd4 q5 qd5 qdd5
     real
```

```
PI = sym(pi); % Importante para cáculo simbólico

% Definición de constantes simbólicas
syms L0 Lcp m Mt Mcp J0 g real
syms jm1 jm2 jm3 jm4 jm5 bm1 bm2 bm3 bm4 bm5 real

% DATOS CINEMÁTICOS DEL BRAZO DEL ROBOT

% Elección entre modelo simplificado (S) o completo (C)
  Tipo_modelo = 'C';
  if ( (Tipo_modelo ~= 'S') & (Tipo_modelo ~='C')); error('Elegir S o C para
      Tipo_modelo'); end


  if (Tipo_modelo=='S') % Comprobar todas las ecuaciones
      L0=0;
      L1B=0;
      L1A=0;
      L2=0;

  % DATOS DINÁMICOS DEL BRAZO DEL ROBOT
    % Eslabón 0 (fijo. No hace falta especificarlo porque no se mueve respecto
        al sistema Base B)
      m0= 0; % kg
      s00 = [0,0,-L0/2]'; % m
      I00=zeros(3); % kg.m2

    % Eslabón 1, Pluma, J0 masa equivalente de la grúa en el
    % extremo
      m1= 1; % kg
      s11 = [ 0, 0, L1B/2]'; % m
      I11=[ J0, 0,  0; 0,  J0, 0; 0, 0, J0]; % kg.m2

    % Eslabón 2
      m2= Mt; % kg
      s22 = [ 0, 0, 0]'; % m
      I22= zeros(3); % kg.m2

    % Eslabón 3
      m3= 0; % kg
      s33 = [ 0, 0, 0]'; % m
      I33= zeros(3); % kg.m2

    % Eslabón 4
      m4= 0; % kg
      s44 = [ 0, 0, 0]'; % m
      I44= zeros(3); % kg.m2

    % Eslabón 5
      m5= m; % kg
      s55 = [ 0, 0, L2/2]'; % m
      I55= zeros(3); % kg.m2

      R1=1;
      R2=1;
      R5=1;
  else
```

```
    syms L0 L1A L1B L2 M1 M0 M5 R1 R2 R5 R real

    L0=L0;
    L1B=L1B;
    L1A=L1A;
    L2=L2;

% DATOS DINÁMICOS DEL BRAZO DEL ROBOT
  % Eslabón 0 (fijo. No hace falta especificarlo porque no se mueve respecto
      al sistema Base B)
    m0= M0; % kg
    s00 = [ 0, 0, -L0/2]'; % m
    I00=[ m0*L0^2/12,        0,         0;
                  0,  m0*L0^2/12,        0;
                  0,          0,  m0*R^2]; % kg.m2

  % Eslabón 1, pluma
    m1= M1 + Mcp; % kg
    s11 = [ 0, 0, L1B/2]'; % m
    I11g= [ m1*L1B^2/12,        0,         0;
            0,  m1*L1B^2/12,        0;
            0,          0,  m1*R^2]; % kg.m2
    I11= I11g + Steiner([0, 0, Lcp+L1B/2],zeros(3),Mcp); % kg.m2

  % Eslabón 2
    m2= Mt; % kg
    s22 = [ 0, 0, 0]'; % m
    I22= zeros(3); % kg.m2

  % Eslabón 3
    m3= 0; % kg
    s33 = [ 0, 0, 0]'; % m
    I33= zeros(3); % kg.m2

  % Eslabón 4
    m4= 0; % kg
    s44 = [ 0, 0, 0]'; % m
    I44= zeros(3); % kg.m2

  % Eslabón 5
    m5= m + M5; % kg
    s55 = [ 0, 0, L2/2]'; % m

    I55g= [ m5*L2^2/12,        0,      0;
                  0,  m5*L2^2/12,     0;
                  0,          0,     0]; % kg.m2
    I55= I55g + Steiner(-s55',zeros(3),m); % kg.m2
  end

% Parámetros de Denavit-Hartenberg (utilizado en primera regla de Newton-Euler)
% Eslabón base
  theta0=0; d0=L0; a0=0; alpha0=0;
% Eslabón 1:
  theta1= q1+PI/2; d1= 0; a1= 0; alpha1= PI/2;
% Eslabón 2:
  theta2= -PI/2; d2= L1A+q2; a2= 0; alpha2= PI/2;
% Eslabón 3:
```

```
  theta3= q3; d3= 0; a3= 0; alpha3= PI/2;
% Eslabón 4:
  theta4= q4+PI/2; d4= 0; a4= 0; alpha4= PI/2;
% Eslabón 5:
  theta5= 0; d5= q5; a5= 0; alpha5= 0;
% Entre eslabón 5 y marco donde se ejerce la fuerza (a definir según
% experimento)
  theta6= 0; d6= L2; a6= 0; alpha6= 0;

% DATOS DE LOS MOTORES
% Inercias
  Jm1=jm1 ; Jm2=jm2 ; Jm3=jm3 ; Jm4=jm4 ; Jm5=jm5 ; % kg.m2
% Coeficientes de fricción viscosa
  Bm1= bm1; Bm2= bm2; Bm3= bm3; Bm4= bm4; Bm5= bm5; % N.m / (rad/s)
% Factores de reducción
  R1= R1; R2= R2; R3= 1; R4= 1; R5= R5;

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

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

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

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

% N-E 1: Asignación a cada eslabón de sistema de referencia de acuerdo con las
    normas de D-H.
% Eslabón 0:
    p00 = [a0, d0*sin(alpha0), d0*cos(alpha0)]';
% Eslabón 1:
    p11 = [a1, d1*sin(alpha1), d1*cos(alpha1)]';
  % Eslabón 2:
    p22 = [a2, d2*sin(alpha2), d2*cos(alpha2)]';
  % Eslabón 3:
    p33 = [a3, d3*sin(alpha3), d3*cos(alpha3)]';
  % Eslabón 4:
    p44 = [a4, d4*sin(alpha4), d4*cos(alpha4)]';
  % Eslabón 5:
    p55 = [a5, d5*sin(alpha5), d5*cos(alpha5)]';
```

```
  % Eslabón 6
    p66 = [a6, d6*sin(alpha6), d6*cos(alpha6)]';

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

  wBB=[0 0 0]';
  wdBB = [0 0 0]';
  vBB = [0 0 0]';
  vdBB = [0 0 g]'; % Aceleración de la gravedad en el eje Z0 negativo

% Condiciones iniciales para el extremo del robot
  f66= [0 0 0]';
  n66= [0 0 0]';

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


% N-E 3: Obtención de las matrices de rotación (i)R(i-1) y de sus inversas
  RB0=[cos(theta0) -cos(alpha0)*sin(theta0) sin(alpha0)*sin(theta0);
      sin(theta0) cos(alpha0)*cos(theta0) -sin(alpha0)*cos(theta0);
      0           sin(alpha0)             cos(alpha0)         ];
  R0B= RB0';

  R01=[cos(theta1) -cos(alpha1)*sin(theta1) sin(alpha1)*sin(theta1);
      sin(theta1) cos(alpha1)*cos(theta1) -sin(alpha1)*cos(theta1);
      0           sin(alpha1)             cos(alpha1)         ];
  R10= R01';

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

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

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

  R45=[cos(theta5) -cos(alpha5)*sin(theta5) sin(alpha5)*sin(theta5);
      sin(theta5) cos(alpha5)*cos(theta5) -sin(alpha5)*cos(theta5);
      0           sin(alpha5)             cos(alpha5)         ];
  R54= R45';

  R56=[cos(theta6) -cos(alpha6)*sin(theta6) sin(alpha6)*sin(theta6);
      sin(theta6) cos(alpha6)*cos(theta6) -sin(alpha6)*cos(theta6);
      0           sin(alpha6)             cos(alpha6)         ];
  R65= R56';
```

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

% N-E 4: Obtención de las velocidades angulares absolutas
% Articulación 0
    w00 = R0B*wBB;
 % Articulación 1
    if (Tipo_Q1=='R');
        w11= R10*(w00+Z*qd1); % Si es de rotación
    else
        w11 = R10*w00;    % Si es de translación
    end
 % Articulación 2
    if (Tipo_Q2=='R');
        w22= R21*(w11+Z*qd2); % Si es de rotación
    else
        w22 = R21*w11;    % Si es de translación
    end
 % Articulación 3
    if (Tipo_Q3=='R');
        w33= R32*(w22+Z*qd3); % Si es de rotación
    else
        w33 = R32*w22;    % Si es de translación
    end

% Articulación 4
    if (Tipo_Q4=='R');
        w44= R43*(w33+Z*qd4); % Si es de rotación
    else
        w44 = R43*w33;    % Si es de translación
    end
% Articulación 5
    if (Tipo_Q5=='R');
        w55= R54*(w44+Z*qd5); % Si es de rotación
    else
        w55 = R54*w44;    % Si es de translación
    end

% N-E 5: Obtención de las aceleraciones angulares absolutas
% Articulación 0
        wd00 = R0B*wdBB;
 % Articulación 1
    if (Tipo_Q1=='R');
        wd11 = R10*(wd00+Z*qdd1+cross(w00,Z*qd1)); % si es de rotación
    else
        wd11 = R10*wd00;                          % si es de translación
    end
 % Articulación 2
     if (Tipo_Q2=='R');
         wd22 = R21*(wd11+Z*qdd2+cross(w11,Z*qd2)); % si es de rotación
     else
         wd22 = R21*wd11;                           % si es de translación
     end
 % Articulación 3
     if (Tipo_Q3=='R');
         wd33 = R32*(wd22+Z*qdd3+cross(w22,Z*qd3)); % si es de rotación
```

```
    else
        wd33 = R32*wd22;                          % si es de translación
    end

% Articulación 4
    if (Tipo_Q4=='R');
        wd44 = R43*(wd33+Z*qdd4+cross(w33,Z*qd4)); % si es de rotación
    else
        wd44 = R43*wd33;                          % si es de translación
    end

% Articulación 5
    if (Tipo_Q5=='R');
        wd55 = R54*(wd44+Z*qdd5+cross(w44,Z*qd5)); % si es de rotación
    else
        wd55 = R54*wd44;                          % si es de translación
    end

% N-E 6: Obtención de las aceleraciones lineales de los orígenes de los
% sistemas
% Articulación 0
    vd00 = cross(wd00,p00)+cross(w00,cross(w00,p00))+R0B*vdBB;
% Articulación 1
    if (Tipo_Q1=='R');
        vd11 = cross(wd11,p11)+cross(w11,cross(w11,p11))+R10*vd00; % si es de
            rotación
    else
        vd11 = R10*(Z*qdd1+vd00)+cross(wd11,p11)+2*cross(w11,R10*Z*qd1) +
            cross(w11,cross(w11,p11)); % si es de translación
    end
% Articulación 2
    if (Tipo_Q2=='R');
        vd22 = cross(wd22,p22)+cross(w22,cross(w22,p22))+R21*vd11; % si es de
            rotación
    else
        vd22 = R21*(Z*qdd2+vd11)+cross(wd22,p22)+2*cross(w22,R21*Z*qd2) +
            cross(w22,cross(w22,p22)); % si es de translación
    end
% Articulación 3
    if (Tipo_Q3=='R');
        vd33 = cross(wd33,p33)+cross(w33,cross(w33,p33))+R32*vd22; % si es de
            rotación
    else
       vd33 = R32*(Z*qdd3+vd22)+cross(wd33,p33)+2*cross(w33,R32*Z*qd3) + cross
           (w33,cross(w33,p33)); % si es de translación
    end

% Articulación 4
    if (Tipo_Q4=='R');
        vd44 = cross(wd44,p44)+cross(w44,cross(w44,p44))+R43*vd33; % si es de
            rotación
    else
       vd44 = R43*(Z*qdd4+vd33)+cross(wd44,p44)+2*cross(w44,R43*Z*qd4) + cross
           (w44,cross(w44,p44)); % si es de translación
    end

% Articulación 5
```

```
    if (Tipo_Q5=='R');
        vd55 = cross(wd55,p55)+cross(w55,cross(w55,p55))+R54*vd44; % si es de
            rotación
    else
       vd55 = R54*(Z*qdd5+vd44)+cross(wd55,p55)+2*cross(w55,R54*Z*qd5) + cross
           (w55,cross(w55,p55)); % si es de translación
    end

% N-E 7: Obtención de las aceleraciones lineales de los centros de gravedad
    a00 = cross(wd00,s00)+cross(w00,cross(w00,s00))+vd00;
    a11 = cross(wd11,s11)+cross(w11,cross(w11,s11))+vd11;
    a22 = cross(wd22,s22)+cross(w22,cross(w22,s22))+vd22;
    a33 = cross(wd33,s33)+cross(w33,cross(w33,s33))+vd33;
    a44 = cross(wd44,s44)+cross(w44,cross(w44,s44))+vd44;
    a55 = cross(wd55,s55)+cross(w55,cross(w55,s55))+vd55;


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

% N-E 8: Obtención de las fuerzas ejercidas sobre los eslabones
  f55=R56*f66+m5*a55;
  f44=R45*f55+m4*a44;
  f33=R34*f44+m3*a33;
  f22=R23*f33+m2*a22;
  f11=R12*f22+m1*a11;
  f00=R01*f11+m0*a00;
  fBB=RB0*f00;


% N-E 9: Obtención de los pares ejercidas sobre los eslabones
  n55 = R56*(n66+cross(R65*p55,f66))+cross(p55+s55,m5*a55)+I55*wd55+cross(w55,
      I55*w55);
  n44 = R45*(n55+cross(R54*p44,f55))+cross(p44+s44,m4*a44)+I44*wd44+cross(w44,
      I44*w44);
  n33 = R34*(n44+cross(R43*p33,f44))+cross(p33+s33,m3*a33)+I33*wd33+cross(w33,
      I33*w33);
  n22 = R23*(n33+cross(R32*p22,f33))+cross(p22+s22,m2*a22)+I22*wd22+cross(w22,
      I22*w22);
  n11 = R12*(n22+cross(R21*p11,f22))+cross(p11+s11,m1*a11)+I11*wd11+cross(w11,
      I11*w11);
  n00 = R01*(n11+cross(R10*p00,f11))+cross(p00+s00,m0*a00)+I00*wd00+cross(w00,
      I00*w00);
  nBB = RB0*n00;

% N-E 10: Obtener la fuerza o par aplicado sobre la articulación
  N5z = n55'*R54*Z; % Si es de rotación
  N5  = n55'*R54;   % Para ver todos los pares, no solo el del eje Z
  F5z = f55'*R54*Z; % Si es de translacion;
  F5  = f55'*R54;   % Para ver todas las fuerzas, no solo la del eje Z
  N4z = n44'*R43*Z; % Si es de rotación
  N4  = n44'*R43;   % Para ver todos los pares, no solo el del eje Z
  F4z = f44'*R43*Z; % Si es de translacion;
  F4  = f44'*R43;   % Para ver todas las fuerzas, no solo la del eje Z
  N3z = n33'*R32*Z; % Si es de rotación
  N3  = n33'*R32;   % Para ver todos los pares, no solo el del eje Z
  F3z = f33'*R32*Z; % Si es de translacion;
  F3  = f33'*R32;   % Para ver todas las fuerzas, no solo la del eje Z
```

```
  N2z = n22'*R21*Z; % Si es de rotación
  N2  = n22'*R21;   % Para ver todos los pares, no solo el del eje Z
  F2z = f22'*R21*Z; % Si es de translacion;
  F2  = f22'*R21;   % Para ver todas las fuerzas, no solo la del eje Z
  N1z = n11'*R10*Z; % Si es de rotación
  N1  = n11'*R10;   % Para ver todos los pares, no solo el del eje Z
  F1z = f11'*R10*Z; % Si es de translacion;
  F1  = f11'*R10;   % Para ver todas las fuerzas, no solo la del eje Z
  N0z = n00'*R0B*Z; % Si es de rotación
  N0  = n00'*R0B;   % Para ver todos los pares, no solo el del eje Z
  F0z = f00'*R0B*Z; % Si es de translacion;
  F0  = f00'*R0B;   % Para ver todas las fuerzas, no solo la del eje Z

  % Robot RRR o PPP
    if (Tipo_Q1=='R'); T1=N1z; else T1=F1z; end
    if (Tipo_Q2=='R'); T2=N2z; else T2=F2z; end
    if (Tipo_Q3=='R'); T3=N3z; else T3=F3z; end
    if (Tipo_Q4=='R'); T4=N4z; else T4=F4z; end
    if (Tipo_Q5=='R'); T5=N5z; else T5=F5z; end

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

% Primera ecuación
% -----------------
% Cálculo de los términos de la matriz de inercia (afines a qdd)
M11 = diff(T1,qdd1);
Taux = simplify(T1 - M11*qdd1);
M12 = diff(Taux,qdd2);
Taux = simplify(Taux-M12*qdd2);
M13= diff(Taux,qdd3);
Taux = simplify(Taux-M13*qdd3);
M14= diff(Taux,qdd4);
Taux = simplify(Taux-M14*qdd4);
M15= diff(Taux,qdd5);
Taux = simplify(Taux-M15*qdd5);
% Taux restante contiene términos Centrípetos/Coriolis y Gravitatorios
% Términos gravitatorios: dependen linealmente de "g"
G1=diff(Taux,g)*g;
Taux=simplify(Taux-G1);
% Taux restante contiene términos Centrípetos/Coriolis
V1=Taux;

% Segunda ecuación
% -----------------
% Cálculo de los términos de la matriz de inercia (afines a qdd)
M21 = diff(T2,qdd1);
Taux = simplify(T2 - M21*qdd1);
M22 = diff(Taux,qdd2);
Taux = simplify(Taux-M22*qdd2);
M23 = diff(Taux,qdd3);
Taux = simplify(Taux-M23*qdd3);
M24 = diff(Taux,qdd4);
Taux = simplify(Taux-M24*qdd4);
```

```
M25 = diff(Taux,qdd5);
Taux = simplify(Taux-M25*qdd5);
% Taux restante contiene términos Centrípetos/Coriolis y Gravitatorios
% Términos gravitatorios: dependen linealmente de "g"
G2=diff(Taux,g)*g;
Taux=simplify(Taux-G2);
% Taux restante contiene términos Centrípetos/Coriolis
V2=Taux;

% Tercera ecuación
% -----------------
% Cálculo de los términos de la matriz de inercia (afines a qdd)
M31 = diff(T3,qdd1);
Taux = simplify(T3 - M31*qdd1);
M32 = diff(Taux,qdd2);
Taux = simplify(Taux-M32*qdd2);
M33 = diff(Taux,qdd3);
Taux = simplify(Taux-M33*qdd3);
M34 = diff(Taux,qdd4);
Taux = simplify(Taux-M34*qdd4);
M35 = diff(Taux,qdd5);
Taux = simplify(Taux-M35*qdd5);
% Taux restante contiene términos Centrípetos/Coriolis y Gravitatorios
% Términos gravitatorios: dependen linealmente de "g"
G3=diff(Taux,g)*g;
Taux=simplify(Taux-G3);
% Taux restante contiene términos Centrípetos/Coriolis
V3=Taux;

% Cuarta ecuación
% -----------------
% Cálculo de los términos de la matriz de inercia (afines a qdd)
M41 = diff(T4,qdd1);
Taux = simplify(T4 - M41*qdd1);
M42 = diff(Taux,qdd2);
Taux = simplify(Taux-M42*qdd2);
M43 = diff(Taux,qdd3);
Taux = simplify(Taux-M43*qdd3);
M44 = diff(Taux,qdd4);
Taux = simplify(Taux-M44*qdd4);
M45 = diff(Taux,qdd5);
Taux = simplify(Taux-M45*qdd5);
% Taux restante contiene términos Centrípetos/Coriolis y Gravitatorios
% Términos gravitatorios: dependen linealmente de "g"
G4=diff(Taux,g)*g;
Taux=simplify(Taux-G4);
% Taux restante contiene términos Centrípetos/Coriolis
V4=Taux;

% Quinta ecuación
% -----------------
% Cálculo de los términos de la matriz de inercia (afines a qdd)
M51 = diff(T5,qdd1);
Taux = simplify(T5 - M51*qdd1);
M52 = diff(Taux,qdd2);
Taux = simplify(Taux-M52*qdd2);
M53 = diff(Taux,qdd3);
```

```
Taux = simplify(Taux-M53*qdd3);
M54 = diff(Taux,qdd4);
Taux = simplify(Taux-M54*qdd4);
M55 = diff(Taux,qdd5);
Taux = simplify(Taux-M55*qdd5);
% Taux restante contiene términos Centrípetos/Coriolis y Gravitatorios
% Términos gravitatorios: dependen linealmente de "g"
G5=diff(Taux,g)*g;
Taux=simplify(Taux-G5);
% Taux restante contiene términos Centrípetos/Coriolis
V5=Taux;


% Simplificación de expresiones
M11=simplify(M11); M12=simplify(M12); M13=simplify(M13); M14=simplify(M14);M15=
    simplify(M15);
M21=simplify(M21); M22=simplify(M22); M23=simplify(M23); M24=simplify(M24);M25=
    simplify(M25);
M31=simplify(M31); M32=simplify(M32); M33=simplify(M33); M34=simplify(M34);M35=
    simplify(M35);
M41=simplify(M41); M42=simplify(M42); M43=simplify(M43); M44=simplify(M44);M45=
    simplify(M45);
M51=simplify(M51); M52=simplify(M52); M53=simplify(M53); M54=simplify(M54);M55=
    simplify(M55);


V1=simplify(V1); V2=simplify(V2); V3=simplify(V3); V4=simplify(V4); V5=simplify
    (V5);
G1=simplify(G1); G2=simplify(G2); G3=simplify(G3); G4=simplify(G4); G5=simplify
    (G5);


% Apilación en matrices y vectores
M = [M11 M12 M13 M14 M15;
    M21 M22 M23 M24 M25;
    M31 M32 M33 M34 M35;
    M41 M42 M43 M44 M45;
    M51 M52 M53 M54 M55];

V = [V1; V2; V3; V4; V5];
G = [G1; G2; G3; G4; G5];


% Inclusión de los motores en la ecuación dinámica
%
% T= Ma(q)qdd+Va(q,qd)+Ga(q)
%
% Ma = M + R^2*Jm   Va=V + R^2*Bm*qd   Ga=G
%
R=diag([R1 R2 R3 R4 R5]);
Jm=diag([Jm1 Jm2 Jm3 Jm4 Jm5]);
Bm=diag([Bm1 Bm2 Bm3 Bm4 Bm5]);
% Kt=diag([Kt1 Kt2 Kt3]); % No utilizado

Ma=M+R*R*Jm;
Va=V+R*R*Bm*[qd1 ; qd2 ; qd3; qd4; qd5];
Ga = G;
```

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

Código A.19 Cálculo del modelo dinámico inverso numérico del sistema.

```
function [Ma_ne,Va_ne,Ga_ne] = Modelo_Din(in)
clc

Tipo_modelo = in;

% Definición de variables simbólicas
syms T1 T2 T3 T4 T5 q1 qd1 qdd1 q2 qd2 qdd2 q3 qd3 qdd3 q4 qd4 qdd4 q5 qd5 qdd5
     real
PI = pi;

% Definición de constantes
L0 = sym(45); % m
L1B = sym(35); % m
L1A = sym(35/2); % m
L2 = sym(15); % m
Lcp = sym(10); % m

M0 = sym(7991.3); % kg
M1 = sym(1236); % kg
M5 = sym(40); % kg
m = 0; % kg
Mcp = sym(7400); % kg

R = sym(1.2); % m

jm1 = sym(360e-4); % kg*m^2
jm2 = sym(34e-4); % kg*m^2
jm3 = 0; % kg*m^2
jm4 = 0; % kg*m^2
jm5 = sym(270e-4); % kg*m^2

bm1 = sym(0.191); % N*m*s
bm2 = sym(0.042); % N*m*s
bm3 = 0; % N*m*s
bm4 = 0; % N*m*s
bm5 = sym(0.130); % N*m*s

R1 = sym(94.21);
R2 = sym(91.53);
R5 = sym(177);

J0=  10215; % kg*m^2

g=sym(9.81); % m/s^2
```

```
% Elección entre modelo simplificado (S) o completo (C)
  if ( (Tipo_modelo ~= 'S') && (Tipo_modelo ~='C') && (Tipo_modelo ~= 'R'));
      error('Elegir S, C o R para Tipo_modelo'); end


  if (Tipo_modelo=='S')
      Mt = 1140.75; % kg

    Ma_ne=[J0 + jm1 + Mt*q2^2 + m*q2^2 + m*q5^2 - 1.0*m*q5^2*cos(q3)^2*cos(q4)
        ^2 + 2.0*m*q2*q5*cos(q4)*sin(q3),  m*q5*sin(q4), m*q5^2*cos(q3)*cos(q4)
        *sin(q4), -1.0*m*q5*(q2*cos(q4) + q5*sin(q3)), -1.0*m*q2*sin(q4);
m*q5*sin(q4),           Mt + jm2 + m,         m*q5*cos(q3)*cos(q4),
    -1.0*m*q5*sin(q3)*sin(q4), m*cos(q4)*sin(q3);
m*q5^2*cos(q3)*cos(q4)*sin(q4),  m*q5*cos(q3)*cos(q4),      m*q5^2*cos(q4)^2 +
     jm3,                              0,               0;
-1.0*m*q5*(q2*cos(q4) + q5*sin(q3)), -1.0*m*q5*sin(q3)*sin(q4),
                        0,                       m*q5^2 + jm4,              0;
-1.0*m*q2*sin(q4),      m*cos(q4)*sin(q3),                          0,
                                  0,          jm5 + m];

    Va_ne= [bm1*qd1 + 2.0*Mt*q2*qd1*qd2 + 2.0*m*q2*qd1*qd2 + 2.0*m*q5*qd1*qd5 -
         2.0*m*q5*qd4*qd5*sin(q3) - 2.0*m*q5^2*qd3*qd4*cos(q3) + m*q2*q5*qd4^2*
        sin(q4) - 2.0*m*q2*qd4*qd5*cos(q4) - 1.0*m*q5^2*qd3^2*cos(q4)*sin(q3)*
        sin(q4) - 2.0*m*q5*qd1*qd5*cos(q3)^2*cos(q4)^2 + 2.0*m*q5^2*qd3*qd4*cos
        (q3)*cos(q4)^2 + 2.0*m*q2*qd1*qd5*cos(q4)*sin(q3) + 2.0*m*q5*qd1*qd2*
        cos(q4)*sin(q3) + 2.0*m*q2*q5*qd1*qd3*cos(q3)*cos(q4) - 2.0*m*q2*q5*qd1
        *qd4*sin(q3)*sin(q4) + 2.0*m*q5^2*qd1*qd3*cos(q3)*cos(q4)^2*sin(q3) +
        2.0*m*q5^2*qd1*qd4*cos(q3)^2*cos(q4)*sin(q4) + 2.0*m*q5*qd3*qd5*cos(q3)
        *cos(q4)*sin(q4);
bm2*qd2 - 1.0*m*q2*qd1^2 - 1.0*Mt*q2*qd1^2 + 2.0*m*qd1*qd5*sin(q4) + 2.0*m*qd3*
    qd5*cos(q3)*cos(q4) - 2.0*m*qd4*qd5*sin(q3)*sin(q4) - 1.0*m*q5*qd1^2*cos(q4
    )*sin(q3) - 1.0*m*q5*qd3^2*cos(q4)*sin(q3) - 1.0*m*q5*qd4^2*cos(q4)*sin(q3)
     + 2.0*m*q5*qd1*qd4*cos(q4) - 2.0*m*q5*qd3*qd4*cos(q3)*sin(q4);
bm3*qd3 + m*q5*cos(q4)*(2.0*qd3*qd5*cos(q4) - 1.0*q2*qd1^2*cos(q3) - 2.0*q5*qd3
    *qd4*sin(q4) + 2.0*qd1*qd5*cos(q3)*sin(q4) - 1.0*q5*qd1^2*cos(q3)*cos(q4)*
    sin(q3) + 2.0*q5*qd1*qd4*cos(q3)*cos(q4));
bm4*qd4 - 1.0*m*q5*(2.0*qd1*qd2*cos(q4) - 2.0*qd4*qd5 + 2.0*qd1*qd5*sin(q3) -
    1.0*q5*qd3^2*cos(q4)*sin(q4) - 1.0*q2*qd1^2*sin(q3)*sin(q4) + q5*qd1^2*cos(
    q3)^2*cos(q4)*sin(q4) + 2.0*q5*qd1*qd3*cos(q3)*cos(q4)^2);
bm5*qd5 - 1.0*m*(q5*qd1^2 + q5*qd4^2 + 2.0*qd1*qd2*sin(q4) + q5*qd3^2*cos(q4)^2
     - 1.0*q5*qd1^2*cos(q3)^2*cos(q4)^2 + q2*qd1^2*cos(q4)*sin(q3) - 2.0*q5*qd1
    *qd4*sin(q3) + 2.0*q5*qd1*qd3*cos(q3)*cos(q4)*sin(q4))];

    Ga_ne= [         0;
                     0;
              g*m*q5*cos(q4)*sin(q3);
              g*m*q5*cos(q3)*sin(q4);
             -1.0*g*m*cos(q3)*cos(q4)];
  elseif (Tipo_modelo=='R')
    Mt = 0; % kg
    m = sym(2000); % kg
   % Se define Ma y Va_ne

    Ga_ne=[                     0
                                0
            g*cos(q4)*sin(q3)*(M5 + m)*(0.5*L2 + q5)
            g*cos(q3)*sin(q4)*(M5 + m)*(0.5*L2 + q5)
```

```
            -2.0*g*cos(q3)*cos(q4)*(0.5*M5 + 0.5*m)];

  else
      Mt = 0; % kg

   % Se define Ma y Va_ne

    Ga_ne=[                    0
                               0
           g*cos(q4)*sin(q3)*(M5 + m)*(0.5*L2 + q5)
           g*cos(q3)*sin(q4)*(M5 + m)*(0.5*L2 + q5)
            -2.0*g*cos(q3)*cos(q4)*(0.5*M5 + 0.5*m)];
  end

 Ma_ne = simplify(Ma_ne);
 Va_ne = simplify(Va_ne);
 Ga_ne = simplify(Ga_ne);

 Ma_ne = vpa(Ma_ne,3);
 Va_ne = vpa(Va_ne,3);
 Ga_ne = vpa(Ga_ne,3);


end
```

Código A.20 Cálculo de las aceleraciones del sistema a partir del modelo. Código usado en simulación.

```
function [qdd] = ModeloDinamico_R5GDL(in)
q1       = in(1);
q2       = in(2);
q3       = in(3);
q4       = in(4);
q5       = in(5);
qd1      = in(6);
qd2      = in(7);
qd3      = in(8);
qd4      = in(9);
qd5      = in(10);
Tau1     = in(11);
Tau2     = in(12);
Tau3     = in(13);
Tau4     = in(14);
Tau5     = in(15);
modelo   = in(16);
% A rellenar por el alumno
T=[Tau1;Tau2;Tau3;Tau4;Tau5];
g=9.81;
if modelo==1
    Ma_ne=[cos(q3)*(cos(q3)*(20.0*q2+350.0)*(2.0*q2+15.0*cos(q4)*sin(q3)+2.0*q5
        *cos(q4)*sin(q3)+35.0)+cos(q3)*sin(q4)^2*(40.0*q5^2+600.0*q5+3000.0))+
        sin(q3)*(750.0*sin(q3)+(40.0*q5+300.0)*(17.5*cos(q4)+7.5*sin(q3)+q2*cos
        (q4)+q5*sin(q3))+sin(q3)*(20.0*q2+350.0)*(2.0*q2+15.0*cos(q4)*sin(q3)
        +2.0*q5*cos(q4)*sin(q3)+35.0))+9.12e+6,20.0*sin(q4)*(2.0*q5+15.0)
        ,0.0833*cos(q3)*cos(q4)*sin(q4)*(480.0*q5^2+7200.0*q5+3.6e+4),-5250.0*
        cos(q4)-3000.0*sin(q3)-40.0*q5^2*sin(q3)-300.0*q2*cos(q4)-700.0*q5*cos(
        q4)-600.0*q5*sin(q3)-40.0*q2*q5*cos(q4),-40.0*sin(q4)*(q2+17.5);
```

```
    20.0*sin(q4)*(2.0*q5+15.0),68.5,20.0*cos(q3)*cos(q4)*(2.0*q5+15.0),-20.0*
        sin(q3)*sin(q4)*(2.0*q5+15.0),40.0*cos(q4)*sin(q3);
    cos(q3)*cos(q4)*sin(q4)*(40.0*q5^2+600.0*q5+3000.0),40.0*cos(q3)*cos(q4)*(
        q5+7.5),cos(q4)^2*(40.0*q5^2+600.0*q5+3000.0),0,0;
    -750.0*sin(q3)-1.0*(40.0*q5+300.0)*(17.5*cos(q4)+7.5*sin(q3)+q2*cos(q4)+q5*
        sin(q3)),-40.0*sin(q3)*sin(q4)*(q5+7.5),0,40.0*(q5+7.5)^2+750.0,0;
    -40.0*sin(q4)*(q2+17.5),40.0*cos(q4)*sin(q3),0,0,886.0];

    Va_ne=[1700.0*qd1+5250.0*qd4^2*sin(q4)+1400.0*qd1*qd2+600.0*qd1*qd5-6000.0*
        qd3*qd4*cos(q3)-1400.0*qd4*qd5*cos(q4)-600.0*qd4*qd5*sin(q3)+300.0*q2*
        qd4^2*sin(q4)+700.0*q5*qd4^2*sin(q4)+80.0*q2*qd1*qd2+80.0*q5*qd1*qd5
        -1.05e+4*qd1*qd4*sin(q3)*sin(q4)+6000.0*qd3*qd4*cos(q3)*cos(q4)^2-80.0*
        q2*qd4*qd5*cos(q4)-1200.0*q5*qd3*qd4*cos(q3)-80.0*q5*qd4*qd5*sin(q3)
        -600.0*qd1*qd5*cos(q3)^2*cos(q4)^2-3000.0*qd3^2*cos(q4)*sin(q3)*sin(q4)
        -80.0*q5^2*qd3*qd4*cos(q3)+1.05e+4*qd1*qd3*cos(q3)*cos(q4)+40.0*q2*q5*
        qd4^2*sin(q4)+600.0*qd1*qd2*cos(q4)*sin(q3)+1400.0*qd1*qd5*cos(q4)*sin(
        q3)-80.0*q5*qd1*qd5*cos(q3)^2*cos(q4)^2+80.0*q5^2*qd3*qd4*cos(q3)*cos(
        q4)^2+6000.0*qd1*qd3*cos(q3)*cos(q4)^2*sin(q3)+6000.0*qd1*qd4*cos(q3)
        ^2*cos(q4)*sin(q4)-600.0*q5*qd3^2*cos(q4)*sin(q3)*sin(q4)+600.0*q2*qd1*
        qd3*cos(q3)*cos(q4)+1400.0*q5*qd1*qd3*cos(q3)*cos(q4)+80.0*q2*qd1*qd5*
        cos(q4)*sin(q3)+80.0*q5*qd1*qd2*cos(q4)*sin(q3)-600.0*q2*qd1*qd4*sin(q3
        )*sin(q4)-1400.0*q5*qd1*qd4*sin(q3)*sin(q4)-40.0*q5^2*qd3^2*cos(q4)*sin
        (q3)*sin(q4)+1200.0*q5*qd3*qd4*cos(q3)*cos(q4)^2+600.0*qd3*qd5*cos(q3)*
        cos(q4)*sin(q4)+80.0*q5*qd3*qd5*cos(q3)*cos(q4)*sin(q4)+1200.0*q5*qd1*
        qd3*cos(q3)*cos(q4)^2*sin(q3)+1200.0*q5*qd1*qd4*cos(q3)^2*cos(q4)*sin(
        q4)+80.0*q2*q5*qd1*qd3*cos(q3)*cos(q4)-80.0*q2*q5*qd1*qd4*sin(q3)*sin(
        q4)+80.0*q5^2*qd1*qd3*cos(q3)*cos(q4)^2*sin(q3)+80.0*q5^2*qd1*qd4*cos(
        q3)^2*cos(q4)*sin(q4);
    352.0*qd2-40.0*q2*qd1^2-700.0*qd1^2+600.0*qd1*qd4*cos(q4)+80.0*qd1*qd5*sin(
        q4)-300.0*qd1^2*cos(q4)*sin(q3)-300.0*qd3^2*cos(q4)*sin(q3)-300.0*qd4
        ^2*cos(q4)*sin(q3)-80.0*qd4*qd5*sin(q3)*sin(q4)-40.0*q5*qd1^2*cos(q4)*
        sin(q3)-40.0*q5*qd3^2*cos(q4)*sin(q3)-40.0*q5*qd4^2*cos(q4)*sin(q3)
        +80.0*q5*qd1*qd4*cos(q4)+80.0*qd3*qd5*cos(q3)*cos(q4)-600.0*qd3*qd4*cos
        (q3)*sin(q4)-80.0*q5*qd3*qd4*cos(q3)*sin(q4);
    -0.0833*cos(q4)*(6.3e+4*qd1^2*cos(q3)-7200.0*qd3*qd5*cos(q4)+7.2e+4*qd3*qd4
        *sin(q4)+3600.0*q2*qd1^2*cos(q3)+8400.0*q5*qd1^2*cos(q3)-960.0*q5*qd3*
        qd5*cos(q4)+1.44e+4*q5*qd3*qd4*sin(q4)+3.6e+4*qd1^2*cos(q3)*cos(q4)*sin
        (q3)+480.0*q2*q5*qd1^2*cos(q3)-7.2e+4*qd1*qd4*cos(q3)*cos(q4)+960.0*q5
        ^2*qd3*qd4*sin(q4)-7200.0*qd1*qd5*cos(q3)*sin(q4)+7200.0*q5*qd1^2*cos(
        q3)*cos(q4)*sin(q3)-1.44e+4*q5*qd1*qd4*cos(q3)*cos(q4)-960.0*q5*qd1*qd5
        *cos(q3)*sin(q4)+480.0*q5^2*qd1^2*cos(q3)*cos(q4)*sin(q3)-960.0*q5^2*
        qd1*qd4*cos(q3)*cos(q4));
    600.0*qd4*qd5-1500.0*qd3^2*sin(2.0*q4+3.14)-6000.0*qd1*qd3*cos(q3)-600.0*
        qd1*qd5*sin(q3)-20.0*q5^2*qd3^2*sin(2.0*q4+3.14)-5250.0*qd1^2*cos(q4
        +1.57)*sin(q3)+80.0*q5*qd4*qd5-300.0*q5*qd3^2*sin(2.0*q4+3.14)-600.0*
        qd1*qd2*sin(q4+1.57)+6000.0*qd1*qd3*cos(q4+1.57)^2*cos(q3)-300.0*q2*qd1
        ^2*cos(q4+1.57)*sin(q3)-700.0*q5*qd1^2*cos(q4+1.57)*sin(q3)-80.0*q5*qd1
        *qd2*sin(q4+1.57)-1200.0*q5*qd1*qd3*cos(q3)-80.0*q5*qd1*qd5*sin(q3)
        +3000.0*qd1^2*cos(q4+1.57)*sin(q4+1.57)*cos(q3)^2-80.0*q5^2*qd1*qd3*cos
        (q3)+600.0*q5*qd1^2*cos(q4+1.57)*sin(q4+1.57)*cos(q3)^2+40.0*q5^2*qd1
        ^2*cos(q4+1.57)*sin(q4+1.57)*cos(q3)^2+1200.0*q5*qd1*qd3*cos(q4+1.57)
        ^2*cos(q3)-40.0*q2*q5*qd1^2*cos(q4+1.57)*sin(q3)+80.0*q5^2*qd1*qd3*cos(
        q4+1.57)^2*cos(q3);
    4070.0*qd5-300.0*qd3^2*cos(q4)^2-40.0*q5*qd1^2-40.0*q5*qd4^2-300.0*qd1
        ^2-300.0*qd4^2-80.0*qd1*qd2*sin(q4)+600.0*qd1*qd4*sin(q3)+300.0*qd1^2*
        cos(q3)^2*cos(q4)^2-40.0*q5*qd3^2*cos(q4)^2-700.0*qd1^2*cos(q4)*sin(q3)
        +40.0*q5*qd1^2*cos(q3)^2*cos(q4)^2-40.0*q2*qd1^2*cos(q4)*sin(q3)+80.0*
```

```
        q5*qd1*qd4*sin(q3)-600.0*qd1*qd3*cos(q3)*cos(q4)*sin(q4)-80.0*q5*qd1*
        qd3*cos(q3)*cos(q4)*sin(q4)];

    Ga_ne=[0;0;392.0*cos(q4)*sin(q3)*(q5+7.5);392.0*cos(q3)*sin(q4)*(q5+7.5)
        ;-392.0*cos(q3)*cos(q4)];
elseif modelo==2

    Ma_ne=[1140.0*q2^2+1.02e+4,0,0,0,0;
    0,1140.0,0,0,0;
    0,0,0,0,0;
    0,0,0,0,0;
    0,0,0,0,0.027];

    Va_ne=[0.001*qd1*(2.28e+6*q2*qd2+191.0);
    -1140.0*q2*qd1^2+0.042*qd2;
    0;
    0;
    0.13*qd5];

    Ga_ne=[0;0;0;0;0];
elseif modelo==3
    %Se definen Ma_ne y Va_ne

    Ga_ne=[0;0;2.0e+4*cos(q4)*sin(q3)*(q5+7.5);2.0e+4*cos(q3)*sin(q4)*(q5+7.5)
        ;-2.0e+4*cos(q3)*cos(q4)];

end
Ma_ne=double(Ma_ne);
Va_ne=double(Va_ne);
Ga_ne=double(Ga_ne);

% Ma, Ga y Va provienen del cálculo iterativo de Newton-Euler
% Aceleraciones
  qdd=Ma_ne\(T-Ga_ne-Va_ne);
  qdd=round(qdd,2);

end
```

A.6 Códigos utilizados para el diseño y comprobación del control dinámico

Código A.21 Extracción de las funciones de transferencia de las articulaciones partiendo de la linealización física y desacoplamiento.

```
syms s k Ti Td q5 q4 q3 q2 q1 qd1 qdd1 qd2 qdd2 qd3 qdd3 qd4 qdd4 qd5 qdd5 g
    real

% Se define Ma_ne y Va_ne

Ga_ne =[
0;
0;
392.0*cos(q4)*sin(q3)*(q5+7.5);
392.0*cos(q3)*sin(q4)*(q5+7.5);
-392.0*cos(q3)*cos(q4)];
```

```
%%%%%%%%%%----- Se sustituyen los valores que se conocen y son deseados
    -----%%%%%%%%%%
Ma_ne= subs(Ma_ne,q3,0);
Ma_ne= subs(Ma_ne,q4,0);
Va_ne= subs(Va_ne,q3,0);
Va_ne= subs(Va_ne,q4,0);
%%%%%%%%%%----- Diagonalización de Ma_ne -----%%%%%%%%%%
Ma_ne=diag(Ma_ne);
%%%%%%%%%%----- Dependencia de la qdi únicamente -----%%%%%%%%%%
Va_ne= [ diff(Va_ne(1),qd1);
diff(Va_ne(2),qd2);
diff(Va_ne(3),qd3);
diff(Va_ne(4),qd4);
diff(Va_ne(5),qd5)];
%%%%%%%%%%----- Anulación de Ga_ne -----%%%%%%%%%%
Ga_ne = zeros(3,1);
%%%%%%%%%%----- Maximizar Ma_ne -----%%%%%%%%%%
Ma_ne= subs(Ma_ne,q2,14);
Ma_ne = double([Ma_ne(1); Ma_ne(2); 0; 0; Ma_ne(5)]);
Va_ne = [1700; 352; 0; 0; 4070];
Tau1= Ma_ne(1)*qdd1+ Va_ne(1)*qd1;
Tau2= Ma_ne(2)*qdd2+ Va_ne(2)*qd2;
Tau3= Ma_ne(5)*qdd5+ Va_ne(5)*qd5;
Tau1= Ma_ne(1)*q1*s^2+ Va_ne(1)*q1*s;
Tau2= Ma_ne(2)*q2*s^2+ Va_ne(2)*q2*s;
Tau3= Ma_ne(5)*q5*s^2+ Va_ne(5)*q5*s;
G11= tf(1,[Ma_ne(1), Va_ne(1), 0])
G22= tf(1,[Ma_ne(2), Va_ne(2), 0])
G33= tf(1,[Ma_ne(5), Va_ne(5), 0])
```

Código A.22 Extracción del espacio de estados del sistema linealizado.

```
clear all;
syms Tau1 Tau2 Tau5 q1 q2 q3 q4 q5 qd1 qd2 qd3 qd4 qd5 qdd1 qdd2 qdd3 qdd4 qdd5
     real

x = [q1 q2 q3 q4 q5 qd1 qd2 qd3 qd4 qd5]';
T = [Tau1 Tau2 0 0 Tau5]';
u = [Tau1 Tau2 Tau5]';
g=9.81;

% Se define la inversa de Ma_ne calculada externamente y Va_ne

Ga_ne =[
                          0
                          0
392.0*cos(q4)*sin(q3)*(q5 + 7.5)
392.0*cos(q3)*sin(q4)*(q5 + 7.5)
      -392.0*cos(q3)*cos(q4)];


qdd = Ma_inv*simplify((T-Ga_ne-Va_ne));

xdot = [qd1;
        qd2;
        qd3;
```

```
        qd4;
        qd5;
        qdd];

A = jacobian(xdot,x);
B = jacobian(xdot,u);

A=subs(A,[Tau1 Tau2 Tau5 q1 q2 q3 q4 q5 qd1 qd2 qd3 qd4 qd5 qdd1 qdd2 qdd3 qdd4
     qdd5], [zeros(1,3),pi/5, 14, pi/18, pi/18, 24, zeros(1,10)]);

B=subs(B,[Tau1 Tau2 Tau5 q1 q2 q3 q4 q5 qd1 qd2 qd3 qd4 qd5 qdd1 qdd2 qdd3 qdd4
     qdd5], [zeros(1,3),pi/5, 14, pi/18, pi/18, 24, zeros(1,10)]);

format short e

A = double(A);
B = double(B);
C = eye(10);
D = zeros(10, 3);

grua = ss(A, B, C, D);
```

Código A.23 Código para la implementación de PD, PID y LQR.

```
function [Tau] = Control(in)

qr1     = in(1);
qr2     = in(2);
qr3     = in(3);
qr4     = in(4);
qr5     = in(5);
qdr1    = in(6);
qdr2    = in(7);
qdr3    = in(8);
qdr4    = in(9);
qdr5    = in(10);
qddr1   = in(11);
qddr2   = in(12);
qddr3   = in(13);
qddr4   = in(14);
qddr5   = in(15);
q1_      = in(16);
q2_      = in(17);
q3_      = in(18);
q4_      = in(19);
q5_      = in(20);
qd1_     = in(21);
qd2_     = in(22);
qd3_     = in(23);
qd4_     = in(24);
qd5_     = in(25);

 paso=[];

 e1=qr1-q1_; ed1=qdr1-qd1_;
 e2=qr2-q2_; ed2=qdr2-qd2_;
```

```
 e3=qr3-q3_; ed3=qdr3-qd3_;
 e4=qr4-q4_; ed4=qdr4-qd4_;
 e5=qr5-q5_; ed5=qdr5-qd5_;

%% Sin Control aplicado
% Tau= [0; 20; 0];

%% Control PID y PD

% e=[e1;e2;e5];
% ed=[ed1;ed2;ed5];

% PID
% Kc1=61824000; Kc2= 51817; Kc5= 632510;
% c11=1; c12=0.18; c15=0.19;
% c21=1; c22=0.18; c25=0.19;
%
% kp1= Kc1*(c11+c21); Kd1= Kc1*c11*c21; Ki1= Kc1;
% kp2= Kc2*(c12+c22); Kd2= Kc2*c12*c22; Ki2= Kc2;
% kp5= Kc5*(c15+c25); Kd5= Kc5*c15*c25; Ki5= Kc5;

% PD
% kp1= 82481000; Kd1= 55262270; Ki1= 0;
% kp2= 9923.9; Kd2= 12901.07; Ki2= 0;
% kp5= 127130; Kd5= 16526.9; Ki5= 0;
%
% Kp=double([kp1;kp2;kp5]);
% Ki=double([Ki1;Ki2;0;0;Ki5]);
% Kd=double([Kd1;Kd2;Kd5]);
%
% Tau=Kp.*e+Kd.*ed;

%% Control LQR
 e=[e1;e2;e3;e4;e5];
 ed=[ed1;ed2;ed3;ed4;ed5];

 % LQR lineal

   % LQR lento
   %  K= [1.3208e+06 -7.4493e+04 1.4186e+05  1.2930e+06 -1.2941e+04  6.3114e+06
       -1.0265e+05 -3.4820e+06 5.2290e+05  2.6494e+03;
   % 7.4459e+00  1.3208e+02  7.3010e+01 -1.8513e+01  2.7462e+00  4.8111e+01
       5.2249e+01  2.6387e+02   -3.0799e+01  4.4878e+00;
   % 3.2680e+00  1.2742e+00 -1.5982e+02 -5.6226e+01 1.3235e+03 -3.2822e+00
       -1.2309e+01 6.0693e+01   -7.7117e+01  5.5730e+02];

   % LQR rápido

    K= [1.2991e+08  2.3328e+07 -6.5430e+07  7.8538e+07  9.1883e+06  3.4906e+08
        2.7604e+07 -1.4393e+08  -1.1317e+08  1.1862e+07;
  -2.3634e+02  1.3010e+03  2.3956e+02 -2.3369e+02  4.2301e+01 -6.5360e+02  1.2834
      e+03  7.6714e+02   2.4411e+02  5.3088e+01;
  -8.0608e+02 -5.5220e+02 -2.7663e+02 -8.3329e+02 1.3199e+04 -2.3802e+03
      -6.1014e+02 2.6155e+03   9.1274e+02  1.2910e+04];
Tau= K * [e;ed];

Tau= [Tau(1);Tau(2);Tau(3)];
```

```
end
```

Código A.24 Cálculo de la ganancia del control LQR.

```
function K = K_LQR()
    %% LQR

    Extr_A_B_SS;

    Q = diag([700 700 50 50 700 1000 1000 1000 1000 1000]);
    % Q =eye(10);

    % R = 100*diag([1/(5*10^5)^2 1/50^2 1/500^2]);
    R = diag([1/(5*10^6)^2 1/50^2 1/500^2]);
    % R =eye(3);

    K = lqr(A, B, Q, R);

end
```

Código A.25 Definición del objeto de Matlab necesario para el control MPC.

```
%% Control Predictivo por Modelo

Extr_A_B_SS;

Ts = 0.03;  % Tiempo de sampleo
% Ts = 0.5;  % Tiempo de sampleo

% Crear el objeto MPC
mpcobj_pend = mpc(grua, Ts);

% 3. Definir los Horizontes
mpcobj_pend.PredictionHorizon = 100.0; % Cuántos pasos a futuro predice
mpcobj_pend.ControlHorizon = 4;  % Cuántos pasos de control calcula
% mpcobj_pend.PredictionHorizon = 10.0; % Cuántos pasos a futuro predice
% mpcobj_pend.ControlHorizon = 3; % Cuántos pasos de control calcula


% Poner restricciones
mpcobj_pend.MV(1).Min = -5e6; mpcobj_pend.MV(1).Max = 5e6; % Límites motor giro
     (Nm)
mpcobj_pend.MV(2).Min = -500; mpcobj_pend.MV(2).Max = 500; % Límites motor
    carro (N)
mpcobj_pend.MV(3).Min = -50000; mpcobj_pend.MV(3).Max = 50000; % Límites motor
    elevación (N)

% Límites para el carrito (Salida 2)
mpcobj_pend.OV(2).Min = -12;
mpcobj_pend.OV(2).Max = 14;

% Límites para la longitud del cable (Salida 5)
mpcobj_pend.OV(5).Min = -14;
mpcobj_pend.OV(5).Max = 24;
```

```
% Límites para la longitud del cable (Salida 3)
mpcobj_pend.OV(3).Min = -pi/3;
mpcobj_pend.OV(3).Max = pi/3;

% % Límites para la longitud del cable (Salida 4)
mpcobj_pend.OV(4).Min = -pi/3;
mpcobj_pend.OV(4).Max = pi/3;

mpcobj_pend.MV(1).ScaleFactor = 5e6;
mpcobj_pend.MV(2).ScaleFactor = 500;
mpcobj_pend.MV(3).ScaleFactor = 50000;

% Pesos para la salida Equivalente a Q
mpcobj_pend.Weights.OV = [3000 3000 500 500 3000 1000 1000 2000 2000 1000];

% Pesos para el control Equivalente a R
mpcobj_pend.Weights.MV = 0.1*[1 1 1];
% mpcobj_pend.Weights.MV = 100*[1 1 1];

% Pesos para la variación de las señales de control
mpcobj_pend.Weights.MVRate = [150 150 150];
```

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] N. Agudelo, G. Tano, and C. A. Vargas, "Historia de la automatización," *Universidad Ecci*, 2020. (Citado en la pág. 1)

[2] J. Lassébie and G. Quintini, "What skills and abilities can automation technologies replicate and what does it mean for workers?: New evidence," OECD Publishing, Paris, OECD Social, Employment and Migration Working Papers 282, 2022. [Online]. Available: https://doi.org/10.1787/646aad77-en (Citado en la pág. 1)

[3] McKinsey Global Institute, "Un futuro que funciona: Automatización, empleo y productividad," McKinsey & Company, Resumen ejecutivo, enero 2017. [Online]. Available: https://www.mckinsey.com/featured-insights/digital-disruption/harnessing-automation-for-a-future-that-works (Citado en la pág. 2)

[4] Royal Institution of Chartered Surveyors, "Global Construction Monitor Q4 2023," RICS, London, Market Survey, 2024. [Online]. Available: https://www.rics.org/news-insights/market-surveys/global-construction-monitor (Citado en la pág. 2)

[5] S. Parascho, "Construction robotics: From automation to collaboration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 6, no. 1, pp. 183–204, 2023. [Online]. Available: https://doi.org/10.1146/annurev-control-080122-090049 (Citado en la pág. 2)

[6] ICON Technology, Inc. (2026) Icon: Architecture for humanity and building technology. [Online]. Available: https://www.iconbuild.com/ (Citado en las págs. 2, 3, and 151)

[7] FBR Limited. (2026) Hadrian X. [Online]. Available: https://www.fbr.com.au/view/hadrian (Citado en las págs. 3 and 151)

[8] Okibo Ltd. (2026) Autonomous robots for the construction site. [Online]. Available: https://okibo.com/ (Citado en las págs. 3, 4, and 151)

[9] Dorce Prefabricated Building and Construction. (2026) Perfil de empresa. [Online]. Available: https://www.dorce.com/es/perfil-de-empresa-2/ (Citado en las págs. 4 and 151)

[10] A. Barrientos, *Fundamentos de robótica*, 2nd ed. Madrid: McGraw-Hill España, 2012, [En Línea]. [Online]. Available: https://elibro.net/es/ereader/bibliotecaus/50193?page=164 (Citado en las págs. 5, 16, 20, and 22)

[11] J. C. Cortés, "La representación de Denavit-Hartenberg," Material docente, Universidad de Sevilla, marzo 2008. [Online]. Available: https://personal.us.es/jcortes/Material/Material.htm (Citado en la pág. 7)

[12] G. G. Slabaugh, "Computing euler angles from a rotation matrix," Aug. 1999, technical report, Queen Mary University of London. [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=es&user=oUK2gu8AAAAJ&citation_for_view=oUK2gu8AAAAJ:2P1L_qKh6hAC (Citado en la pág. 9)

[13] A. M. Ramírez, "Diseño y cálculo de una grúa torre," 2011, acceso abierto. [Online]. Available: https://upcommons.upc.edu/entities/publication/094cab97-9299-4d1d-9628-bf159621843a (Citado en las págs. 12, 30, and 73)

[14] H. L. Thi, V. C. Nguyen, and T. L. Nguyen, "Mathematical model," in *Adaptive finite-time extended state observer-based model predictive control with Flatness motivated trajectory planning for 5-DOF tower cranes*. Elsevier, 2025, vol. 81, p. Section 2. (Citado en las págs. 28, 30, and 34)

[15] "Momento y tensor de inercia (CMR)," https://tesla.us.es/wiki/index.php/Momento_y_tensor_de_inercia_(CMR), Wiki del Departamento de Física Aplicada III, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla. (Citado en la pág. 29)

[16] Bonfiglioli Riduttori S.p.A., *Motorreductores coaxiales, de ejes paralelos y ortogonales: Serie C, A, F, S*, Bonfiglioli, Bologna, Italia, catálogo técnico BR_CAT_CAFS_STD_SPA_R11_1. [Online]. Available: https://www.bonfiglioli.com/international/en/product/a-series_industrial-gearmotors_right-angle-gear-units#download (Citado en la pág. 31)

[17] D.-J. Kim, J.-H. Choi, Y.-D. Chun, D.-H. Koo, and P.-W. Han, "The study of the stray load loss and mechanical loss of three phase induction motor considering experimental results," *Journal of Electrical Engineering and Technology*, vol. 9, no. 1, pp. 121–126, 2014. [Online]. Available: https://www.koreascience.kr/article/JAKO201401671902952.page (Citado en la pág. 31)

[18] F. G. Alvarez, "Control óptimo," Ph.D. dissertation, Universidad de Sevilla. (Citado en la pág. 58)

[19] Nocoń, M. Grzyb, P. Szmidt, Z. Koruba, and Nowakowski, "Control analysis with modified LQR method of anti-tank missile with vectorization of the rocket engine thrust," *Energies*, vol. 15, no. 1, p. 356, 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/1/356 (Citado en la pág. 61)

[20] E. Camacho and C. Bordons, "Control predictivo: Pasado, presente y futuro," *Revista iberoamericana de automatica e informatica industrial (RIAI), ISSN 1697-7912, Vol. 1, Nº. 3, 2004*, vol. 1, 10 2010. (Citado en la pág. 65)

[21] D. Martínez Ribes, "Diseño y cálculo de la estructura de una grúa pórtico de 50 t de capacidad y 50 m de luz," Trabajo de Fin de Grado, Universitat Jaume I, Castellón de la Plana, julio 2016. (Citado en la pág. 72)