Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Análisis cinemático y dinámico de una plataforma de Stewart.

Escuela Técnica Superior de Ingeniería de Sevilla

Autor: Juan Carlos Sánchez Palomino
Tutora: Juana María Mayo Nuñez

Dpto. de Ingeniería Mecánica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Ingeniería Aeroespacial

Escuela Técnica Superior de Ingeniería de Sevilla

Autor:
Juan Carlos Sánchez Palomino

Tutora:
Juana María Mayo Núñez
Catedrática de Universidad

Dpto. de Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Grado: Escuela Técnica Superior de Ingeniería de Sevilla

Autor: Juan Carlos Sánchez Palomino

Tutora: Juana María Mayo Núñez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A Pilar y Juan Carlos, os quiero.

Agradecimientos

Es curioso como un duro camino como este ablanda a las personas. Jamás me imaginaba abriéndome de esta manera para resaltar a todas aquellas personas que han hecho posible que hoy esté aquí, bien por su apoyo o bien por su “no eres capaz”.

En primer lugar, mi infinita gratitud a mi madre, Pilar, por ser la principal fuente de amor, cariño y apoyo que siempre he tenido y que siempre tendré. Eres mi razón de ser, un referente en mi vida que me ha enseñado lo más importante: a ser humano. No existen palabras suficientes en el mundo para expresarte cuánto te quiero y cómo de necesaria eres para mí, ni existen años suficientes en un milenio para hacerte un agradecimiento justo. No me faltes nunca. Te quiero.

En segundo lugar, agradecer a mi padre, Juan Carlos, su tesón, esfuerzo y sacrificio diario para que consiga mis objetivos. La vida no se te ha presentado fácil y sin embargo eso nunca te ha impedido dárnoslo todo. Por tu profesionalidad y tu buen hacer, esta victoria es tan mía como tuya.

Agradecer también a mi hermana, Cristina, todo su cariño y confidencia. Fuiste una segunda madre para mí y tu orgulloso hermano te dedica sus logros por ser parte esencial de ellos.

Tampoco puedo dejar de agradecer a mis amigos, mi segunda familia, todo lo que, sin ellos saberlo, han hecho por mí. Sois otra razón más por la que levantarme feliz cada día. Me gustaría hacer una mención especial a mis dos pilares, Celia y Antonio. Para toda la vida.

A aquellos del “no eres capaz”: gracias también por enseñarme que vida sólo hay una, y que no podemos vivirla al gusto de los demás.

Por último, gracias a mi tutora Juana por su ayuda, dedicación, paciencia y profesionalidad, virtudes que forman a una gran docente y referente en su profesión.

Gracias también a la Universidad Pública, que tantos sueños hace realidad.

Índice

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Este proyecto tiene como fin profundizar en el estudio de la cinemática y dinámica de sistemas multicuerpos. En particular, se analizará la cinemática y la dinámica inversa, además de realizar la simulación cinemática del mecanismo de una Plataforma de Stewart.

La definición académica de un sistema multicuerpo es: "una colección de sólidos rígidos o deformables conectados por pares cinemáticos (pares de revolución, prismáticos, …), suspensiones (muelles y amortiguadores) y actuadores (motores eléctricos, cilindros hidráulicos, …)". Como se expresa anteriormente, en el presente proyecto tiene como objetivo principal el análisis cinemático y dinámico de dicha plataforma, entendiéndose el análisis cinemático como el estudio de la posición y el movimiento de sistemas, independientemente de las fuerzas que puedan actuar sobre dicho mecanismo. Por el contrario, el análisis dinámico inverso consiste en, conocido el movimiento, calcular las fuerzas que son necesarias para reproducir dicho movimiento.

Se estructuran a continuación un total de 4 capítulos. En el primer capítulo se expone una introducción para poner al lector en contexto, introduciéndolo en el mundo de los manipuladores paralelos, familia a la cual pertenece la plataforma de Stewart. Se explicará en que consiste exactamente un manipulador en serie y uno paralelo, explicando las características de cada uno y las diferencias entre ellos, además de algunas aplicaciones y ventajas de uno frente a otro.

Posteriormente se da paso a un capítulo extenso de fundamentos teóricos, donde el primer paso será dar una breve introducción a la plataforma de Stewart, mostrando algunas partes de su historia y origen, así como algunos ingenieros y científicos cuya aportación fue importante para su creación. Seguidamente se desarrollan dos subapartados de cinemática 2D y 3D, donde se detallan los aspectos más importantes y necesarios para conseguir la implementación de este mecanismo con éxito, mediante el uso de coordenadas de referencia, explicadas más adelante. Además, se detallará también el estudio de la dinámica inversa, tanto en 2D como en 3D, y estos conocimientos serán aplicados en el mecanismo biela manivela de ejemplo y en la propia plataforma de Stewart. Por último, se expone un tercer capítulo con los resultados obtenidos de los diferentes análisis llevados a cabo y un último capítulo con los códigos creados.

1.1 Manipuladores.

1.1.1 Manipuladores en serie y paralelos.

Los robots se pueden clasificar, según su estructura, en robots tipo serie y robots o manipuladores paralelos. Los robots tipo serie están formados por una cadena cinemática abierta, similar a la del brazo humano. Por el contrario, los robots o manipuladores paralelos están compuestos por dos plataformas, una fija y otra móvil, unidas en varias cadenas cinemáticas en paralelo, y por ello, formando cadenas cinemáticas cerradas.

Los robots en serie han sido utilizados a lo largo de la historia en numerosas aplicaciones, y raro es el proceso industrial automatizado que no lleve relacionado algún proceso de ensamblaje o automatización con robots en serie. Por el contrario, los robots paralelos son más modernos y usados en menor medida.

El primer enfoque dado a los robots tipo serie es el de simular un brazo humano, para así automatizar con mayor facilidad tareas que anteriormente eran manuales. Estos manipuladores tienen la ventaja

de tener un amplio espacio de trabajo, mientras que los robots paralelos no. Sin embargo, los robots tipo serie no son capaces de soportar altos pesos de trabajo, provocando vibraciones cuando trabajan a grandes velocidades y pesos. Por tanto, para tareas cuyo objetivo sea la precisión, rapidez y el trabajo con una carga mayor los robots paralelos resultan ideales. Los actuadores paralelos frecuentemente se sitúan en la base fija del mecanismo, y por ello, son fijos, mientras que en los robots en serie dichos actuadores se sitúan en las articulaciones del mecanismo, lo que aumenta a su vez el riesgo de sufrir averías, debido al desplazamiento de los motores y el aumento de las fuerzas de inercia.

La creciente demanda de la industria a la hora de realizar tareas de mayor complejidad y flexibilidad ha ido dando lugar al estudio de mecanismos y manipuladores de más de un grado de libertad que permitan simular cualquier tipo de movimiento en el espacio. Sin ir más lejos, la plataforma de Stewart tratada en este proyecto, consta de 6 grados de libertad, lo que permite el libre movimiento de su plataforma móvil.

1.1.2 Manipuladores paralelos.

Un mecanismo robótico es un sistema compuesto por diferentes solidos rígidos unidos entre sí mediante articulaciones. Cuando estos sólidos están unidos entre sí siguiendo un único camino se denominan manipuladores en serie, mientras que si estos están conectados mediante dos o más caminos se denominan en paralelo.

De este modo, se puede definir un manipulador paralelo como un mecanismo conformado por una base fija y otra móvil, unidas entre sí mediante varios actuadores o cadenas cinemáticas en serie. El número máximo de grados de libertad de este tipo de manipuladores ha de ser seis ya que se necesitan tres coordenadas de translación y tres de rotación independientes para definir la posición del efector final.

Un manipulador paralelo que se ha demostrado con potencial a lo largo de los últimos años es la Plataforma de Stewart: formada a partir de seis actuadores lineales que soportan una base móvil y la separan de la fija.

Aunque los manipuladores paralelos son relativamente modernos, estos llevan siendo estudiados desde hace ya décadas. Su historia comienza en los años 30, concretamente en 1931, cuando Grwinnet patentó una plataforma para ubicar los asientos de un teatro, con el fin de proporcionar una sensación más realista al espectáculo; sin embargo, de acuerdo con la información disponible, esta nunca llegó a construirse. Ya en 1942, Gough diseñó y construyó un manipulador paralelo con 6 grados de libertad, utilizado por la empresa Dunlop para el ensayo de neumáticos. Tras esto, se han seguido implementando diseños de manipuladores paralelos hasta que, en 1965, Stewart patentó una plataforma de 6 grados de libertad, a la cual le dio su nombre, utilizada en principio para simuladores de vuelo[5].

1.1.3 Aplicaciones.

Dadas las ventajas económicas y de operatividad que este tipo de manipuladores tenía y sigue teniendo, no se tardó en implantar dicha idea de la plataforma de Stewart en diferentes sectores de la industria. Por ejemplo, en la empresa Volkswagen se diseñó un simulador de automóvil de 3 grados de libertad a principios de los años setenta. Posteriormente, la empresa Ford, introdujo el Virttex, con 6 grados de libertad. Posteriormente la NADS mostró en la Universidad de Iowa su simulador de 6 gdl. También la empresa Renault desarrolló un robot con 6 GDL sobre una base que se desplaza en los ejes XY. Uno de los últimos desarrollados en el ámbito industrial fue el elaborado por Toyota, con un diseño similar al de la NADS, pero más largo. [4]

En el área de la manufactura y el mecanizado de piezas, se tiene el caso de robots como el Variax, VOH-1000, el robot Tornado, así como fresadoras, taladradoras y otras máquinas.

En soldadura y ensamble, destaca la empresa FANUC en la industria automotriz. En electrónica, la empresa Physik Instrumente creó el robot F-206, usado para el ensamble de dispositivos electrónicos, manipulación de semiconductores y prueba de elementos ópticos. También en medicina, particularmente en el área de cirugía. Uno de los más conocidos es el SurgiScope, que se ha usado en el Laboratorio de Robots Quirúrgicos de dicha universidad y fue desarrollado para neurocirugía. Otro robot empleado en el área de medicina es el MARS, usado como posicionador de alta precisión para hacer taladrados en cirugía a nivel intramedular. [4]

Imagen 1.1: Simulador NADS, Universidad de Iowa.

Todo este despliegue tecnológico también ha sido utilizado, como no, por la industria aeronáutica, donde se han creado simuladores de vuelo sofisticados basados en los mecanismos de la plataforma. Un ejemplo lo tenemos en el Airbus Military Training Center de Sevilla, donde conviven los simuladores de vuelo del A400M y el C295, dos aviones militares ensamblados también en Sevilla.

1.1.4 Ventajas y desventajas de los manipuladores paralelos frente a manipuladores en serie.

Como ya se explicó anteriormente, un manipulador paralelo está compuesto por una plataforma móvil y otra fija, unidas entre sí por cadenas cinemáticas. Normalmente, estas cadenas tienen algún tipo de accionamiento (eléctrico, hidráulico…). También se puede dar el caso de que dichos accionamientos se encuentren en la base fija, todo dependiendo de la aplicación futura de dicho mecanismo. En el presente proyecto, se hará uso de una plataforma de Stewart cuyos actuadores son lineales.

Una de las primeras ventajas de los manipuladores paralelos frente a los manipuladores en serie es que permiten trabajar con un mayor número de grados de libertad, normalmente 5 o 6, aunque existen también mecanismos de 3 grados de libertad para otorgarles mayor ligereza.

Además, para tareas correspondientes a manipulación de objetos, los actuadores paralelos poseen una mayor precisión, velocidad y rigidez. La relación carga/peso de un manipulador paralelo es también muy superior al de un manipulador en serie.

Sin embargo, también poseen fuertes desventajas. Los manipuladores paralelos poseen una cinemática compleja, unos requisitos de computación mayores y una dificultad de control añadida respecto a los manipuladores en serie.

El uso de los manipuladores paralelos es relativamente moderno si lo comparamos con los manipuladores en serie, lo que implica que la tecnología de estos últimos está más desarrollada e implementada que la de los robots paralelos, los cuales requieren una mayor inversión para seguir desarrollándolos.

1.2 Objetivos del Proyecto.

El objetivo fundamental de este proyecto es la implementación y modelado de una plataforma de Stewart con actuadores lineales mediante un sistema de referencia definido previamente, y su posterior análisis cinemático y dinámico.

Para entender los fundamentos de la cinemática y dinámica 3D es necesario tener destreza en el estudio cinemático y dinámico 2D utilizando coordenadas de referencia. Para ello se realizarán cálculos y análisis a un mecanismo simple de biela manivela, con el objetivo de profundizar en la cinemática y dinámica de sistemas multicuerpos.

2 FUNDAMENTOS TEÓRICOS

"Aquel que ama la práctica sin la teoría es como un marinero que navega sin timón ni brújula y nunca sabe a dónde ir."

Leonardo Da Vinci.

En este apartado se desarrollan todos los fundamentos teóricos necesarios y utilizados para la realización del presente proyecto, desde un breve recorrido por la historia de la Plataforma de Stewart, su diseño conceptual, aplicaciones que actualmente tiene, pasando por la definición del modelo. Por último, se explicarán los fundamentos teóricos en los que se ha basado dicho proyecto en cuanto a análisis computacional: se definirá los diferentes pares que este mecanismo posee y cómo se modelan dichos pares en función del sistema de referencia escogido.

2.1. Plataforma de Stewart.

2.1.1 Historia de la Plataforma de Stewart

Como se ha explicado en anteriores apartados, la plataforma de Stewart es un mecanismo de 6 grados de libertad compuesto por dos plataformas, una móvil y otra fija, unidas entre sí por 6 actuadores que forman entre ellos una estructura triangular, y que están unidos a las plataformas mediante pares Cardan. Para mayor claridad, se muestra la figura a continuación.

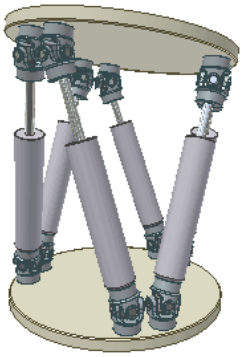


Imagen 2.1: Plataforma de Stewart. Imagen extraída de Wikipedia.

Este mecanismo resulta ideal a la hora de intentar imitar movimientos de diferentes transportes, por ejemplo, un avión o un automóvil ya que los actuadores, dependiendo de cómo se extiendan y contraigan, pueden otorgar cualquier posición a la plataforma móvil, situada en la parte superior.

En 1965, Stewart publicó un documento en el cual describía un mecanismo capaz de realizar simulaciones de vuelo para el entreno de los pilotos. Como se nombra en capítulos anteriores, Stewart no es el pionero en manipuladores paralelos. Por ejemplo, Gaugh ya presentó uno en 1947, y fue este en persona uno de los evaluadores del artículo de Stewart.

Tampoco se considera que Gaugh sea el pionero en este tipo de mecanismos. Dicha invención se otorga a Cauchy, quien hablaba de un "octaedro articulado" [5].

2.1.2 Diseño conceptual de la Plataforma de Stewart.

La plataforma de Stewart puede adoptar diferentes formas dependiendo de la función que esta vaya a desempeñar. Entran en juego aspectos como con qué cargas se va a trabajar, cual es el espacio de trabajo necesario, qué grado de rigidez demanda, la precisión requerida… Dependiendo de todos estos aspectos la geometría de dicha plataforma adopta configuraciones de lo más diversas.

Este diseño podría complicarse tanto como se quisiese, pero ya que este documento tiene principalmente un fin académico se opta por realizar una serie de simplificaciones, de manera que facilite su futura implementación numérica, así como su modelaje.

Se opta así por una plataforma de Stewart compuesta por dos bases circulares, cuyos eslabones se distribuyen equidistantes en su borde, separados entre sí 120 grados, y desfasados entre ambas plataformas 60 grados.

Por lo tanto, los eslabones que se encuentran más cerca están separados un ángulo de 30º, mientras que entre los eslabones consecutivos que están más separados existen 90º de ángulo. Así, entre eslabones no consecutivos existe un ángulo total de 120º. Por último, la plataforma móvil será idéntica a la fija, pero desfasada con respecto a esta un ángulo de 60º, lo que provocará que las barras se encuentren inclinadas.

Una vez explicada la configuración geométrica de la plataforma, se pasa al desarrollo de los fundamentos teóricos de cinemática 2D necesarios para poder entender la implementación cinemática de cualquier mecanismo cinemático. Se comienza explicando los diferentes sistemas de referencias con los que se puede trabajar, así como los pares de restricción existentes y como modelar estas restricciones. Posteriormente se explican cómo resolver el problema de posición, velocidad y aceleración de manera numérica.

Una vez explicada y desarrollada la cinemática 2D, se expondrá la analogía en mecanismos 3D, destacando aquellas peculiaridades y herramientas que sólo se usan a la hora de la cinemática 3D.

Por último, para cada caso se hará un ejemplo. Para el 2D se realizará un ejemplo de un mecanismo biela-manivela y para el 3D, la plataforma de Stewart. Cabe destacar que se explicará cómo modelar las ecuaciones de restricción de los pares involucrados en dichos ejemplos.

2.2. Cinemática 2D

El primer paso realizado en este proyecto es entender profundamente los conceptos del análisis cinemático de sistemas multicuerpos en 2D, es decir, movimiento plano. [3]

La primera decisión que se tuvo que tomar al comenzar el proyecto fue decidir qué sistema de referencia sería el idóneo para la simulación cinemática de una Plataforma de Stewart. Para ello, se exponen más adelante y brevemente las opciones planteadas.

Dichas coordenadas serán elegidas no de manera arbitraria, sino que tendrán que estar establecidas de una manera concreta para que posteriormente satisfagan las ecuaciones de restricción. Es por esto que se dice que dichas coordenadas son dependientes.

Además de dichas coordenadas existen otras magnitudes denominadas parámetros, que se mantienen constantes a diferencia de las coordenadas, que van variando durante el movimiento del mecanismo. Ejemplos de parámetros son, por ejemplo, las longitudes de las barras en un mecanismo.

Una vez explicado esto, se procede a exponer los diferentes sistemas de coordenadas planteados.

2.2.1 Selección de coordenadas.

2.2.1.1 Coordenadas Lagrangianas.

Las coordenadas lagrangianas son un conjunto de magnitudes, distancias o ángulos, que indican la posición y orientación de cada uno de los sólidos con respecto a la barra fija. En este sentido, se dicen que son coordenadas absolutas.

Dicho conjunto de coordenadas se agrupa en una matriz columna denominada **q**.

Ejemplo: Sea un mecanismo biela-manivela definido por sus ángulos $\varphi, \theta \; y \; s$ según se muestra en la siguiente figura:

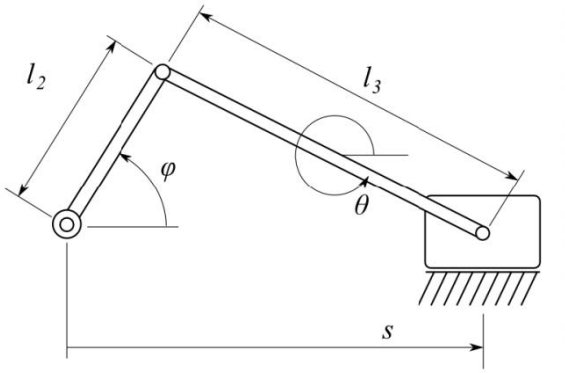


Imagen 2.2: Mecanismo biela-manivela con coordenadas Lagrangianas. Imagen extraída de [3].

El conjunto de coordenadas quedaría definido como:

$$\mathbf{q} = \begin{bmatrix} \varphi \\ \theta \\ s \end{bmatrix}$$

Estas coordenadas deben cumplir las condiciones de la Teoría de Máquinas para poder establecer las ecuaciones de lazo, que son las siguientes:

$$l_2 \cos\varphi + l_3 \cos\theta - s = 0 \qquad (1)$$

$$l_2 \operatorname{sen}\varphi + l_3 \operatorname{sen}\theta = 0 \qquad (2)$$

2.2.1.2 Coordenadas relativas.

Las coordenadas relativas son coordenadas más libres de escoger en el caso de mecanismos de

cadena abierta. Por el contrario, esto produce que la definición de las ecuaciones de restricción sea más compleja. En el ejemplo anterior escogemos estas coordenadas:

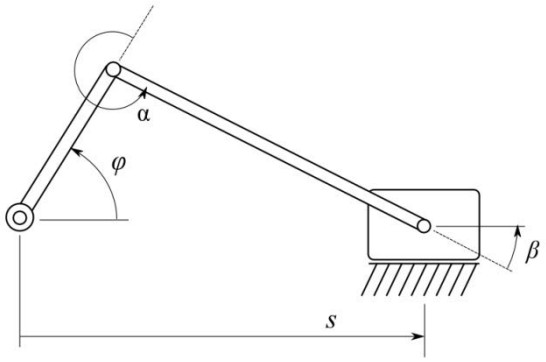


Imagen 2.3: Mecanismo biela-manivela con coordenadas relativas. Imagen extraída de [3].

En este caso tenemos cuatro coordenadas definidas porque en este tipo de coordenadas se escoge una coordenada por cada grado de libertad relativo que cada sólido tiene con el anterior, considerando la barra fija como parte del mecanismo.

Por tanto, dado que el número de grados de libertad está definido y es 1, el número de ecuaciones de restricción necesariamente ha de ser 3. Las dos primeras definen la misma condición que en el apartado anterior, mientras que la tercera expresa que la suma de los tres ángulos ha de ser 360º:

$$l_2 \cos\varphi + l_3 \cos(\varphi + \alpha) - s = 0 \quad (3)$$

$$l_2 \sin\varphi + l_3 \cos(\varphi + \alpha) = 0 \quad (4)$$

$$\varphi + \alpha + \beta - 2\pi = 0 \quad (5)$$

Este tipo de coordenadas son especialmente útiles cuando se trabaja con mecanismos robots de cadena abierta.

2.2.1.3 Coordenadas de referencia.

Este tipo de coordenadas son las que se utilizarán en el presente proyecto debido a que son las más sistemáticas de implementar numéricamente a la hora de modelar un sistema multicuerpo. Utilizando el mismo ejemplo anterior, en este caso sus coordenadas serían las presentadas en la siguiente figura:

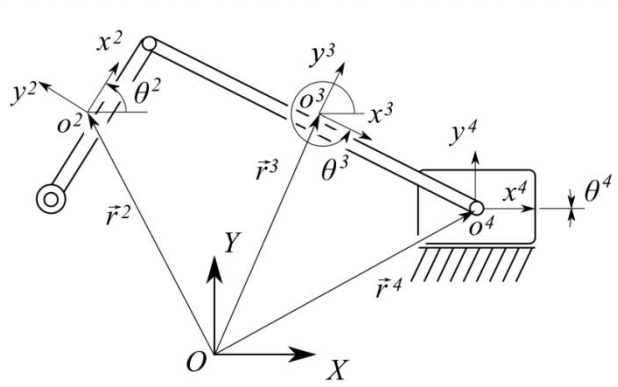


Imagen 2.4: Mecanismo biela-manivela en coordenadas de referencia. Imagen extraída de [3].

Como se aprecia en la imagen, este mecanismo consta de un sistema de referencia global y tres sistemas de referencia locales, uno para cada barra móvil. Las coordenadas de referencia son 3: x, y el ángulo, medido en sentido anti horario, desde la horizontal hasta el eje x, como muestra la imagen 2.4. En el caso tridimensional se contará no con 3, sino mínimo con 6 coordenadas. Existen coordenadas (como el ángulo de la corredera) cuyos valores no varían, pero esto se hace así para que la tarea de implementación numérica sea sistemática.

Por tanto, tendríamos en total 3*n coordenadas, siendo n el número de barras móviles, y como este mecanismo es de un grado de libertad serán necesarias 8 ecuaciones de restricción. Claramente, este tipo de coordenadas aumenta considerablemente el número de restricciones a imponer para un mecanismo, por lo que resulta tedioso para resolver a mano. Sin embargo, a la hora del modelado computacional de sistemas multicuerpos esto permite una sistematización que hace posible el estudio de mecanismos más complejos.

Por último, es esencial definir la diferencia que existe entre el sistema de referencia local y el global. Todas las coordenadas **q** de un sólido deberán estar según el sistema de referencia global establecido, de modo que para pasar del sistema de referencia local de cada barra al global es necesario una matriz de rotación que relacione ambos sistemas de coordenadas. En el caso 2D esta matriz de rotación es la siguiente:

$$\mathbf{A} = \begin{bmatrix} \cos(\theta) & -\mathrm{sen}(\theta) \\ \mathrm{sen}(\theta) & \cos(\theta) \end{bmatrix}$$

Siendo θ el ángulo que forman ambos sistemas de referencia, medido desde el eje X del sistema global al eje x del sistema local en sentido antihorario.

2.2.1.4 Coordenadas naturales.

Las coordenadas naturales constituyen una herramienta que se encuentra a medio camino entre las relativas y las de referencia. Por tanto, nos permiten ser más sistemáticos que con las coordenadas relativas, pero menos que con las de referencia. En el ejemplo de la biela-manivela, un sistema de coordenadas útil sería:

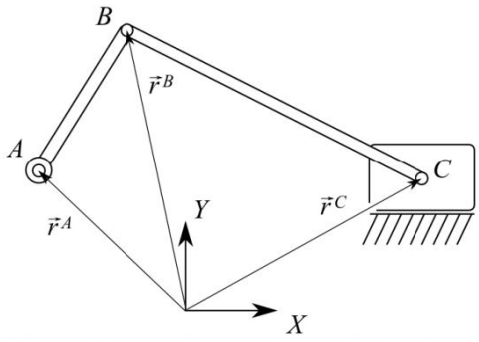


Imagen 2.5: Mecanismo biela-manivela con coordenadas naturales. Imagen extraída de [3].

Se muestran tres vectores que indican la posición de tres puntos característicos del mecanismo A,B y C. Por lo tanto, se tienen en total 6 coordenadas y se necesitan 5 ecuaciones de restricción. Dichas ecuaciones representan que el punto A nunca cambia de posición, que el punto C siempre tiene la misma coordenada en el eje Y global, y que las distancias entre los puntos A y B y B y C permanecen constantes durante el movimiento. Cabe destacar que en coordenadas naturales no se usan ángulos.

2.2.2 Simulación cinemática de un mecanismo manivela-biela-corredera.

Previo al desarrollo de la Plataforma de Stewart, con el objetivo de afianzar los conocimientos adquiridos se llevó a cabo la implementación de un mecanismo simple de manivela-biela-corredera como el de las imágenes de los sistemas de referencias.

Este mecanismo está compuesto por 3 sólidos más la barra fija, tres pares de revolución y un par prismático. Los pares de revolución restringen las dos translaciones y sólo permite un giro y el par prismático restringe una translación y un giro. Si se calcula el número de grados de libertad del mecanismo observamos que este mecanismo sólo tiene un grado de libertad, es decir, con imponer un movimiento de alguna barra se tendría completamente definido el movimiento del mecanismo completo.

$$\mathrm{GDL} = 3(\mathrm{N}-1) - 2\mathrm{r} - 2\mathrm{p} = 1 \qquad (6)$$

Donde N=4 barras, r=3 pares de rotación y p=1 par prismático. Se procede a continuación a explicar cada par por separado.

2.2.2.1 Restricciones debidas a un par de revolución.

El par de revolución consiste en la unión de dos sólidos a través de un punto C tal y como se muestra en la figura:

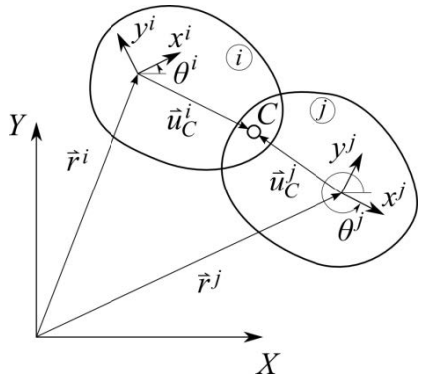


Imagen 2.6: Representación general de un par de revolución en coordenadas de referencia. Imagen extraída de [3].

En este par se tienen los sólidos i y j, cada uno con su sistema de referencia local. Las restricciones pasan por imponer que este punto C siempre pertenece a ambos sólidos de la siguiente manera.

$$\mathbf{r^i} + \mathbf{A_i}\mathbf{u_C^i}' = \mathbf{r^j} + \mathbf{A_j}\mathbf{u_C^j}' \qquad (7)$$

Esta ecuación vectorial, como se puede observar fácilmente, corresponde a dos ecuaciones escalares, de manera que ya quedan restringidos los dos grados de libertad del par de revolución. Los vectores **r** corresponden a la posición (coordenada x e y) de los dos orígenes de los sistemas de referencia locales de ambos sólidos vistos desde el sistema global. Los vectores $\boldsymbol{u_C}$ son el vector que une cada sistema de referencia local con el punto C, en coordenadas locales. Para evitar complejidades, estos sistemas de referencia deberán escogerse con cuidado, para que no sea demasiado difícil situar el punto C. Por último, las matrices **A** son las matrices de rotación de cada sólido expuestas en el apartado anterior.

2.2.2.2 Restricciones debidas a un par prismático.

Un par prismático consiste en la unión de dos sólidos en los que sólo se permite el deslizamiento relativo entre ellos a través de lo que se llama línea de deslizamiento. Para mayor claridad, obsérvese la imagen siguiente:

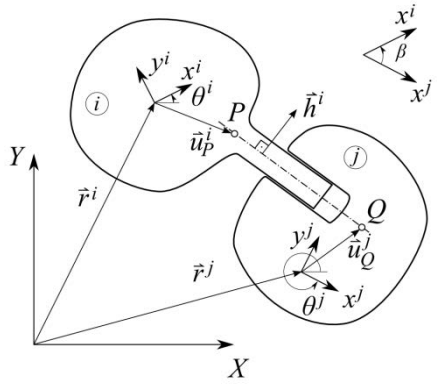


Imagen 2.7: Representación general de un par prismático en coordenadas de referencia. Imagen extraída de [3].

La línea de deslizamiento es aquella que une los puntos P (perteneciente al sólido i) y Q (perteneciente al sólido j). El vector **h** es un vector unitario y perpendicular a dicha línea, y el resto de parámetros y coordenadas representan la analogía del par de rotación en el par presente.

En este tipo de par hay que imponer dos condiciones:

1) Que la orientación relativa de los sólidos no varía.
2) El vector PQ permanece en todo instante perpendicular al vector **h**.

Matemáticamente, esto sería:

$$\theta_i - \theta_j - \beta = 0 \tag{8}$$

$$\left(\mathbf{A^i h^{i'}}\right)^{\mathrm{T}} \left(\mathbf{r^i} + \mathbf{A^i u_P^{i\,\prime}} - \mathbf{r^j} + \mathbf{A^j u_Q^{j\,\prime}}\right) = 0 \tag{9}$$

Como en el par anterior, los vectores **h** y **u** están en coordenadas locales y el **r** en coordenadas globales. De nuevo tenemos las dos ecuaciones de restricción necesarias para definir los dos movimientos que restringe dicho par.

Una vez se tienen definidas las ecuaciones de restricción se puede proceder a la resolución del problema de posición. Para la resolución de dicho problema numéricamente se recurre a un algoritmo Newton-Raphson avanzado que está implementado en la función “fsolve” de Matlab. La explicación detallada de este algoritmo, así como el procedimiento de resolución del problema no lineal que se plantea son explicados en detalle en libros de Dinámica de Sistemas Multicuerpos. El presente proyecto se centra más en el aspecto práctico de dichos algoritmos.

2.2.2.3 Resolución del problema de velocidad y aceleración.

Para resolver el problema de velocidad y aceleración sólo hay que resolver las derivadas de las ecuaciones de restricción con respecto al tiempo una y dos veces, respectivamente. Se tiene que:

$$\frac{\mathrm{dC}(\mathbf{q},\mathrm{t})}{\mathrm{dt}} = \dot{\mathbf{C}}(\mathbf{q},\mathrm{t}) = 0 \tag{10}$$

$$\frac{\mathrm{d}\dot{\mathbf{C}}(\mathbf{q},\mathrm{t})}{\mathrm{dt}} = \ddot{\mathbf{C}}(\mathbf{q},\mathrm{t}) = 0 \tag{11}$$

Se recurre a la regla de la cadena para calcular ambos términos, que quedan:

$$\dot{\mathbf{C}}(\mathbf{q},\mathrm{t}) = \frac{\partial \mathbf{C}(\mathbf{q},\mathrm{t})}{\partial \mathrm{t}}\frac{\mathrm{d}\mathbf{q}}{\mathrm{dt}} + \frac{\partial \mathbf{C}(\mathbf{q},\mathrm{t})}{\partial \mathrm{t}} = \mathbf{C_q}\dot{\mathbf{q}} + \mathbf{C_t} \tag{12}$$

$$\ddot{\mathbf{C}}(\mathbf{q},\mathrm{t}) = \mathbf{C_q}\ddot{\mathbf{q}} + \dot{\mathbf{C}}_\mathbf{q}\dot{\mathbf{q}} + \dot{\mathbf{C}}_\mathbf{t} \tag{13}$$

De manera que la velocidad y la aceleración se resuelven como:

$$\dot{\mathbf{q}} = -\mathbf{C_q^{-1}}\mathbf{C_t} \tag{14}$$

$$\ddot{\mathbf{q}} = -\mathbf{C_q^{-1}}(\dot{\mathbf{C}}_\mathbf{q}\dot{\mathbf{q}} - \dot{\mathbf{C}}_\mathbf{t}) \quad (15)$$

Donde $\dot{\mathbf{C}}_\mathbf{q}$ y $\dot{\mathbf{C}}_\mathbf{t}$ son las derivadas con respecto al tiempo. Como las ecuaciones de restricción geométricas no dependen implícitamente del tiempo (sistemas esclerónomos), se recurre a lo siguiente:

$$\dot{\mathbf{C}}_\mathbf{q} = \frac{\partial \mathrm{Cq}}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \mathbf{t}} = \frac{\partial \mathrm{Cq}}{\partial \mathbf{q}}\dot{\mathbf{q}} = \frac{\partial \mathrm{Cq}}{\partial \mathbf{q}}\mathbf{v} \quad (16)$$

A la hora de resolver numéricamente las aceleraciones, la función DtJacobiano.m llevará en sí la matriz que se obtiene a la hora de derivar el jacobiano de nuevo respecto a sus coordenadas, y la aceleración será:

```
a(:,i) = -Cq\(DCq*v(:,i).^2+DCt)
```

Con DCq y DCt las derivadas del jacobiano y las restricciones respecto al tiempo, respectivamente. En capítulos posteriores de detallará el código completo.

2.2.2.4 Jacobiano de las ecuaciones de restricción debidas a un par de revolución y a un par prismático.

Para resolver el problema de velocidad necesitamos el jacobiano, que consiste en una matriz cuadrada con la misma dimensión que la suma de ecuaciones de restricción y movilidad del mecanismo. Para mayor claridad y como ejemplo: para el mecanismo tratado en este apartado se necesitan 11 ecuaciones de restricción más una ecuación de movilidad para conseguir un sistema compatible, de 12 ecuaciones con 12 incógnitas. Por tanto, se necesitan en principio tantas ecuaciones de movilidad como grados de libertad tenga el mecanismo (esto no siempre es así, pero para dar una vista general puede ser de utilidad entenderlo de esta manera).

Puede parecer una contradicción que en capítulos anteriores se halla explicado la necesidad de 9 ecuaciones de restricción, mientras que ahora se expresa la necesidad de 12. Esto se debe a que, a la hora de programar dicho mecanismo, para evitar un tratamiento diferente de los pares que tengan ligados la barra fija, se opta por añadir 3 ecuaciones de restricción más correspondientes a la barra fija, en las cuales se impone que sus tres coordenadas sean siempre idénticamente nulas.

La matriz jacobiana está compuesta entonces, en cada fila, por las derivadas parciales de cada ecuación de restricción y movilidad con respecto a las diferentes coordenadas del problema, debiéndose poner correctamente cada elemento en la matriz como se especifica en el libro de Dinámica de Sistemas Multicuerpos de José Luis Escalona. Esto es, cada columna corresponde a la coordenada con respecto a la cual deriva, siendo por ejemplo las tres primeras columnas, las coordenadas cartesianas y el ángulo de la barra fija.

Las submatrices jacobianas debidas al par de revolución son, para el sólido i:

$$\begin{bmatrix} 1 & 0 & -\mathrm{u}_{\mathrm{C_x}}^{\mathrm{i}}\mathrm{sen}(\theta_\mathrm{i}) - \mathrm{u}_{\mathrm{C_y}}^{\mathrm{i}}\cos(\theta_\mathrm{i}) \\ 0 & 1 & \mathrm{u}_{\mathrm{C_x}}^{\mathrm{i}}\cos(\theta_\mathrm{i}) - \mathrm{u}_{\mathrm{C_y}}^{\mathrm{i}}\mathrm{sen}(\theta_\mathrm{i}) \end{bmatrix}$$

Y para el sólido j:

$$\begin{bmatrix} -1 & 0 & \mathrm{u}_{\mathrm{C_x}}^{\mathrm{j}}\mathrm{sen}(\theta_{\mathrm{j}}) + \mathrm{u}_{\mathrm{C_y}}^{\mathrm{j}}\cos(\theta_{\mathrm{j}}) \\ 0 & -1 & -\mathrm{u}_{\mathrm{C_x}}^{j}\cos(\theta_{\mathrm{j}}) + \mathrm{u}_{\mathrm{C_y}}^{\mathrm{j}}\mathrm{sen}(\theta_{\mathrm{j}}) \end{bmatrix}$$

Para el par prismático estas submatrices resultan algo más laboriosas, pero se calculan de la misma manera que en el caso del par de revolución. Para el sólido i, se tiene:

$$\begin{bmatrix} \mathbf{0} & 1 \\ (\mathbf{A^i h^i})^{\mathrm{T}} & \boldsymbol{C_{q,1}} \end{bmatrix}$$

Y para el sólido j:

$$\begin{bmatrix} \mathbf{0} & -1 \\ -(\mathbf{A^i h^i})^{\mathrm{T}} & \boldsymbol{C_{q,2}} \end{bmatrix}$$

Con:

$$\mathbf{C_{q,1}} = (\mathbf{A_\theta^i h^i})^{\mathbf{T}}\left(\mathbf{r^i} + \mathbf{A^i u_P^i} - \mathbf{r^j} - \mathbf{A^j u_Q^j}\right) + (\mathbf{A^i h^i})^{\mathbf{T}}(\mathbf{A_\theta^i u_P^i}); \quad (17)$$

$$\mathbf{C_{q,2}} = (\mathbf{A^i h^i})^{\mathbf{T}}(-\mathbf{A_\theta^j u_Q^j}); \quad (18)$$

La implementación numérica de dichas submatrices se realiza en el código de matlab Jacobiano.m, que se puede ver al final de este documento. El procedimiento, explicado de manera general, consiste en ir añadiendo estas submatrices en el mismo orden en el que se añadieron las restricciones en el script Restricciones.m. Empezando por la barra fija, donde tendremos una matriz identidad, seguido por los pares cinemáticos en el mismo orden y, por último, el jacobiano de las ecuaciones de movilidad. De esta forma se consigue una matriz cuadrada que podrá posteriormente ser usada para resolver el problema de velocidad. Para este mismo problema, también será necesario calcular las derivadas con respecto al tiempo de las ecuaciones de restricción y movilidad, lo cual queda reflejado en el programa dtRestr.m. En el ejemplo tratado, se tienen ecuaciones esclerónomas (no tienen el tiempo de forma explícita) excepto en las ecuaciones de movilidad, por lo que hallar estas derivadas resulta sencillo.

Tras la resolución del problema de velocidad, se quieren resolver las aceleraciones, para lo cual se necesitará la derivada de la matriz jacobiana con respecto al tiempo y la derivada segunda respecto al tiempo de la matriz columna de restricciones. Por el mismo motivo que antes, estas dos matrices resultan muy sencillas de calcular, no sólo en un mecanismo sencillo 2D como el tratado en este capítulo, si no en cualquier mecanismo cuyas ecuaciones de restricción no tengan el tiempo de manera explícita.

2.2.2.5 Matriz derivada del jacobiano con respecto a sus coordenadas.

Como se explicó en dos capítulos anteriores, la resolución de la aceleración pasa por crear una matriz que defina las derivadas de cada componente con respecto a la coordenada correspondiente. En el caso que ocupa en la cinemática 2D de este mecanismo tenemos, que para el jacobiano del par de revolución dicha matriz queda, para el sólido i:

$$\begin{bmatrix} 0 & 0 & -\mathrm{u}_{\mathrm{C_x}}^{\mathrm{i}}\cos(\theta_{\mathrm{i}}) + \mathrm{u}_{\mathrm{C_y}}^{\mathrm{i}}\mathrm{sen}(\theta_{\mathrm{i}}) \\ 0 & 0 & -\mathrm{u}_{\mathrm{C_x}}^{\mathrm{i}}\,\mathrm{sen}(\theta_{\mathrm{i}}) - \mathrm{u}_{\mathrm{C_y}}^{\mathrm{i}}\cos(\theta_{\mathrm{i}}) \end{bmatrix}$$

Y para el sólido j:

$$\begin{bmatrix} 0 & 0 & \mathrm{u}_{\mathrm{C_x}}^{\mathrm{j}}\cos(\theta_{\mathrm{j}}) - \mathrm{u}_{\mathrm{C_y}}^{\mathrm{j}}\mathrm{sen}(\theta_{\mathrm{j}}) \\ 0 & 0 & \mathrm{u}_{\mathrm{C_x}}^{j}\,\mathrm{sen}(\theta_{\mathrm{j}}) + \mathrm{u}_{\mathrm{C_y}}^{\mathrm{j}}\cos(\theta_{\mathrm{j}}) \end{bmatrix}$$

En cuanto al par prismático, se tiene, para el sólido i:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \mathbf{C_{q,3}} \end{bmatrix}$$

Y para el sólido j:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \mathbf{C_{q,4}} \end{bmatrix}$$

Con:

$$\mathbf{C_{q,3}} = \left(\mathbf{A_{\theta_2}^i h^i}\right)^{\mathbf{T}}\left(\mathbf{r^i} + \mathbf{A^i u_P^i} - \mathbf{r^j} - \mathbf{A^j u_Q^j}\right) + 2(\mathbf{A_\theta^i h^i})(\mathbf{A_\theta^i u_P^i}) + (\mathbf{A^i h^i})^{\mathbf{T}}(\mathbf{A_{\theta_2}^i u_P^i}) \quad (19)$$

$$\mathbf{C_{q,4}} = (\mathbf{A^i h^i})^{\mathbf{T}}(-\mathbf{A_{\theta_2}^j u_Q^j}); \quad (20)$$

Con $\mathbf{A_{\theta_2}^j}$ y $\mathbf{A_{\theta_2}^i}$ las derivadas con respecto a θ_{i} y θ_{j} de las matrices $\mathbf{A_\theta^i}$ y $\mathbf{A_\theta^j}$.

Por último, cabe definir cómo serán las matrices de las derivadas primera y segunda con respecto al tiempo de las restricciones, pero puesto que estas se pueden calcular de manera sencilla y muchos de sus términos son nulos, se opta por expresar el código en capítulos posteriores.

2.3. Cinemática 3D. Simulación cinemática de una plataforma de Stewart.

Una vez estudiada y entendida la cinemática 2D a la perfección, la programación en 3D es en muchos casos análoga, exceptuando algunos matices nuevos que son necesarios añadir. Para empezar, mientras en 2D se necesitaba un ángulo y dos coordenadas cartesianas para definir el sólido, en el espacio serán necesarios al menos, tres ángulos y tres coordenadas cartesianas. Estos ángulos son llamados coordenadas de orientación.

Existe una gran variedad de tipos de coordenadas de orientación que se pueden elegir a la hora de definir un sólido 3D: ángulos Cardan, ángulos de Euler y parámetros de Euler entre otras. Este proyecto se centrará exclusivamente en los parámetros de Euler, ya que han sido los escogidos a la hora de modelar el sólido debido a que son los más útiles para evitar singularidades cuando se trata el análisis computacional.

2.3.1 Parámetros de Euler.

Los parámetros de Euler son cuatro coordenadas de orientación que se definen como:

$$\theta_0 = \cos\frac{\alpha}{2};\ \theta_1 = \mathrm{v_x}\mathrm{sen}\frac{\alpha}{2};\ \theta_2 = \mathrm{v_y}\mathrm{sen}\frac{\alpha}{2};\ \theta_3 = \mathrm{v_z}\mathrm{sen}\frac{\alpha}{2}$$

El ángulo α es el ángulo que se encuentra girado el sistema de referencia local con respecto al global

y el vector **v** es el vector unitario que indica la dirección del eje de giro. La matriz de rotación de estos parámetros es:

$$\mathbf{A} = \mathbf{I} + 2\tilde{\boldsymbol{\theta}}(\mathbf{I}\theta_0 + \tilde{\boldsymbol{\theta}}) \quad (21)$$

Con $\tilde{\boldsymbol{\theta}}$:

$$\tilde{\boldsymbol{\theta}} = \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix}$$

Los parámetros de Euler añaden lo que se llaman restricciones cinemáticas de orientación, que para este caso concreto no es más que imponer que este vector tiene módulo unidad:

$$\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} - 1 = 0 \quad (22)$$

Cada sólido tendrá asociado, además de sus tres coordenadas cartesianas, sus 4 parámetros de Euler. En principio puede parecer que tendremos tantas restricciones de orientación como sólidos tenga el mecanismo, pero esto no será así, ya que numéricamente se le da un trato especial a la barra fija, como se detallará en capítulos posteriores. Una vez definidos los parámetros de Euler, se sigue con la explicación de las restricciones debidas a pares Cardan y cilíndricos, que son los dos pares que tendrá el mecanismo de este proyecto.

2.3.2 Ecuaciones de restricción debidas a un par Cardan.

En la siguiente figura se muestra un par Cardan, también llamado junta universal. Este par permite dos giros, restringiendo los otros 4 grados de libertad. Para establecer las ecuaciones de restricción, obsérvese la siguiente figura, extraída del libro “Computer-Aided Analysis of Mechanical Systems”, de Parviz E. Nikravesh:

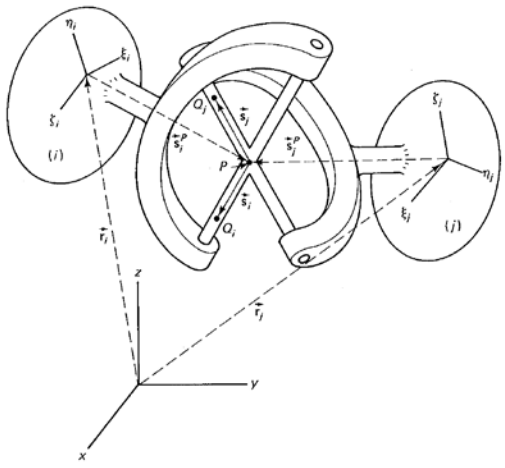


Imagen 2.8: Representación general de un par Cardan o junta universal. Imagen extraída de [2].

Como se puede observar, se tienen que definir dos vectores $\mathbf{s_i}$ y $\mathbf{s_j}$ perpendiculares entre sí, cada uno en la dirección de una de las crucetas del par, así como dos vectores $\mathbf{u_p^i}$ y $\mathbf{u_p^j}$ (en la figura $\mathbf{s_i^P}$ y $\mathbf{s_j^P}$) que van desde el origen de coordenadas hasta el punto P común a ambos sólidos. Debido a que este par permite sólo dos giros, serán necesarias cuatro ecuaciones de restricción, que corresponden a:

1) Imponer que el punto P pertenece permanentemente a ambos sólidos (3 ecuaciones)
2) Imponer que los vectores **s** permanecen perpendiculares entre sí siempre.

Estos dos requerimientos se traducen matemáticamente como:

$$\mathbf{r^i} + \mathbf{A_i u_p^i} = \mathbf{r^j} + \mathbf{A_j u_p^j} \tag{23}$$

$$(\mathbf{A_i} \boldsymbol{s_i})^T (\boldsymbol{A_j s_j}) = \mathbf{0} \tag{24}$$

Siendo **r** los vectores correspondientes a las tres coordenadas cartesianas en globales de cada sólido i y j. Como en cinemática 2D, los vectores **u** quedan vistos desde cada sistema de referencia local de cada sólido, igual que los vectores **s**.

2.3.3 Ecuaciones de restricción debidas a un par cilíndrico.

El par cilíndrico permite una tanslación relativa entre los dos sólidos a través de una línea de deslizamiento, y un giro alrededor de este eje, tal y como se muestra en la figura:

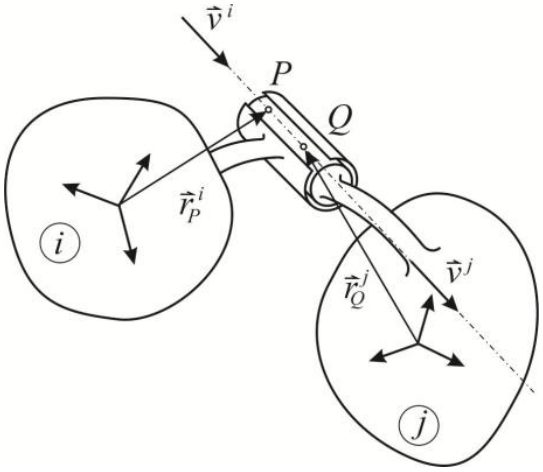


Imagen 2.9: Representación general de un par cilíndrico. Imagen extraída de [3].

Dicho par restringe cuatro grados de libertad, por tanto, se necesitan cuatro ecuaciones de restricción.

Se introducen los vectores **v**, que tienen módulo unidad y ambos la misma dirección del eje de deslizamiento, y los vectores **u**, que indican la posición de los puntos P y Q en locales. Dichos puntos pertenecen cada uno a un sólido diferente y deben estar alineados en el eje de giro del par.

Las condiciones que hay que imponer en este caso son:

1) Los vectores **v** son paralelos entre sí.
2) El vector que conecta P y Q también es paralelo a **v**.

Para establecer estas condiciones de paralelismos, se recurre a imponer una doble perpendicularidad. Para ello, se crean dos vectores $\mathbf{v_1^i}$ y $\mathbf{v_2^i}$ de la siguiente manera:

$$\mathbf{v_1^i} = \begin{bmatrix}1\\0\\0\end{bmatrix} \text{x } \mathbf{v^i} \text{ , } \mathbf{v_2^i} = \mathbf{v^i} \text{ x } \mathbf{v_1^i}$$

Si el vector $\mathbf{v^i}$ estuviera en la dirección del eje x se escoge otro vector para evitar que el producto vectorial sea nulo.

Las restricciones se expresan matemáticamente como:

$$(\mathbf{A^i v_1^i})^{\boldsymbol{T}} \mathbf{A^j v^j} = 0 \quad (25)$$

$$(\mathbf{A^i v_2^i})^{\boldsymbol{T}} \mathbf{A^j v^j} = 0 \quad (26)$$

$$(\mathbf{A^i v_1^i})^{\mathbf{T}} (\mathbf{r^i} + \mathbf{A^i u_P^i} - \mathbf{r^j} - \mathbf{A^j u_Q^j}) = 0 \quad (27)$$

$$(\mathbf{A^i v_2^i})^{\mathbf{T}} \left(\mathbf{r^i} + \mathbf{A^i u_P^i} - \mathbf{r^j} - \mathbf{A^j u_Q^j}\right) = 0 \quad (28)$$

Se tratan de cuatro ecuaciones algebraicas de restricción.

2.3.4 Jacobianos del par Cardan y cilíndricos. Jacobiano de las restricciones de orientación.

Hallar los jacobianos de las ecuaciones de restricción de ambos pares en mecanismos 3D es completamente análogo al 2D, pero más laborioso debido al aumento de coordenadas. A continuación, se expresan las submatrices de cada sólido, tal y como se expusieron en el 2D. Estas submatrices deberán incorporarse correspondientemente en el jacobiano, en la fila correspondiente a la ecuación de restricción que estemos tratando y en la columna correspondiente a la variable con respecto a la cual estamos derivando.

La submatriz del jacobiano debido a un par Cardan para el sólido i es:

$$\begin{bmatrix} 1 & 0 & 0 & & & & \\ 0 & 1 & 0 & \mathbf{A^i_{\theta_0} u_P^i} & \mathbf{A^i_{\theta_1} u_P^i} & \mathbf{A^i_{\theta_2} u_P^i} & \mathbf{A^i_{\theta_3} u_P^i} \\ 0 & 0 & 1 & & & & \\ 0 & 0 & 0 & \left(\mathbf{A^i_{\theta_0} s_i}\right)^{\mathbf{T}} (\mathbf{A_j s_i}) & \left(\mathbf{A^i_{\theta_1} s_i}\right)^{\mathbf{T}} (\mathbf{A_j s_i}) & \left(\mathbf{A^i_{\theta_2} s_i}\right)^{\mathbf{T}} (\mathbf{A_j s_i}) & \left(\mathbf{A^i_{\theta_3} s_i}\right)^{\mathbf{T}} (\mathbf{A_j s_i}) \end{bmatrix}$$

Y para el sólido j:

$$\begin{bmatrix} -1 & 0 & 0 & & & & \\ 0 & -1 & 0 & -\mathbf{A^j_{\theta_0} u_P^j} & -\mathbf{A^j_{\theta_1} u_P^j} & -\mathbf{A^j_{\theta_2} u_P^j} & -\mathbf{A^j_{\theta_3} u_P^j} \\ 0 & 0 & -1 & & & & \\ 0 & 0 & 0 & (\mathbf{A_i s_i})^{\mathbf{T}} \left(\mathbf{A^j_{\theta_0} s_i}\right) & (\mathbf{A_i s_i})^{\mathbf{T}} \left(\mathbf{A^j_{\theta_1} s_i}\right) & (\mathbf{A_i s_i})^{\mathbf{T}} \left(\mathbf{A^j_{\theta_2} s_i}\right) & (\mathbf{A_i s_i})^{\mathbf{T}} \left(\mathbf{A^j_{\theta_3} s_i}\right) \end{bmatrix}$$

Cada submatriz es de 4x7, correspondientes a las cuatro ecuaciones de restricción y a las siete coordenadas de cada sólido. Las matrices $\boldsymbol{A_\theta}$ corresponden a las matrices cuyos componentes son

las derivadas parciales con respecto al parámetro de Euler que corresponda.

La implementación de la matriz jacobiana debida al par cilíndrico resulta algo más laboriosa, debido a que algunas derivadas parciales requieren aplicar la regla de la cadena, lo que hace que dicha expresión se alargue. Para el sólido i se tiene:

$$\begin{bmatrix} 0 & 0 & 0 & \left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_0}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\mathbf{A}_{\mathbf{j}}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_1}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\mathbf{A}_{\mathbf{j}}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_2}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\mathbf{A}_{\mathbf{j}}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_3}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\mathbf{A}_{\mathbf{j}}\mathbf{v}_{\mathbf{j}} \\ 0 & 0 & 0 & \left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_0}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\mathbf{A}_{\mathbf{j}}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_1}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\mathbf{A}_{\mathbf{j}}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_2}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\mathbf{A}_{\mathbf{j}}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_3}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\mathbf{A}_{\mathbf{j}}\mathbf{v}_{\mathbf{j}} \\ \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}} & & \mathbf{C}^{\boldsymbol{\theta}_0}_{\mathbf{q},\mathbf{1}} & \mathbf{C}^{\boldsymbol{\theta}_1}_{\mathbf{q},\mathbf{1}} & \mathbf{C}^{\boldsymbol{\theta}_2}_{\mathbf{q},\mathbf{1}} & \mathbf{C}^{\boldsymbol{\theta}_3}_{\mathbf{q},\mathbf{1}} \\ \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}} & & \mathbf{C}^{\boldsymbol{\theta}_0}_{\mathbf{q},\mathbf{2}} & \mathbf{C}^{\boldsymbol{\theta}_1}_{\mathbf{q},\mathbf{2}} & \mathbf{C}^{\boldsymbol{\theta}_2}_{\mathbf{q},\mathbf{2}} & \mathbf{C}^{\boldsymbol{\theta}_3}_{\mathbf{q},\mathbf{2}} \end{bmatrix}$$

Donde:

$$\mathbf{C}^{\boldsymbol{\theta}_k}_{\mathbf{q},\mathbf{1}} = \left(\mathbf{A}_{\boldsymbol{\theta}_k}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\left(\mathbf{r}^{\mathbf{i}} + \mathbf{A}_{\mathbf{i}}\mathbf{u}^{\mathbf{i}}_{\mathbf{P}} - \mathbf{r}^{\mathbf{j}} - \mathbf{A}_{\mathbf{j}}\mathbf{u}^{\mathbf{j}}_{\mathbf{Q}}\right) + \left(\mathbf{A}^{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_k}\mathbf{u}^{\mathbf{i}}_{\mathbf{P}}\right) \tag{29}$$

$$\mathbf{C}^{\boldsymbol{\theta}_k}_{\mathbf{q},\mathbf{2}} = \left(\mathbf{A}_{\boldsymbol{\theta}_k}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\left(\mathbf{r}^{\mathbf{i}} + \mathbf{A}_{\mathbf{i}}\mathbf{u}^{\mathbf{i}}_{\mathbf{P}} - \mathbf{r}^{\mathbf{j}} - \mathbf{A}_{\mathbf{j}}\mathbf{u}^{\mathbf{j}}_{\mathbf{Q}}\right) + \left(\mathbf{A}^{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\left(\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\theta}_k}\mathbf{u}^{\mathbf{i}}_{\mathbf{P}}\right) \tag{30}$$

Y, por último, para el sólido j:

$$\begin{bmatrix} 0 & 0 & 0 & \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_0}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_1}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_2}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_3}\mathbf{v}_{\mathbf{j}} \\ 0 & 0 & 0 & \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_0}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_1}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_2}\mathbf{v}_{\mathbf{j}} & \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_3}\mathbf{v}_{\mathbf{j}} \\ \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}} & & \mathbf{C}^{\boldsymbol{\theta}_0}_{\mathbf{q},\mathbf{1}} & \mathbf{C}^{\boldsymbol{\theta}_1}_{\mathbf{q},\mathbf{1}} & \mathbf{C}^{\boldsymbol{\theta}_2}_{\mathbf{q},\mathbf{1}} & \mathbf{C}^{\boldsymbol{\theta}_3}_{\mathbf{q},\mathbf{1}} \\ \left(\mathbf{A}_{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}} & & \mathbf{C}^{\boldsymbol{\theta}_0}_{\mathbf{q},\mathbf{2}} & \mathbf{C}^{\boldsymbol{\theta}_1}_{\mathbf{q},\mathbf{2}} & \mathbf{C}^{\boldsymbol{\theta}_2}_{\mathbf{q},\mathbf{2}} & \mathbf{C}^{\boldsymbol{\theta}_3}_{\mathbf{q},\mathbf{2}} \end{bmatrix}$$

Donde:

$$\mathbf{C}^{\boldsymbol{\theta}_k}_{\mathbf{q},\mathbf{1}} = -\left(\mathbf{A}^{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{1}}\right)^{\mathrm{T}}\left(\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_k}\mathbf{u}^{\mathbf{j}}_{\mathbf{Q}}\right) \tag{31}$$

$$\mathbf{C}^{\boldsymbol{\theta}_k}_{\mathbf{q},\mathbf{2}} = -\left(\mathbf{A}^{\mathbf{i}}\mathbf{v}^{\mathbf{i}}_{\mathbf{2}}\right)^{\mathrm{T}}\left(\mathbf{A}^{\mathbf{j}}_{\boldsymbol{\theta}_k}\mathbf{u}^{\mathbf{j}}_{\mathbf{Q}}\right) \tag{32}$$

Ya se tienen, por tanto, los jacobianos debidos a las ecuaciones de restricción cinemáticas. Faltarían por añadir la parte del jacobiano correspondiente a las restricciones de orientación y las restricciones de movilidad que el usuario impone según el movimiento que quiera simular en el mecanismo, por lo que estas últimas pueden tener cualquier forma. En el caso del presente proyecto, y debido al uso de los parámetros de Euler, las restricciones de orientación que se tienen son todas de la forma $\boldsymbol{\theta}^{\mathbf{T}}\boldsymbol{\theta} = 1$. La fila correspondiente a dicha restricción tendrá la siguiente forma:

$$\begin{bmatrix} 0 & 0 & 0 & 2\theta_0 & 2\theta_1 & 2\theta_2 & 2\theta_3 \end{bmatrix}$$

2.3.5 Matriz de la derivada del jacobiano $\mathbf{DC_q}$.

Para poder resolver el problema de aceleración se procede igual que en el caso del 2D. La matriz de la derivada del jacobiano debido al par Cardan, para el sólido i, es:

$$\begin{bmatrix} 0 & 0 & 0 & & & & \\ 0 & 0 & 0 & \mathbf{A^i_{\theta_{oo}} u^i_P} & \mathbf{A^i_{\theta_{11}} u^i_P} & \mathbf{A^i_{\theta_{22}} u^i_P} & \mathbf{A^i_{\theta_{33}} u^i_P} \\ 0 & 0 & 0 & & & & \\ 0 & 0 & 0 & \mathbf{\left(A^i_{\theta_{oo}} s_i\right)^T (A_j s_i)} & \mathbf{\left(A^i_{\theta_{11}} s_i\right)^T (A_j s_i)} & \mathbf{\left(A^i_{\theta_{22}} s_i\right)^T (A_j s_i)} & \mathbf{\left(A^i_{\theta_{33}} s_i\right)^T (A_j s_i)} \end{bmatrix}$$

Donde $\mathbf{A_{\theta_{kk}}}$ es la derivada con respecto al parámetro de Euler θ_k de la matriz $\mathbf{A_{\theta_k}}$

Y para el sólido j:

$$\begin{bmatrix} 0 & 0 & 0 & & & & \\ 0 & 0 & 0 & \mathbf{-A^j_{\theta_{oo}} u^j_P} & \mathbf{-A^j_{\theta_{11}} u^j_P} & \mathbf{-A^j_{\theta_{22}} u^j_P} & \mathbf{-A^j_{\theta_{33}} u^j_P} \\ 0 & 0 & 0 & & & & \\ 0 & 0 & 0 & \mathbf{(A_i s_i)^T \left(A^j_{\theta_{oo}} s_i\right)} & \mathbf{(A_i s_i)^T \left(A^j_{\theta_{11}} s_i\right)} & \mathbf{(A_i s_i)^T \left(A^j_{\theta_{22}} s_i\right)} & \mathbf{(A_i s_i)^T \left(A^j_{\theta_{33}} s_i\right)} \end{bmatrix}$$

Con respecto al par prismático, se tiene, para el sólido i:

$$\begin{bmatrix} 0 & 0 & 0 & \mathbf{\left(A^i_{\theta_{oo}} v^i_1\right)^T A_j v_j} & \mathbf{\left(A^i_{\theta_{11}} v^i_1\right)^T A_j v_j} & \mathbf{\left(A^i_{\theta_{22}} v^i_1\right)^T A_j v_j} & \mathbf{\left(A^i_{\theta_{33}} v^i_1\right)^T A_j v_j} \\ 0 & 0 & 0 & \mathbf{\left(A^i_{\theta_{oo}} v^i_2\right)^T A_j v_j} & \mathbf{\left(A^i_{\theta_{11}} v^i_2\right)^T A_j v_j} & \mathbf{\left(A^i_{\theta_{22}} v^i_2\right)^T A_j v_j} & \mathbf{\left(A^i_{\theta_{33}} v^i_2\right)^T A_j v_j} \\ 0 & 0 & 0 & \mathbf{C^{\theta_{oo}}_{q,1}} & \mathbf{C^{\theta_{11}}_{q,1}} & \mathbf{C^{\theta_{22}}_{q,1}} & \mathbf{C^{\theta_{33}}_{q,1}} \\ 0 & 0 & 0 & \mathbf{C^{\theta_{oo}}_{q,2}} & \mathbf{C^{\theta_{11}}_{q,2}} & \mathbf{C^{\theta_{22}}_{q,2}} & \mathbf{C^{\theta_{33}}_{q,2}} \end{bmatrix}$$

Donde:

$$\mathbf{C^{\theta_{kk}}_{q,1}} = (\mathbf{A_{\theta_{kk}} v^i_1})^T (\mathbf{r^i + A_i u^i_P - r^j - A_j u^j_Q}) + 2(\mathbf{A_{\theta_k} v^i_1})^T (\mathbf{A_{\theta_k} u^i_P}) + (\mathbf{A^i v^i_1})^T (\mathbf{A^i_{\theta_{kk}} u^i_P}) \quad (33)$$

$$\mathbf{C^{\theta_{kk}}_{q,2}} = (\mathbf{A_{\theta_{kk}} v^i_2})^T (\mathbf{r^i + A_i u^i_P - r^j - A_j u^j_Q}) + 2(\mathbf{A_{\theta_k} v^i_2})^T (\mathbf{A_{\theta_k} u^i_P}) + (\mathbf{A^i v^i_2})^T (\mathbf{A^i_{\theta_{kk}} u^i_P}) \quad (34)$$

Por último, para el sólido j:

$$\begin{bmatrix} 0 & 0 & 0 & \mathbf{(A_i v^i_1)^T A^j_{\theta_{oo}} v_j} & \mathbf{(A_i v^i_1)^T A^j_{\theta_{11}} v_j} & \mathbf{(A_i v^i_1)^T A^j_{\theta_{22}} v_j} & \mathbf{(A_i v^i_1)^T A^j_{\theta_{33}} v_j} \\ 0 & 0 & 0 & \mathbf{(A_i v^i_2)^T A^j_{\theta_{oo}} v_j} & \mathbf{(A_i v^i_2)^T A^j_{\theta_{11}} v_j} & \mathbf{(A_i v^i_2)^T A^j_{\theta_{22}} v_j} & \mathbf{(A_i v^i_2)^T A^j_{\theta_{33}} v_j} \\ 0 & 0 & 0 & \mathbf{C^{\theta_o}_{q,1}} & \mathbf{C^{\theta_1}_{q,1}} & \mathbf{C^{\theta_2}_{q,1}} & \mathbf{C^{\theta_3}_{q,1}} \\ 0 & 0 & 0 & \mathbf{C^{\theta_o}_{q,2}} & \mathbf{C^{\theta_1}_{q,2}} & \mathbf{C^{\theta_2}_{q,2}} & \mathbf{C^{\theta_3}_{q,2}} \end{bmatrix}$$

Con:

$$\mathbf{C^{\theta_k}_{q,1}} = -(\mathbf{A^i v^i_1})^T (\mathbf{A^j_{\theta_{kk}} u^j_Q}) \quad (35)$$

$$\mathbf{C^{\theta_k}_{q,2}} = -(\mathbf{A^i v^i_2})^T (\mathbf{A^j_{\theta_{kk}} u^j_Q}) \quad (36)$$

2.3.6 Creación de la estructura de la plataforma de Stewart.

El primer paso de la modelización de la plataforma es establecer su configuración geométrica, así como los sistemas de referencias global y locales de cada sólido. Por simplicidad, el sistema de referencia global se sitúa en el centro de la plataforma fija, con el plano x-y perteneciente a dicha plataforma y el eje z perpendicular y apuntando a la plataforma móvil.

Por motivos de comodidad a la hora de crear el fichero de estructura los sistemas de referencias locales de cada una de las 12 barras que unen las dos plataformas se han situado en los extremos de las barras, pegados a las plataformas, de manera que el eje z pertenece siempre al eje de las barras y el plano x-y permanece perpendicular a dicho eje. Por último, el sistema de coordenadas local de la plataforma móvil se sitúa en el centro de esta y con su plano x-y paralelo al plano de la plataforma fija, pero girado con respecto a estos 60 grados.

De esta forma, cuando se modelen los pares cilíndricos, los vectores $\mathbf{v^i}, \mathbf{v^j}$ tendrán la dirección y sentido del eje z. Y los vectores $\mathbf{u_P^i}$ y $\mathbf{u_Q^j}$ serán nulos debido a que los puntos P y Q se sitúan en los orígenes del sistema.

Por último, los pares Cardan se sitúan en ambas plataformas de manera que la parte de la cruceta perteneciente a dicha plataforma sea perpendicular a la dirección que une el centro de la plataforma con el par, quedando así definidos los vectores $\mathbf{s_i}, \mathbf{s_j}, \mathbf{u_P^i}$ y $\mathbf{u_P^j}$ tal y como se aprecia en el programa de Matlab StewartPlatform.m, expuesto al final de este documento, con anotaciones que tienen como objetivo facilitar su comprensión.

Cabe destacar que, con el objetivo de realizar luego un estudio dinámico del mecanismo, estos sistemas de referencias de las barras se trasladan a su centro de gravedad de forma que sus ejes sean ejes principales de inercia. Esto resulta útil a la hora de montar la matriz de masa para que esta sea diagonal y como se explica en el apartado de dinámica inversa.

2.4. Dinámica Inversa 2D

Tras el estudio de la cinemática de los mecanismos 2D y 3D, se procede al estudio de la dinámica inversa, donde también se trabajará primero en 2D, y posteriormente se explicarán los nuevos detalles que atañen al caso 3D.

Para empezar la dinámica inversa consiste en, conocido el movimiento de un mecanismo, averiguar las fuerzas que surgen en el para producir dichos movimientos. La dinámica directa, por el contrario, consiste en dadas unas fuerzas, calcular cómo será el movimiento del mecanismo.

Para realizar el estudio dinámico del mecanismo lo primero que se ha tenido en cuenta es la posición de los sistemas de referencia. Estos podrían estar donde mismo que estaban a la hora de la resolución del problema cinemático, sin embargo, para facilitar el cálculo se procede a desplazar dichos sistemas de referencias a los centros de inercia de las barras, de esta manera el cálculo se simplificará considerablemente, como se verá a continuación.

Las ecuaciones del movimiento, para un sólido no sometido a restricciones, se expresa de la siguiente manera:

$$m_i \ddot{x}_i = f_x \tag{37}$$

$$m_i \ddot{y}_i = f_y \tag{38}$$

$$\mu_i \ddot{\phi}_i = n_i \tag{39}$$

O lo que es lo mismo, en forma matricial:

$$\mathbf{M_i \ddot{q}_i} = \mathbf{g_i} \tag{40}$$

Donde la matriz $\mathbf{M_i}$ es la matriz de masas que, al tener los sistemas de referencia en ejes principales de inercia, es diagonal con los dos primeros componentes la masa de la barra y el tercer componente la inercia respecto al eje perpendicular al plano. El vector $\ddot{\mathbf{q}}_\mathbf{i}$ está compuesto por las aceleraciones del sólido i en x, y y la aceleración angular. Por último, el vector $\mathbf{g_i}$ es el vector de fuerzas externas que actúan sobre el sólido (como por ejemplo, la gravedad). Quedaría:

$$\begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & \mu_i \end{bmatrix} \begin{bmatrix} \ddot{x}_i \\ \ddot{y}_i \\ \ddot{\phi}_i \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ n_i \end{bmatrix} \tag{41}$$

Para un sólido con restricciones habrá que añadir al sistema anterior otro vector de reacciones debido a esas restricciones. Quedaría, por tanto:

$$\mathbf{M\ddot{q}} = \mathbf{g} + \mathbf{g^{(c)}} \tag{42}$$

Donde $\mathbf{g^{(c)}}$ es el vector de reacciones, el cual resulta ser:

$$\mathbf{g^{(c)}} = \mathbf{\Phi_q^T \lambda} \tag{43}$$

Donde la matriz $\mathbf{\Phi_q^T}$ es la matriz jacobiana traspuesta calculada en el problema cinemático y el vector $\boldsymbol{\lambda}$ consistiría en el vector incógnita a calcular. Despejando dicho vector, el sistema a resolver sería:

$$\boldsymbol{\lambda} = (\mathbf{\Phi_q^T})^{-1}(\mathbf{M\ddot{q}} - \mathbf{g}) \tag{44}$$

Este vector $\boldsymbol{\lambda}$ es llamado vector de reacciones, y cada uno de sus componentes está asociado a las ecuación de restricción correspondiente que se halla establecido en el problema cinemático.

2.5. Dinámica Inversa 3D.

La dinámica inversa en el caso 3D comparte la misma base que el caso 2D, pero resulta más laborioso y se añaden conceptos y cálculos nuevos que hay que realizar debido a la inclusión de los parámetros de Euler.

El primer paso a realizar, como en el caso 2D es el de poner todos los sistemas de coordenadas en ejes principales de inercia para cada sólido, para así conseguir una matriz de masas diagonal.

Lo primero que cambia es que a la hora de plantear las ecuaciones de movimiento hay que distinguir entre ecuaciones de movimiento debido a las traslaciones (a lo largo de los ejes cartesianos) y las ecuaciones del movimiento debido a las rotaciones (alrededor de los ejes cartesianos). Estas últimas están relacionadas con las velocidades angulares de los sólidos alrededor de los ejes cartesianos. Sin embargo, a lo largo de toda la cinemática en este proyecto se ha trabajado con los llamados ángulos de Euler, con lo que será necesario el desarrollo de una formulación específica para pasar de trabajar con los parámetros de Euler a trabajar con velocidades angulares.

Yendo parte por parte, las ecuaciones del movimiento traslacional son, como en el caso 2D:

$$\mathbf{N_i}\ddot{\mathbf{r}}_\mathbf{i} = \mathbf{f_i} \tag{45}$$

Donde la primera matriz es una matriz diagonal (porque se trabaja con ejes de coordenadas en ejes principales de inercia) de masa de los sólidos de dimensión 3 y con la masa del sólido en cada componente de la diagonal. Esta matriz se encuentra multiplicada por el vector de aceleraciones, y se iguala dicho producto a las fuerzas externas.

Para el caso de las ecuaciones de movimiento rotacionales se va a llevar a cabo la siguiente formulación extraída de la literatura, concretamente del libro de Parviz E.Nikravesh “Computer-Aided Analysis of Mechanical Systems”:

$$\mathbf{J}'_\mathbf{i}\dot{\boldsymbol{\omega}}_\mathbf{i}' + \widetilde{\boldsymbol{\omega}}_\mathbf{i}\mathbf{J}_\mathbf{i}'\boldsymbol{\omega}_\mathbf{i}' = \mathbf{n}_\mathbf{i}' \tag{46}$$

Donde $\boldsymbol{\omega}'_i$ $\boldsymbol{y}$ $\dot{\boldsymbol{\omega}}_i'$ son la velocidad y aceleración angular del sólido i, que se calculan a partir de los parámetros de Euler del sólido i de la siguiente forma:

$$\boldsymbol{\omega}'_\mathbf{i} = 2\begin{bmatrix} -\mathrm{e}_1 & \mathrm{e}_0 & \mathrm{e}_3 & -\mathrm{e}_2 \\ -\mathrm{e}_2 & -\mathrm{e}_3 & \mathrm{e}_0 & \mathrm{e}_1 \\ -\mathrm{e}_3 & \mathrm{e}_2 & -\mathrm{e}_1 & \mathrm{e}_0 \end{bmatrix}\begin{bmatrix} \dot{\mathrm{e}}_0 \\ \dot{\mathrm{e}}_1 \\ \dot{\mathrm{e}}_2 \\ \dot{\mathrm{e}}_3 \end{bmatrix} \tag{47}$$

$$\boldsymbol{\omega}'_{\mathbf{i}} = 2\mathbf{L}\dot{\mathbf{p}} \tag{48}$$

Y la derivada se calcula como:

$$\dot{\boldsymbol{\omega}}'_{\mathbf{i}} = 2\mathbf{L}\ddot{\mathbf{p}} \tag{49}$$

Y, por último:

$$\widetilde{\boldsymbol{\omega}}_{\mathbf{i}} = 2\dot{\mathbf{G}}\mathbf{L}^{\mathbf{T}}\mathbf{L}\mathbf{G}^{\mathbf{T}} \tag{50}$$

Donde:

$$\mathbf{G} = \begin{bmatrix} -e_1 & e_0 & -e_3 & e_2 \\ -e_2 & e_3 & e_0 & -e_1 \\ -e_3 & -e_2 & e_1 & e_0 \end{bmatrix} \tag{51}$$

Posteriormente, se expresan las ecuaciones de movimiento para sólidos sometidos a restricciones. Para las ecuaciones traslacionales se tiene, al igual que en el caso 2D:

$$\mathbf{N}_{\mathbf{i}}\ddot{\mathbf{r}}_{\mathbf{i}} = \mathbf{f}_{\mathbf{i}} + \mathbf{f}_{\mathbf{i}}^{(\mathbf{c})} \tag{52}$$

A la cual se les ha añadido las reacciones debidas a dichas restricciones. Dichas reacciones son:

$$\mathbf{f}_{\mathbf{i}}^{(\mathbf{c})} = \boldsymbol{\Phi}_{\boldsymbol{r_i}}^{\boldsymbol{T}}\boldsymbol{\lambda} \tag{53}$$

Por tanto, se tiene:

$$\mathbf{N}_{\mathbf{i}}\ddot{\mathbf{r}}_{\mathbf{i}} - \boldsymbol{\Phi}_{\boldsymbol{r_i}}^{\boldsymbol{T}}\boldsymbol{\lambda} = \mathbf{f}_{\mathbf{i}} \tag{54}$$

Como en el caso bidimensional, la incógnita es $\boldsymbol{\lambda}$.

Para las ecuaciones rotacionales, las ecuaciones de la formulación presentada anteriormente quedan:

$$\mathbf{J}'_{\mathbf{i}}\dot{\boldsymbol{\omega}}_{\mathbf{i}}{}' + \widetilde{\boldsymbol{\omega}}_{\mathbf{i}}\mathbf{J}_{\mathbf{i}}'^{\boldsymbol{\omega}_{\mathbf{i}}'} - \frac{1}{2}\mathbf{L}_{\mathbf{i}}\boldsymbol{\Phi}_{\mathbf{P_i}}^{\mathbf{T}}\boldsymbol{\lambda} = \mathbf{n}_{\mathbf{i}}{}' \tag{55}$$

Para unificar el problema y unificar las ecuaciones de movimiento translacionales y rotacionales habría que resolver el siguiente sistema:

$$\begin{bmatrix} \mathbf{M} & \mathbf{B^T} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{h}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \gamma^{\#} \end{bmatrix} \quad (56)$$

Donde **M** es la matriz de masa, diagonal (por estar los sistemas de referencia en ejes principales de inercia), compuesta por submatrices 6x6 (una por cada sólido del mecanismo) donde los tres primeros elementos de la diagonal son la masa del sólido y los 3 siguientes las inercias con respecto a x, y y z respectivamente.

La matriz **B** es:

$$\mathbf{B} = \left[\boldsymbol{\Phi}_{\mathbf{r_1}}, \quad \frac{1}{2}\boldsymbol{\Phi}_{\boldsymbol{p_1}} \boldsymbol{L_1^T}, \quad \dots, \quad \boldsymbol{\Phi}_{\mathbf{r_b}}, \quad \frac{1}{2}\boldsymbol{\Phi}_{\boldsymbol{p_b}} \boldsymbol{L_b^T}, \right] \quad (57)$$

La matriz $\dot{\mathbf{h}}$ y **b**:

$$\dot{\mathbf{h}} = \begin{bmatrix} \ddot{\boldsymbol{r}}_{\mathbf{1}} \\ \dot{\boldsymbol{\omega}}_{\mathbf{1}}{}' \\ \dots \\ \ddot{\boldsymbol{r}}_{\boldsymbol{b}} \\ \dot{\boldsymbol{\omega}}_{\boldsymbol{b}}{}' \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{0} \\ \widetilde{\boldsymbol{\omega}_{\mathbf{1}}} \mathbf{J}_{\mathbf{1}}{}' \boldsymbol{\omega}_{\mathbf{1}}{}' \\ \dots \\ \mathbf{0} \\ \widetilde{\boldsymbol{\omega}_{\mathbf{b}}} \mathbf{J}_{\mathbf{b}}{}' \boldsymbol{\omega}_{\mathbf{b}}{}' \end{bmatrix}$$

Para entender estos fundamentos en mayor profundidad el lector puede recurrir al libro de Nikravesh, expuesto en la bibliografía. Como se puede observar la matriz **B** no es cuadrada, lo que implica que para poder resolver el sistema se necesita completar dicho sistema con una artificialidad, que en este caso es añadir el vector $\gamma^{\#}$ como se puede ver en la expresión matricial anterior. Al resolver el sistema, se obtendrán los valores de $\dot{\mathbf{h}}$ que ya eran conocidos, pero además el vector $\boldsymbol{\lambda}$, que nos servirá de manera análoga al caso 2D para conocer fuerzas que se producen en el mecanismo cuando imponemos algún movimiento.

Una vez implementado este procedimiento en la plataforma de Stewart aparece una singularidad en el que la matriz compuesta por la matriz de masa **M** y la matriz **B**. Por ello se opta por otro procedimiento, extraído del libro de Edward J. Haug: “Computer Aided Kinematics and Dynamics of Mechanical Systems”. Dicho procedimiento consiste en resolver la siguiente expresión.

$$\mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathbf{T}} \boldsymbol{\lambda} = \mathbf{F}$$

Donde las matrices $\ddot{\mathbf{q}}$, $\boldsymbol{\Phi}_{\boldsymbol{q}}^{\boldsymbol{T}}$ y $\boldsymbol{\lambda}$ no varían respecto al procedimiento anterior. Sin embargo, las matrices **M** y **F** sufren una ligera modificación:

$$\mathbf{M} = \begin{bmatrix} \mathbf{m_i} & \mathbf{0} \\ \mathbf{0} & \mathbf{4 L_i^T J_i L_i} \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} \mathbf{F_i} \\ \mathbf{2 L_i^T n_i + 8 \dot{L}_i^T J_i \dot{L}_i p_i} \end{bmatrix}$$

La matriz **M** es una matriz de dimensión 7n, con n el número de sólidos del mecanismo. Las tres

primeras componentes de cada submatriz diagonal de 7x7 son la masa del sólido correspondiente, mientras que las cuatro siguientes es la operación que se observa más arriba. La matriz **F** es una matriz de 7n filas y una columna. Los tres primeros componentes son las acciones externas a lo largo de los ejes coordenados, mientras que los cuatro siguientes representan la operación que aparece arriba, donde **L** es la matriz ya expuesta con anterioridad, **J** la matriz de inercias y **p**, la matriz de los cuatro parámetros de Euler correspondientes. La solución sería:

$$\boldsymbol{\lambda} = (\boldsymbol{\Phi}_{\mathbf{q}}^{\mathbf{T}})^{-1}(\mathbf{F} - \mathbf{M}\ddot{\mathbf{q}}) \tag{58}$$

2.6. Fuerza en los pistones.

El ultimo paso realizado en el presente Proyecto ha sido el cálculo de las fuerzas necesarias que deben realizar los seis pistones hidráulicos de la plataforma para conseguir representar el movimiento impuesto. Para ello se ha recurrido como base teórica al Teorema de las Potencias Virtuales.

De manera resumida, el problema consiste en solucionar la siguiente expresión:

$$\begin{bmatrix} \mathrm{F}_{\mathrm{x},14} \\ \mathrm{F}_{\mathrm{y},14} \\ \mathrm{F}_{\mathrm{z},14} \\ \mathrm{F}_{\mathrm{e}_0,14} \\ \mathrm{F}_{\mathrm{e}_1,14} \\ \mathrm{F}_{\mathrm{e}_2,14} \\ \mathrm{F}_{\mathrm{e}_3,14} \end{bmatrix} = [\boldsymbol{\gamma}_{\mathbf{2-13}} \quad \boldsymbol{\gamma}_{\mathbf{3-8}} \quad \boldsymbol{\gamma}_{\mathbf{4-9}} \quad \boldsymbol{\gamma}_{\mathbf{5-10}} \quad \boldsymbol{\gamma}_{\mathbf{6-11}} \quad \boldsymbol{\gamma}_{\mathbf{7-12}}] \begin{bmatrix} \mathrm{F}_{2-13} \\ \mathrm{F}_{3-8} \\ \mathrm{F}_{4-9} \\ \mathrm{F}_{5-10} \\ \mathrm{F}_{6-11} \\ \mathrm{F}_{7-12} \end{bmatrix} \tag{59}$$

Escrito más compacto sería:

$$\mathbf{F_{14}} = \boldsymbol{\Gamma}\ \mathbf{F} \tag{60}$$

La matriz $\mathbf{F_{14}}$**,** de 7x1 corresponde los valores de las fuerzas de la plataforma móvil, extraídas de resolver la dinámica inversa en dicha plataforma. La matriz $\boldsymbol{\Gamma}$**,** de dimensión 7x6, es la matriz correspondiente a los términos que se desarrollarán a continuación. Cada columna de esta matriz representa los 7 términos que se han identificando multiplicando a las 7 variables independientes de la plataforma móvil. Cada pistón tiene sus 7 términos identificados.

Por último, la última matriz corresponde al vector de fuerzas en los pistones que habrá que calcular. De esta manera, se trata de resolver dicho sistema.

Para el pistón 3-8 por ejemplo, tenemos, aplicando movimiento relativo:

$$\mathbf{v_{38}^{G8}} = \mathbf{v_{31}^{G3}} - \mathbf{v_{81}^{G8}} + \boldsymbol{\omega}_{\mathbf{31}} \mathbf{x} \overrightarrow{\mathbf{G_3 G_8}} \tag{61}$$

Multiplicando en ambas partes por dt obtenemos:

$$\boldsymbol{\delta q_{38}^{G8}} = \boldsymbol{A^T}(\delta \boldsymbol{r_{31}^{G3}} - \delta \boldsymbol{r_{81}^{G8}} + (\boldsymbol{\omega_{31}} \mathbf{x} \overrightarrow{\mathbf{G_3 G_8}})dt) \quad (62)$$

Donde la velocidad angular se expresa cómo:

$$\boldsymbol{\omega_{31}} = 2\mathbf{G}\dot{\mathbf{q}}_{\mathbf{31}} = 2\begin{bmatrix} -e_1 & e_0 & -e_3 & e_2 \\ -e_2 & e_3 & e_0 & -e_1 \\ -e_3 & -e_2 & e_1 & e_0 \end{bmatrix}\begin{bmatrix} \dot{q}_{18} \\ \dot{q}_{19} \\ \dot{q}_{20} \\ \dot{q}_{21} \end{bmatrix}$$

Donde $q_{18}, q_{19}, q_{20}, q_{21}$ son los parámetros de Euler del sólido 3.

$$\delta \boldsymbol{r_{31}^{G3}} - \delta \boldsymbol{r_{81}^{G8}} = \begin{bmatrix} \delta q_{50} - \delta q_{15} \\ \delta q_{51} - \delta q_{16} \\ \delta q_{52} - \delta q_{17} \end{bmatrix}$$

La expresión del producto vectorial quedaría:

$$\begin{aligned} &(\boldsymbol{\omega_{31}} \mathbf{x} \overrightarrow{\mathbf{G_3 G_8}})dt \\ &= \begin{bmatrix} (-e_2\delta q_{18} + e_3\delta q_{19} + e_0\delta q_{20} - e_1\delta q_{21})(q_{52} - q_{17}) - (e_3\delta q_{18} - e_2\delta q_{19} + e_1\delta q_{20} + e_0\delta q_{21})(q_{51} - q_{16}) \\ (e_1\delta q_{18} - e_0\delta q_{19} - e_3\delta q_{20} + e_2\delta q_{21})(q_{52} - q_{17}) + (-e_3\delta q_{18} - e_2\delta q_{19} + e_1\delta q_{20} + e_0\delta q_{21})(q_{50} - q_{15}) \\ (-e_1\delta q_{18} + e_0\delta q_{19} - e_3\delta q_{20} + e_2\delta q_{21})(q_{51} - q_{16}) + (e_2\delta q_{18} - e_3\delta q_{19} + e_0\delta q_{20} - e_1\delta q_{21})(q_{50} - q_{15}) \end{bmatrix} \end{aligned}$$

Lo único que quedaría sería premultiplicar por la traspuesta en ambos términos e identificar términos de manera que resultara:

$$[a \quad b \quad c \quad d \quad e \quad f \quad g \quad h \quad i \quad j]\begin{bmatrix} \delta q_{15} \\ \delta q_{16} \\ \delta q_{17} \\ \delta q_{18} \\ \delta q_{19} \\ \delta q_{20} \\ \delta q_{21} \\ \delta q_{50} \\ \delta q_{51} \\ \delta q_{52} \end{bmatrix}$$

Donde cada término a,b,c… son términos agrupados que multiplican a cada uno de los miembros de la segunda matriz.

Es sabido que la fuerza del pistón que interesa es la fuerza en la dirección del eje z, por lo que se obtendrá sólo la tercera componente de dicho vector.

Como estos vectores están en función de las variables dependientes del problema (todas las demás excepto las de la plataforma móvil) hay que poner dichas variables dependientes en función de las independientes de la siguiente forma:

1) Se eliminan las ultimas 7 filas del jacobiano, correspondientes a las variables independientes. Dicho jacobiano quedaría de dimensión 91x98.

2) Se cumple que: $\boldsymbol{\Phi_q \dot{q}} = \mathbf{0}$, ya que se han eliminado del jacobiano las filas correspondientes a las ecuaciones de movilidad.

3) Se tiene por tanto que:

$$\Phi_{qd}\delta q_d + \Phi_{qi}\delta q_i = 0 \tag{63}$$

$$\delta q_d = -(\Phi_{qd})^{-1}\Phi_{qi}\delta q_i \tag{64}$$

3 RESULTADOS.

4.1. Cinemática y dinámica de un mecanismo biela manivela

Como recordatorio, se tenía un mecanismo de manivela-biela-corredera de un grado de libertad. Por lo tanto, se precisan 12 ecuaciones para completar el sistema de 12 ecuaciones con 12 incógnitas. Los pares proporcionan 11 de estas, por lo que será necesario una ecuación de movilidad que imponga el movimiento del mecanismo. En este caso se ha optado por hacer que la manivela gire a una velocidad angular constante, quedando así el mecanismo completamente definido. El tiempo en el que se realiza dicha simulación se ha elegido de manera que al final de este la manivela haya dado dos vueltas completas.

Se representará la posición de la corredera, que deberá ser, si todo está correcto, nula en el eje y, variable entre dos valores extremos en el eje x, y nula en el ángulo:

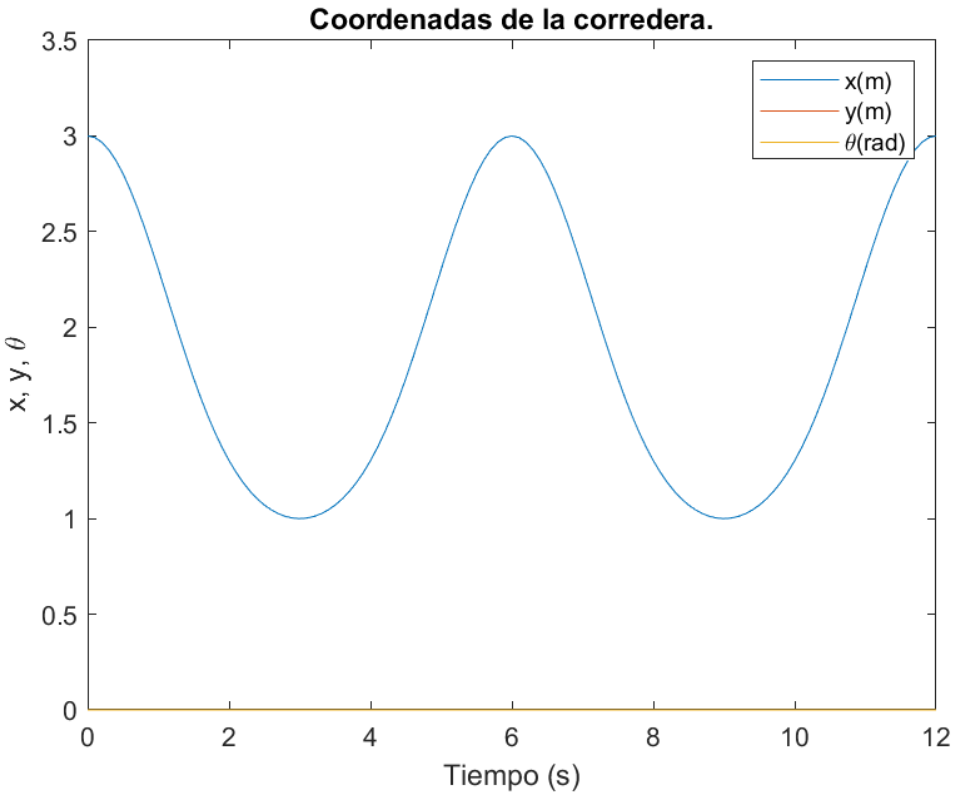


Imagen 3.1: Representación de la posición de la corredera.

Efectivamente, al ser la barra de la manivela de longitud 1 metro y la biela de 2, dicha corredera oscila entre los valores de x entre 1 y 3, mientras que tanto el ángulo, como el valor de y son nulos.

En cuanto a las velocidades, se obtienen los siguientes resultados.

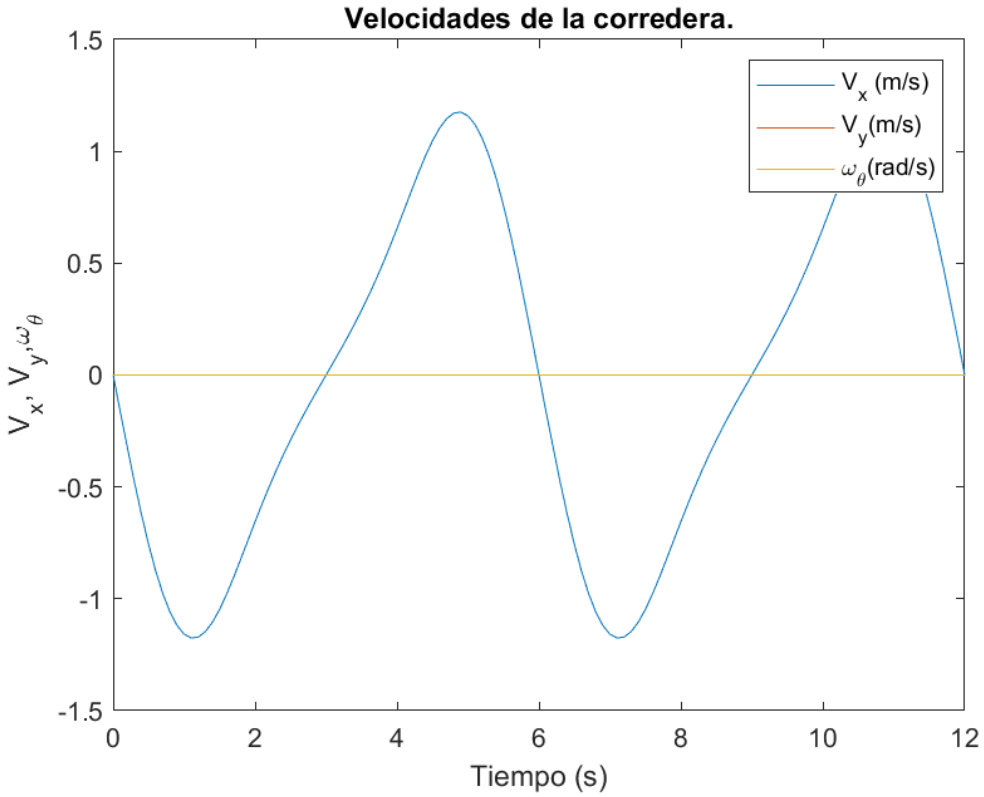


Imagen 3.2: Representación de la velocidad de la corredera.

Por último, los resultados para la aceleración en la corredera es el siguiente:

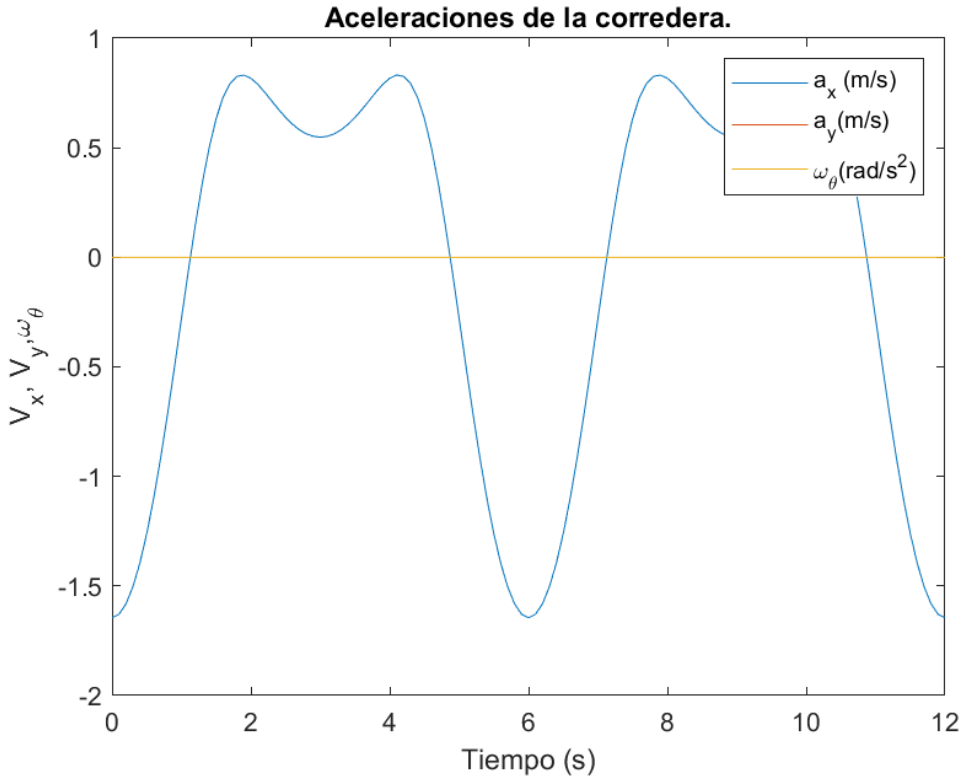


Imagen 3.3: Representación de la aceleración de la corredera.

Como comprobación, podemos representar velocidad y aceleración a lo largo del eje x de la corredera. El resultado debe ser que, aproximadamente, el máximo de velocidades coincida con valores cercanos a cero de las aceleraciones.

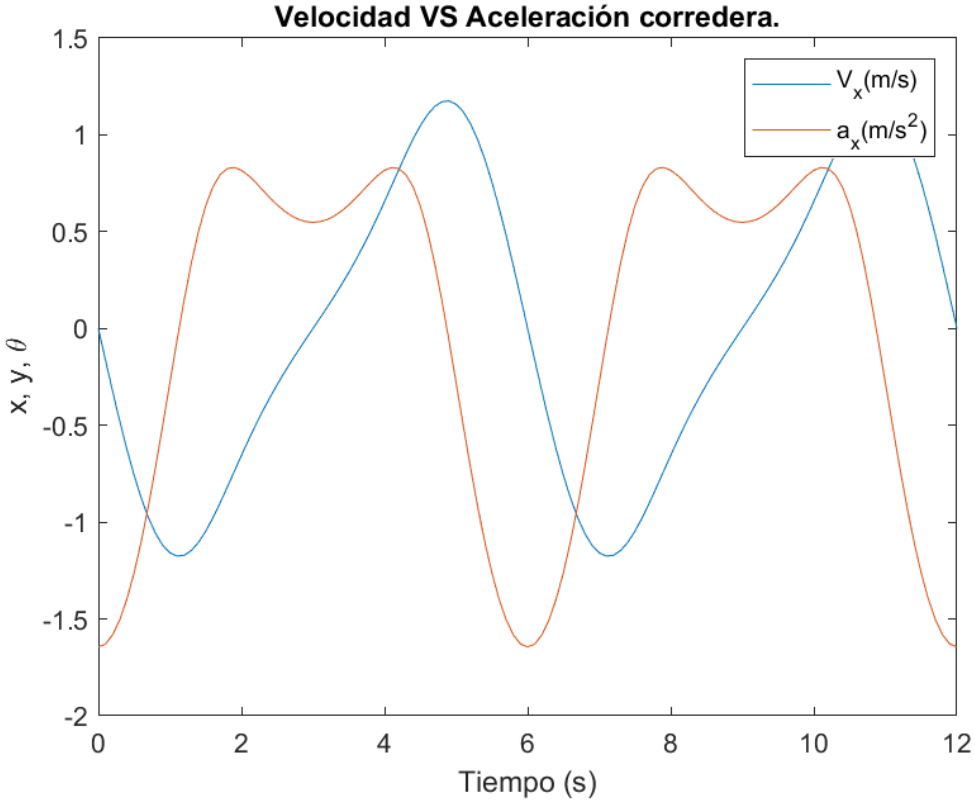


Imagen 3.4: Comparación velocidad y aceleración de la corredera en el eje X.

Para el análisis dinámico del mecanismo 2D presente en este trabajo, se montará la matriz M teniendo en cuenta las masas que se aprecian en el código correspondiente. La matriz g estará compuesta sólo y exclusivamente por la fuerza de la gravedad. Una vez montadas estas matrices y resuelto el problema cinemático, no hay más que realizar el mismo procedimiento, recorriendo de nuevo todos los tiempos recorridos en el cinemático. Se calculará para cada instante de tiempo un vector $\boldsymbol{\lambda}$.

El objetivo de calcular la dinámica en este sencillo mecanismo ha sido el de calcular el par motor realizado por la manivela. Es decir, el par que es necesario darle a dicha manivela para que realice el movimiento impuesto en el problema cinemático. Para calcular esto, y teniendo en cuenta la definición del vector $\boldsymbol{\lambda}$, la ecuación de restricción que ha impuesto el movimiento de la manivela en este mecanismo ha sido la ecuación de movilidad (la número 12). Por tanto, para conocer el valor de dicho par motor se procede de la siguiente forma:

1) Se calcula el vector $\boldsymbol{\lambda}$ mediante el procedimiento anteriormente descrito.
2) Se hacen 0 todos los componentes de $\boldsymbol{\lambda}$ excepto el correspondiente a la ecuación de movilidad (el número 12)
3) Se opera de la siguiente forma:

$$\mathbf{g_1} = \mathbf{C_q^T \lambda_{12}}$$

4) Se escoge del vector $\mathbf{g_1}$ la sexta componente, correspondiente al ángulo de la manivela. Éste es el par motor.

Realizado este procedimiento numéricamente, que puede observarse en el apartado códigos del presente documento, se obtiene el siguiente resultado:

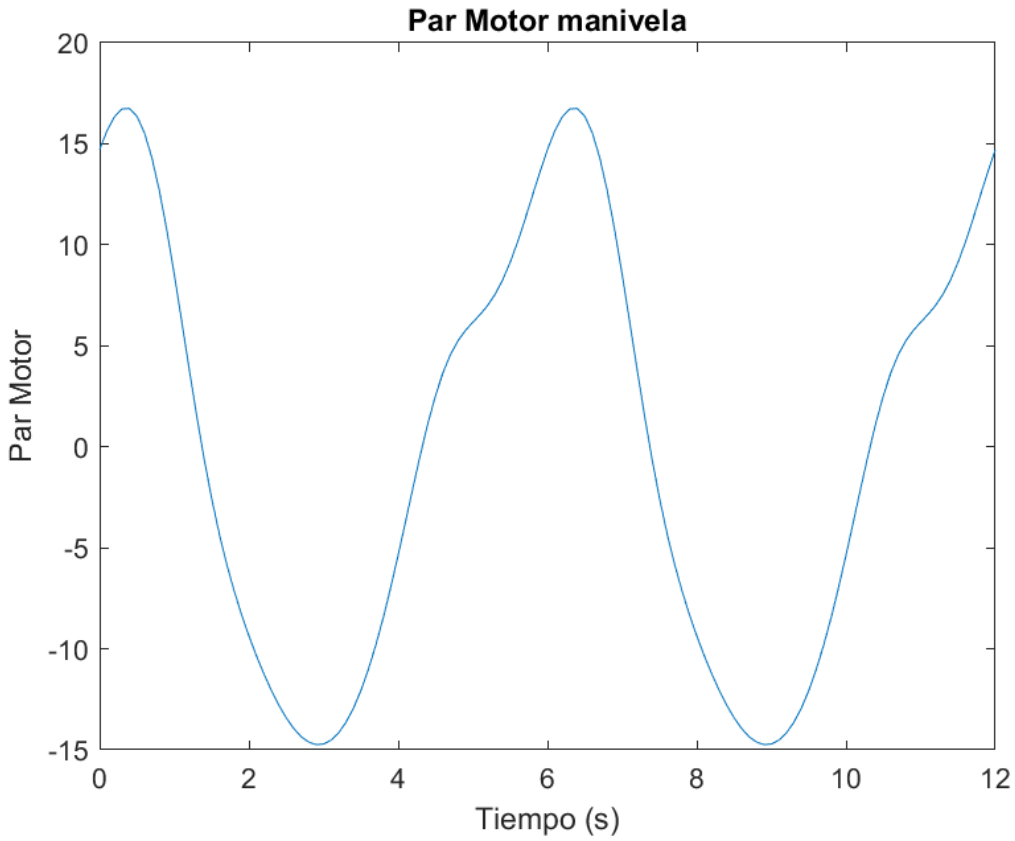
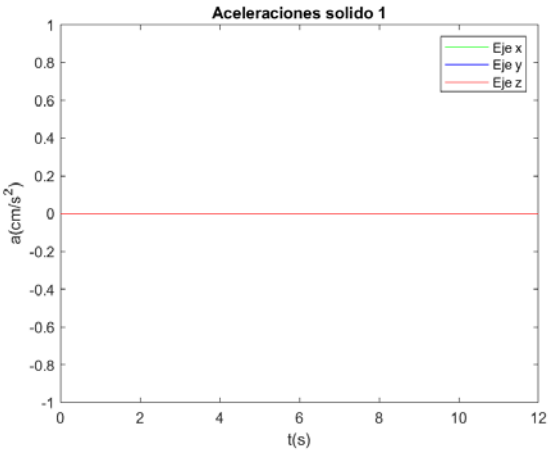
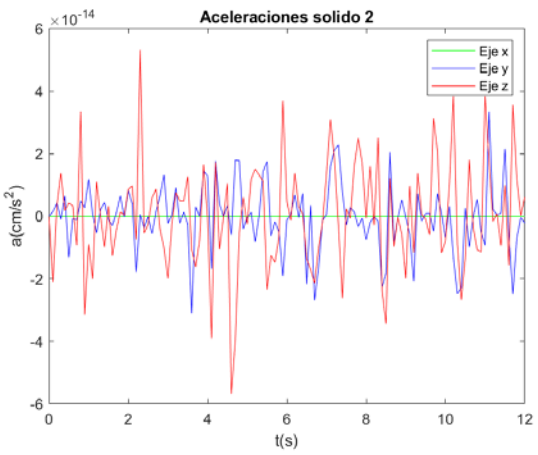


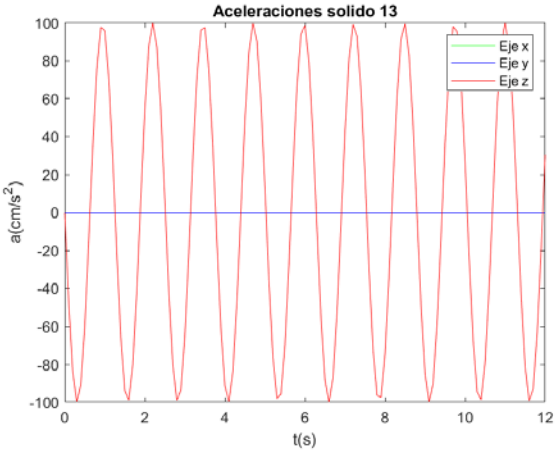
Imagen 3.5: Representación del par motor con respecto al tiempo.

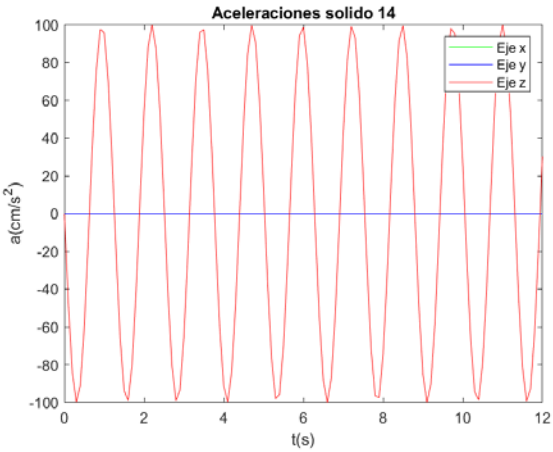
4.2. Resultados de la cinemática y dinámica de la plataforma de Stewart.

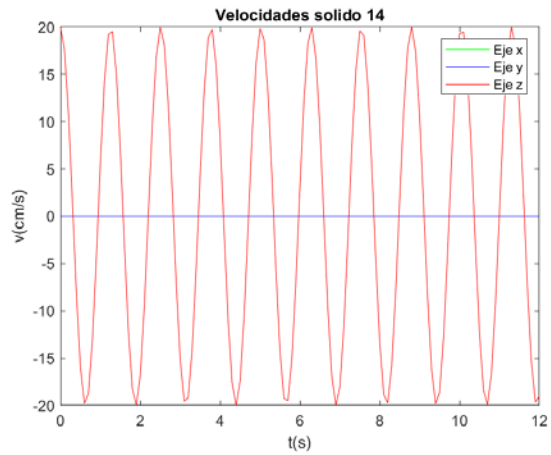
Primeramente, se ha optado por representar movimientos sencillos para ver si la implementación era correcta. Para ello, se ha modelado una plataforma de Stewart cuyas barras son todas rectas y no están inclinadas respecto a la plataforma. De esta forma, todas las velocidades y aceleraciones de todas las barras pegadas a la plataforma móvil deberían ser iguales. Las velocidades y aceleraciones en las barras pegadas a la plataforma fija deben de ser nulas. Efectivamente, esto es así y el resultado es el siguiente expuesto. El sólido 1 es la plataforma fija, el 2 una de las barras pegadas a la fija mediante el par Cardan, el 13 una de las barras pegadas a la plataforma móvil y el 14 la plataforma móvil. Por tanto, las barras de la 3 a la 7 tienen el mismo resultado que la 2 y las barras de la 8 a las 12 tienen el mismo resultado que la 13.

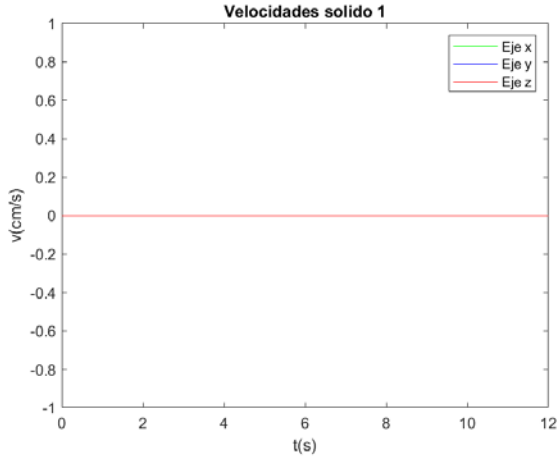

Aceleraciones solido 1
Eje x
Eje y
Eje z
a(cm/s^2)
t(s)


Aceleraciones solido 2
×10^-14
Eje x
Eje y
Eje z
a(cm/s^2)
t(s)


Aceleraciones solido 13
Eje x
Eje y
Eje z
a(cm/s^2)
t(s)


Aceleraciones solido 14
Eje x
Eje y
Eje z
a(cm/s^2)
t(s)


Velocidades solido 14
Eje x
Eje y
Eje z
v(cm/s)
t(s)


Velocidades solido 1
Eje x
Eje y
Eje z
v(cm/s)
t(s)

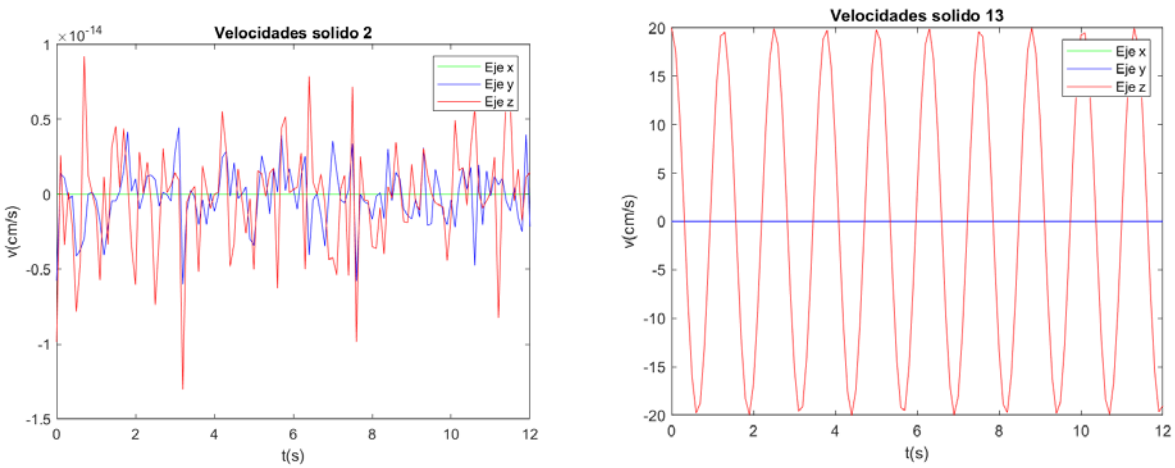
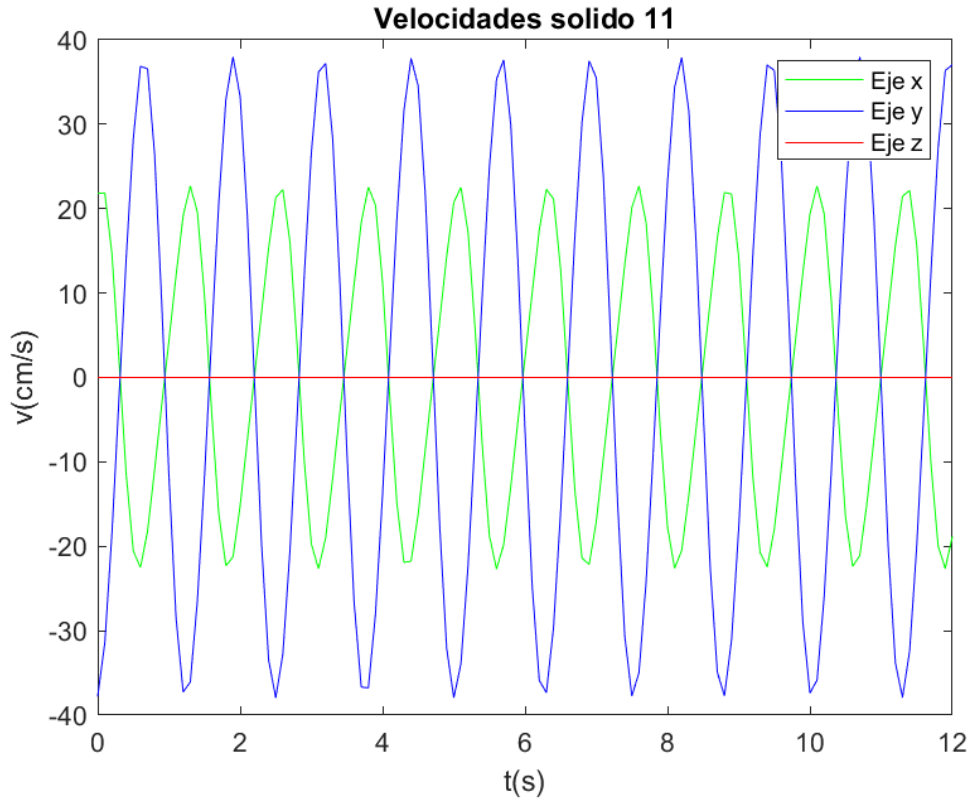


Imagen 3.6: Representación cinemática plataforma de Stewart.

Posteriormente se representa la cinemática de la plataforma de Stewart real, con sus barras inclinadas. Se representan los tres movimientos básicos de una aeronave: guiñada, cabeceo y alabeo. Para cada actuación, se va a representar la velocidad y aceleración de alguna de las barras pegadas a la plataforma:



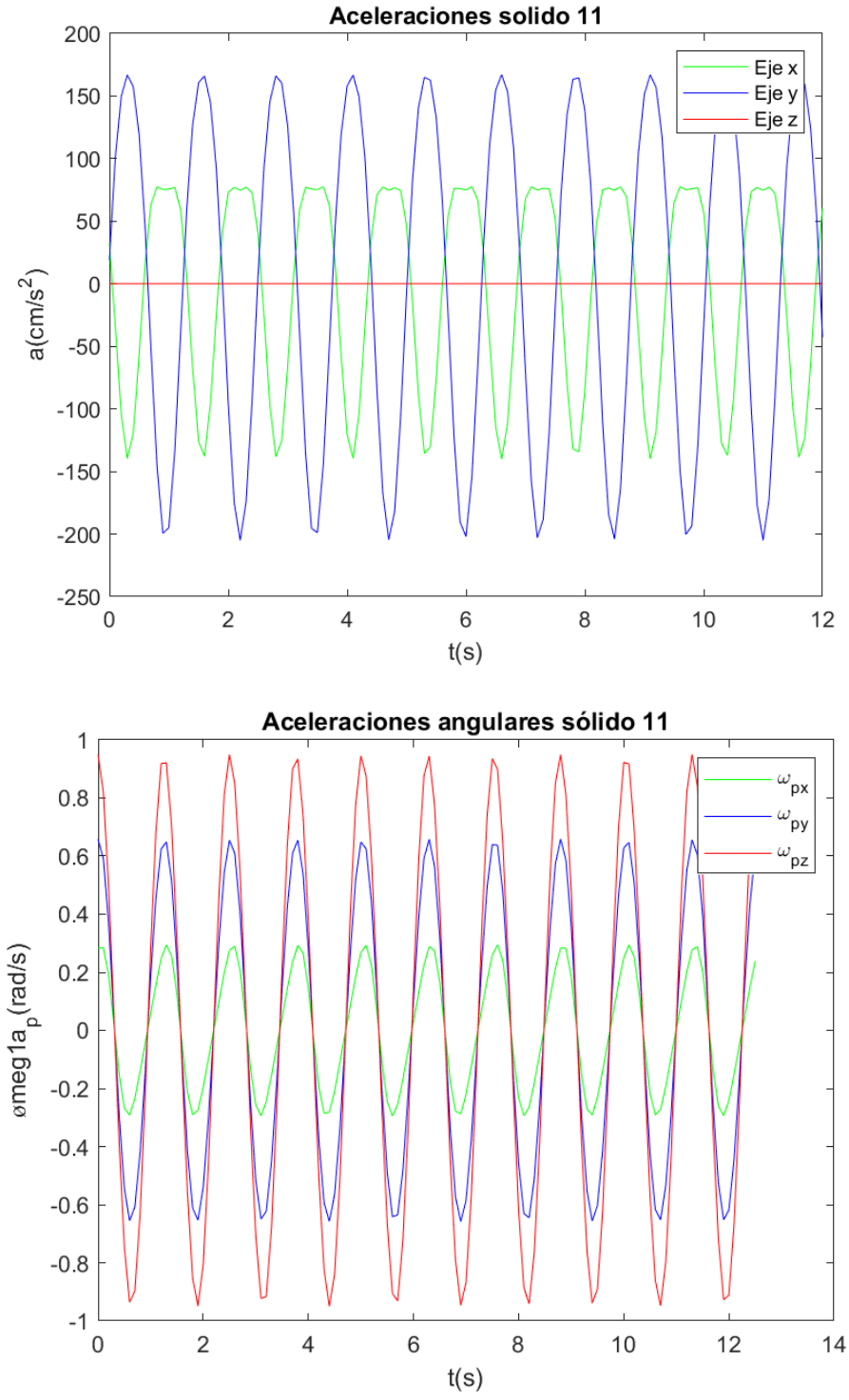
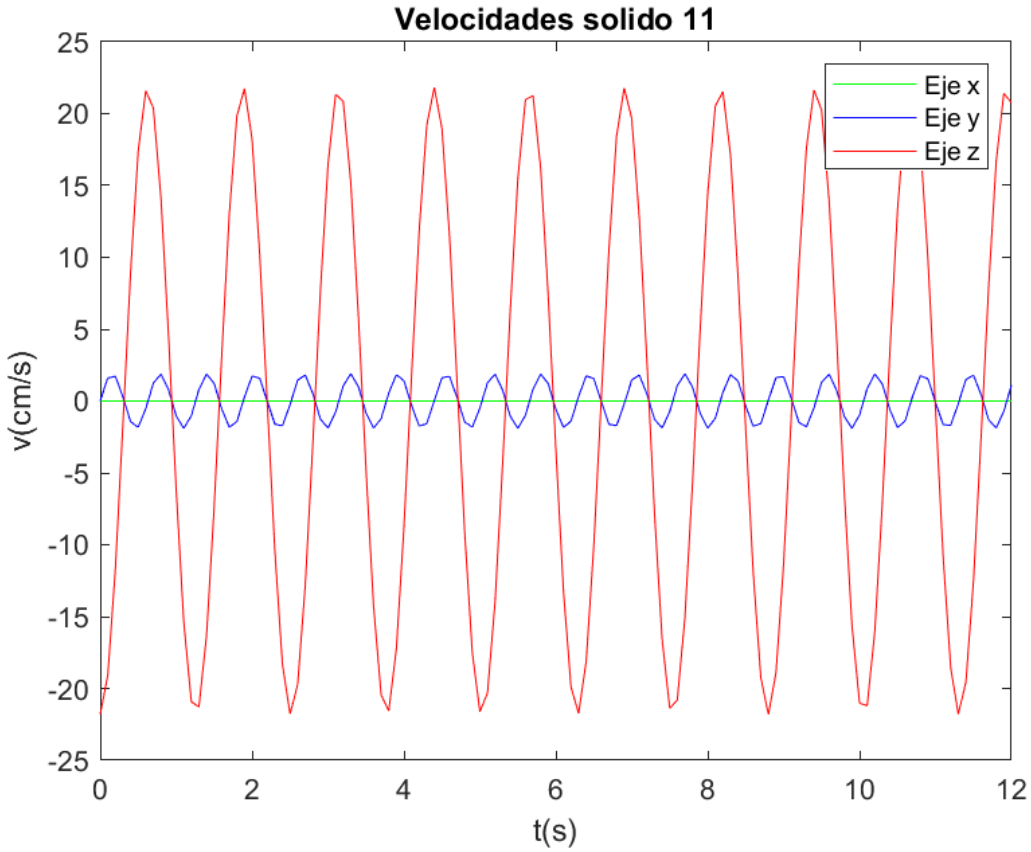
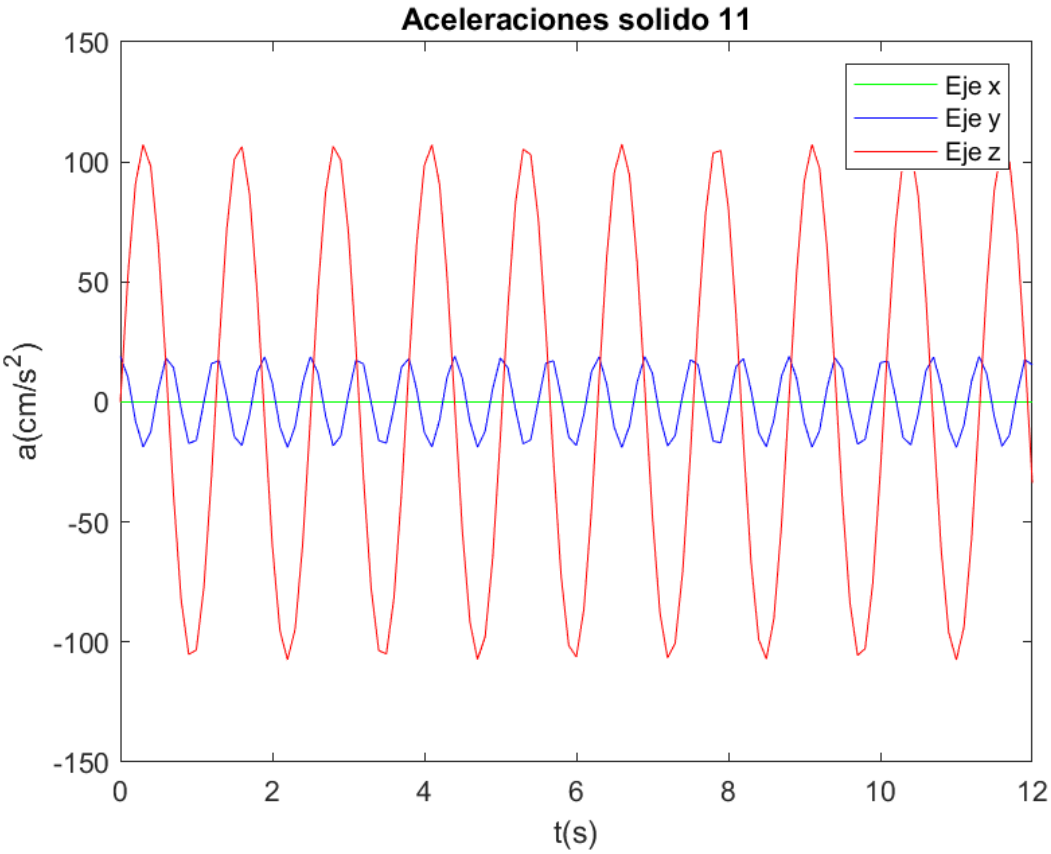


Imagen 3.7: Resultados guiñada.


Velocidades solido 11
Eje x
Eje y
Eje z
v(cm/s)
t(s)
25
20
15
10
5
0
-5
-10
-15
-20
-25
0
2
4
6
8
10
12


Aceleraciones solido 11
Eje x
Eje y
Eje z
a(cm/s^2)
t(s)
150
100
50
0
-50
-100
-150
0
2
4
6
8
10
12

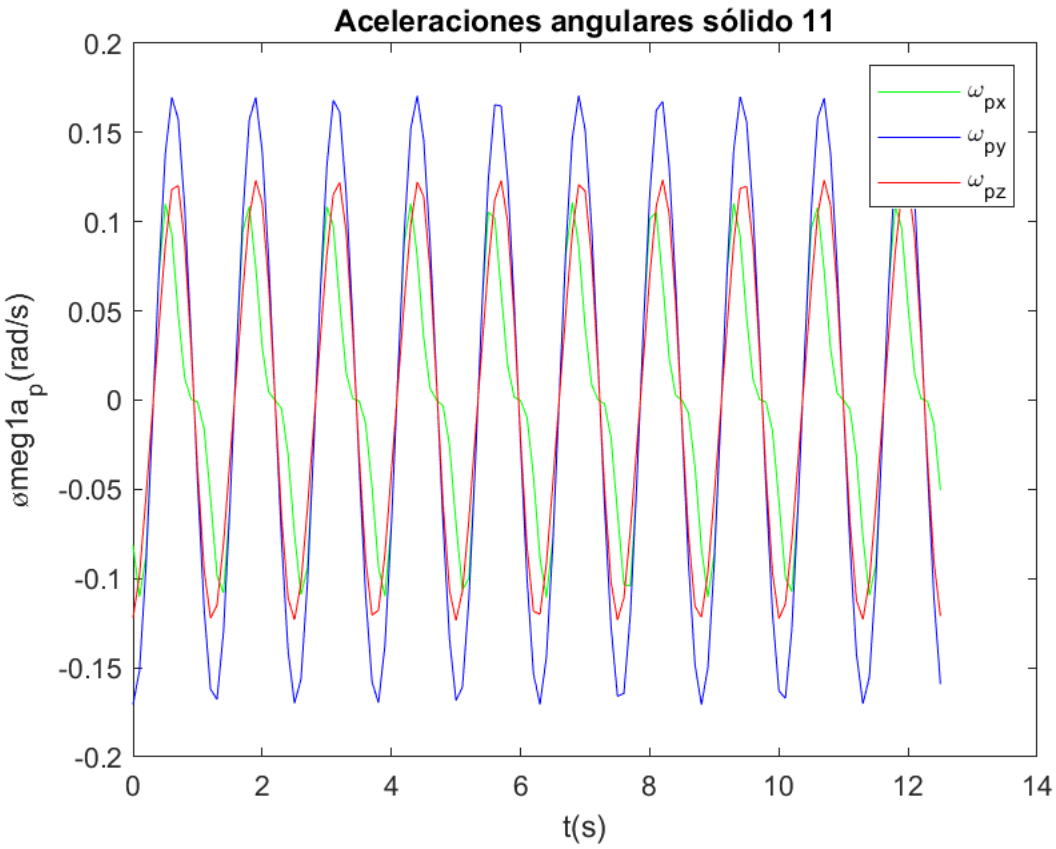
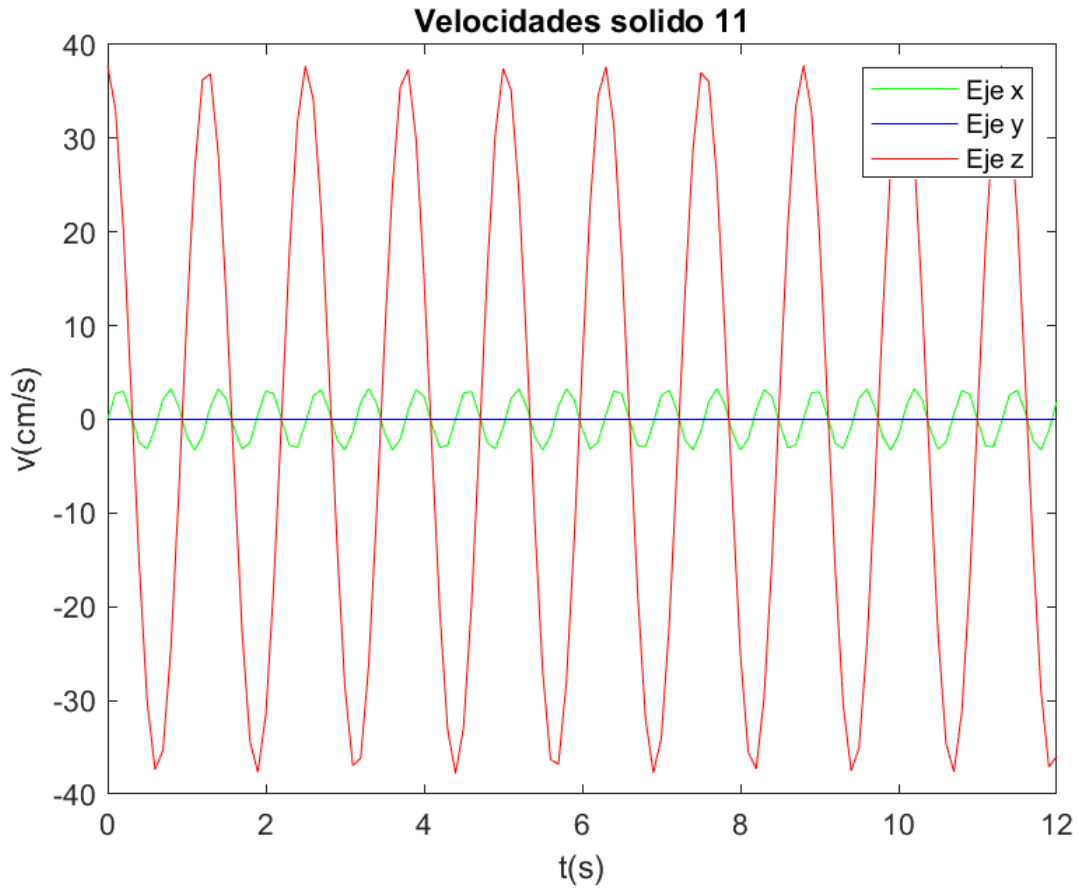
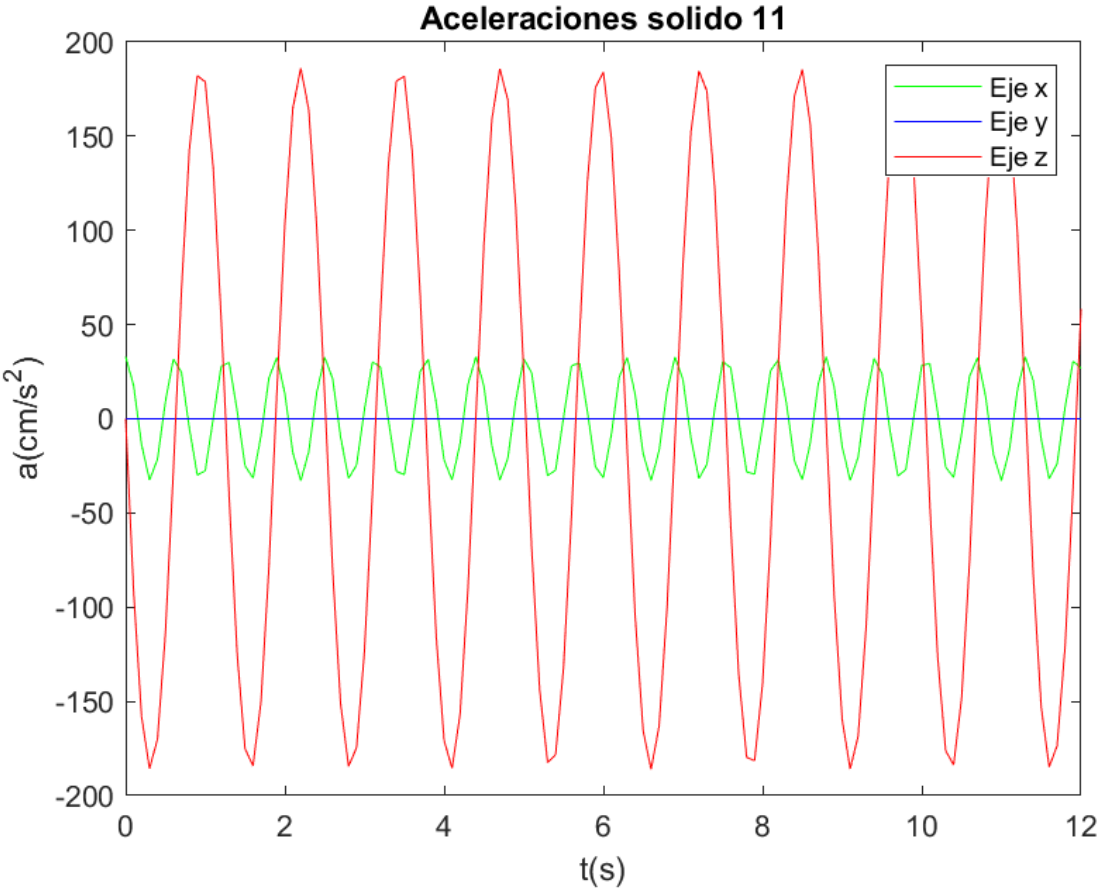


Imagen 3.8: Resultados alabeo.


Velocidades solido 11
Eje x
Eje y
Eje z
v(cm/s)
t(s)
40
30
20
10
0
-10
-20
-30
-40
0
2
4
6
8
10
12


Aceleraciones solido 11
Eje x
Eje y
Eje z
a(cm/s2)
t(s)
200
150
100
50
0
-50
-100
-150
-200
0
2
4
6
8
10
12

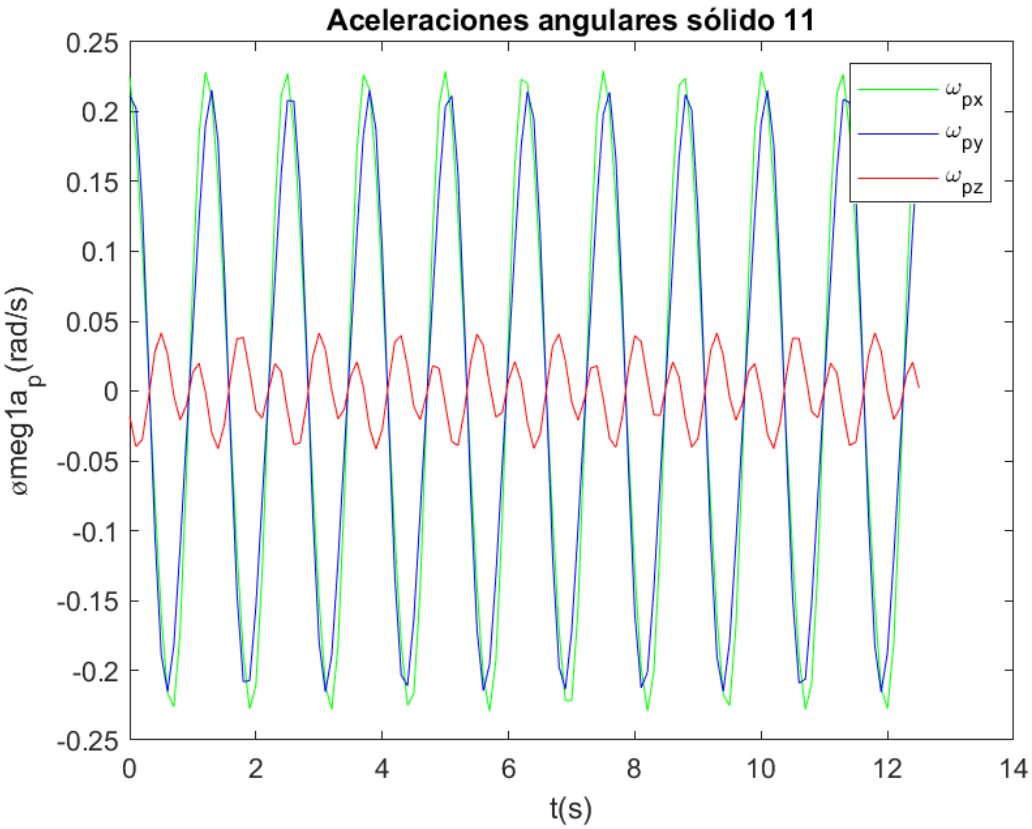


Imagen 3.9: Resultados cabeceo.

Una vez expresados algunos de los resultados de la cinemática, se procede a la dinámica, donde los resultados que se expondrán a continuación serán, la fuerza que es necesario aplicar en el centro de la plataforma móvil para permitir estos movimientos básicos de una aeronave. Como se expuso en el capítulo anterior, los sistemas de coordenadas de las barras han cambiado a su centro de gravedad y formando ejes principales de inercia.

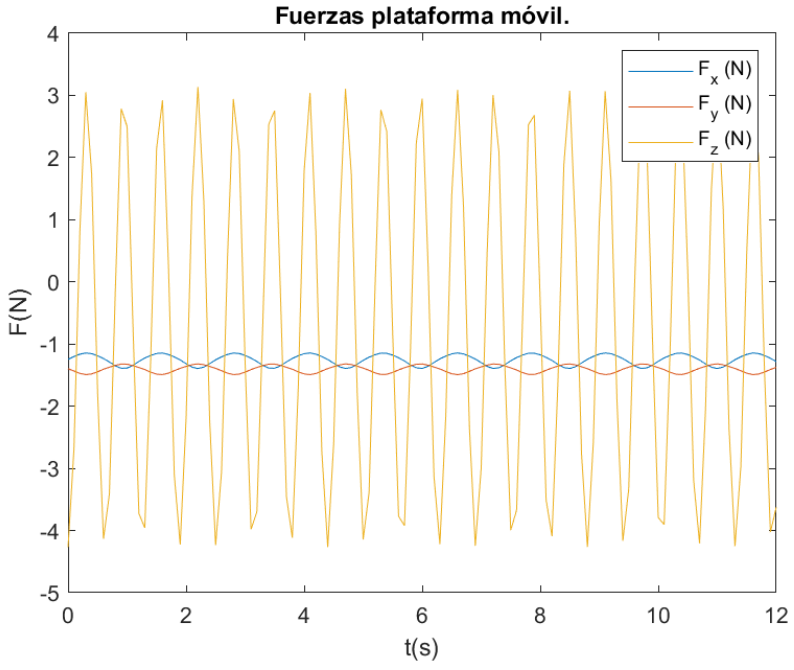


Imagen 3.10: Resultados de fuerzas en guiñada.

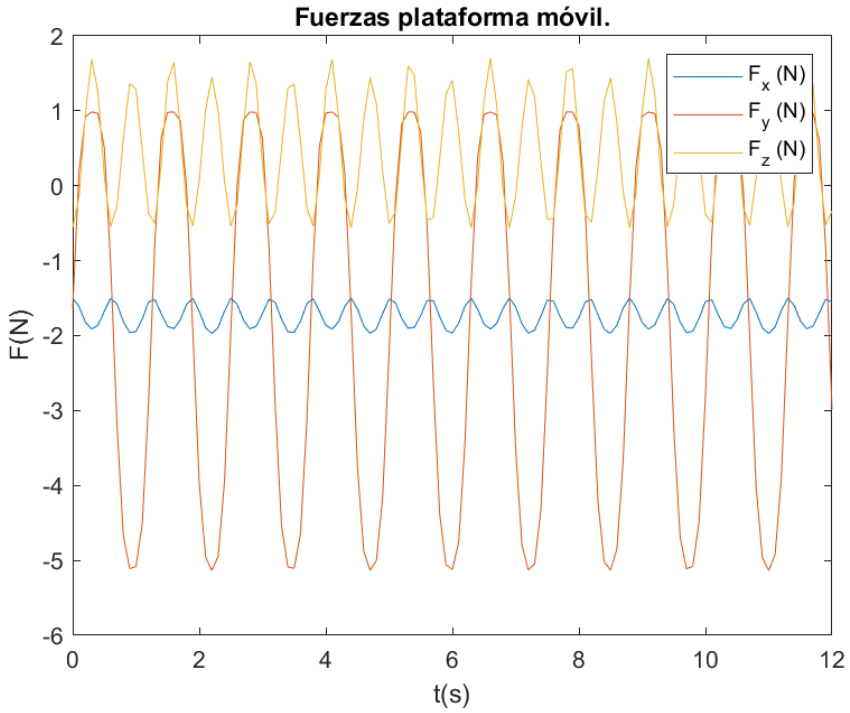


Imagen 3.11: Resultados de fuerzas en alabeo.

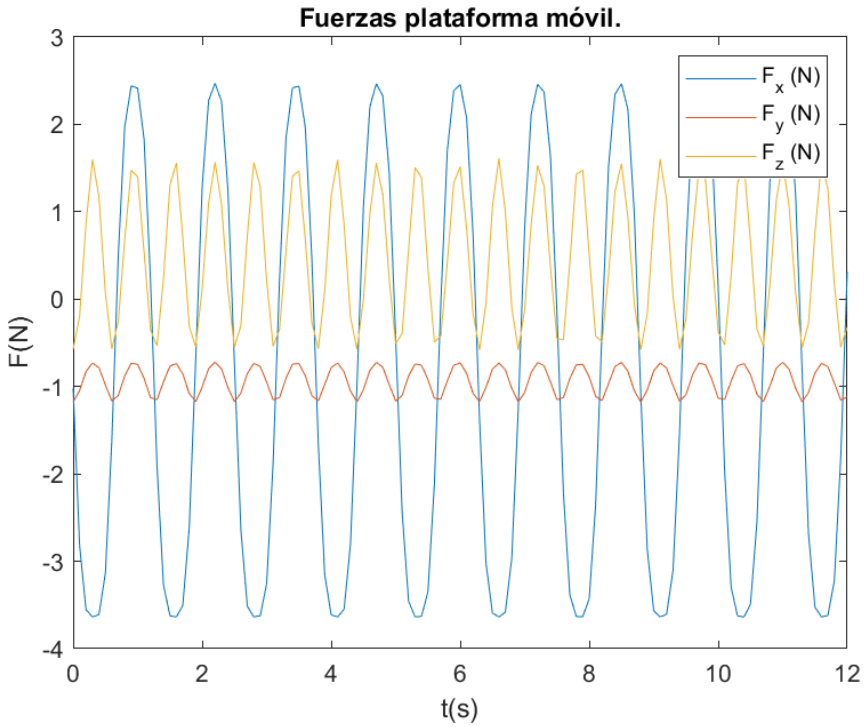
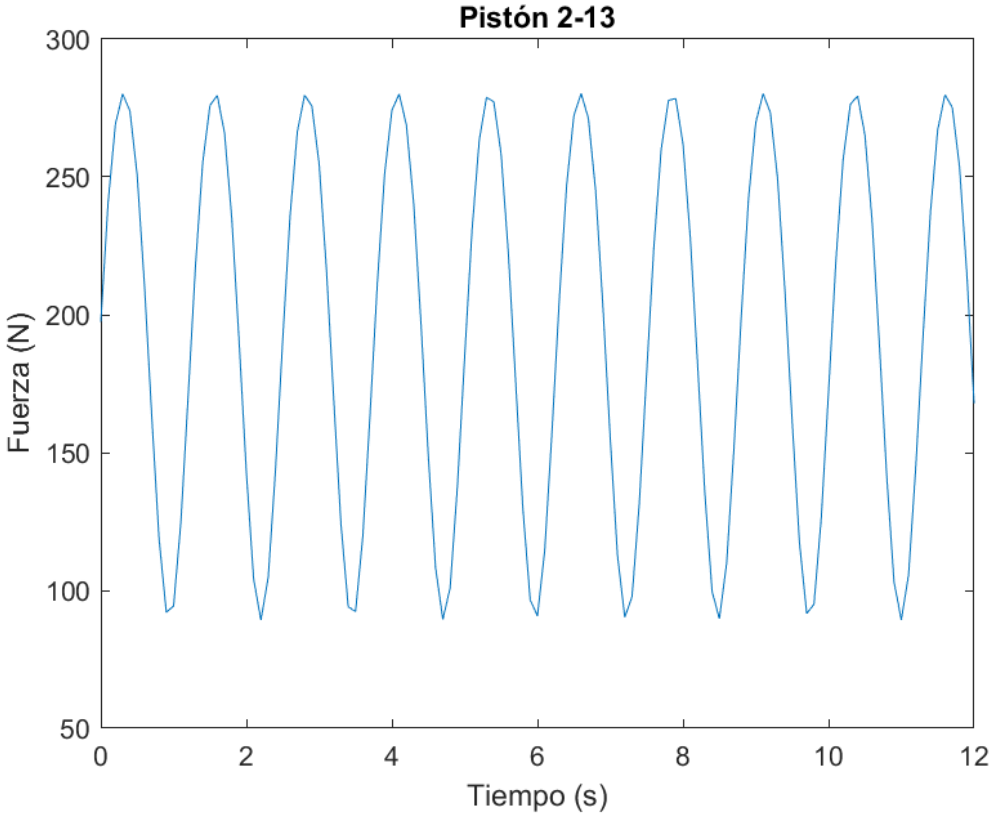
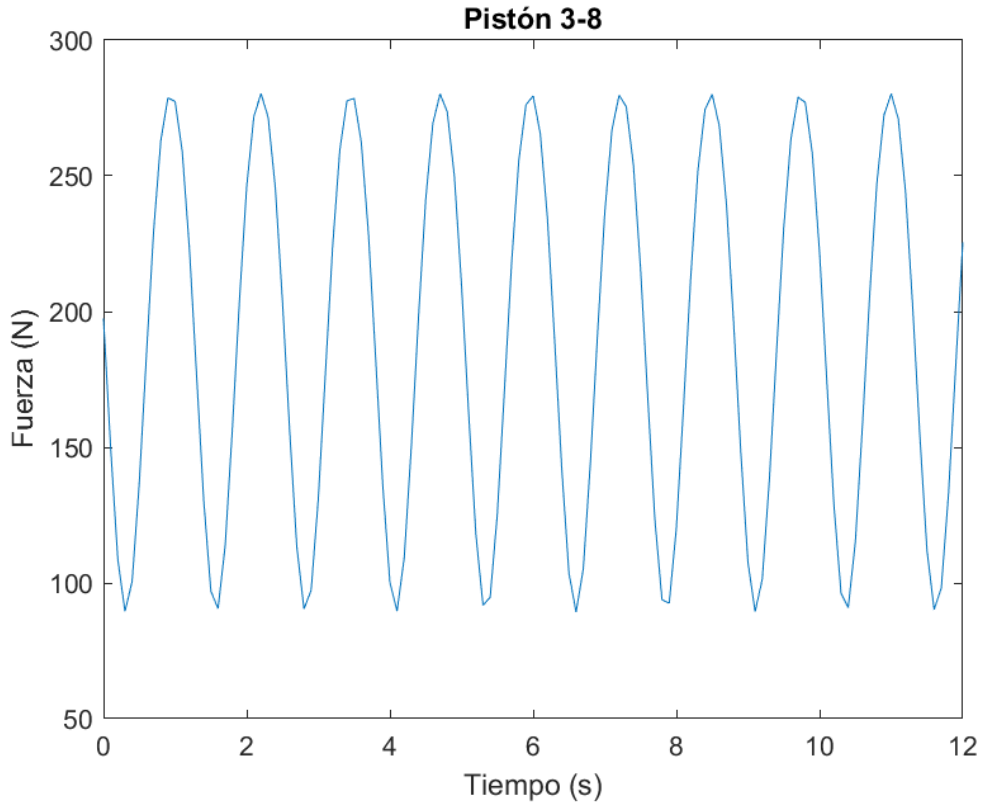


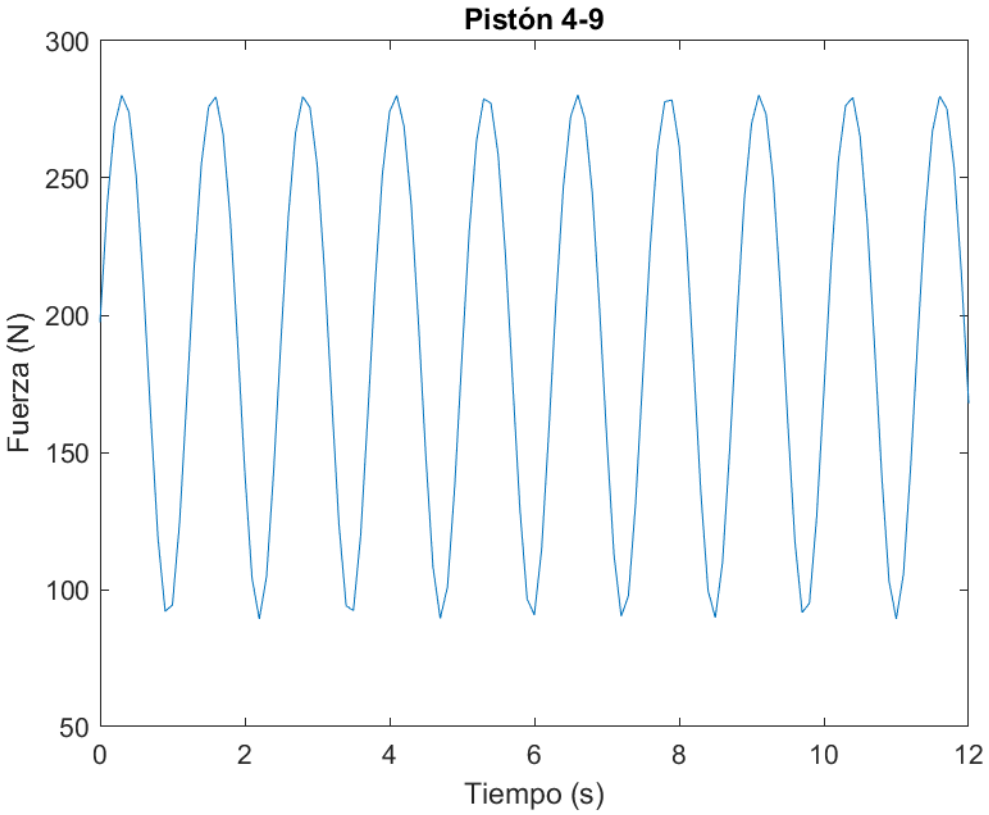
Imagen 3.12: Resultado de fuerzas en cabeceo.

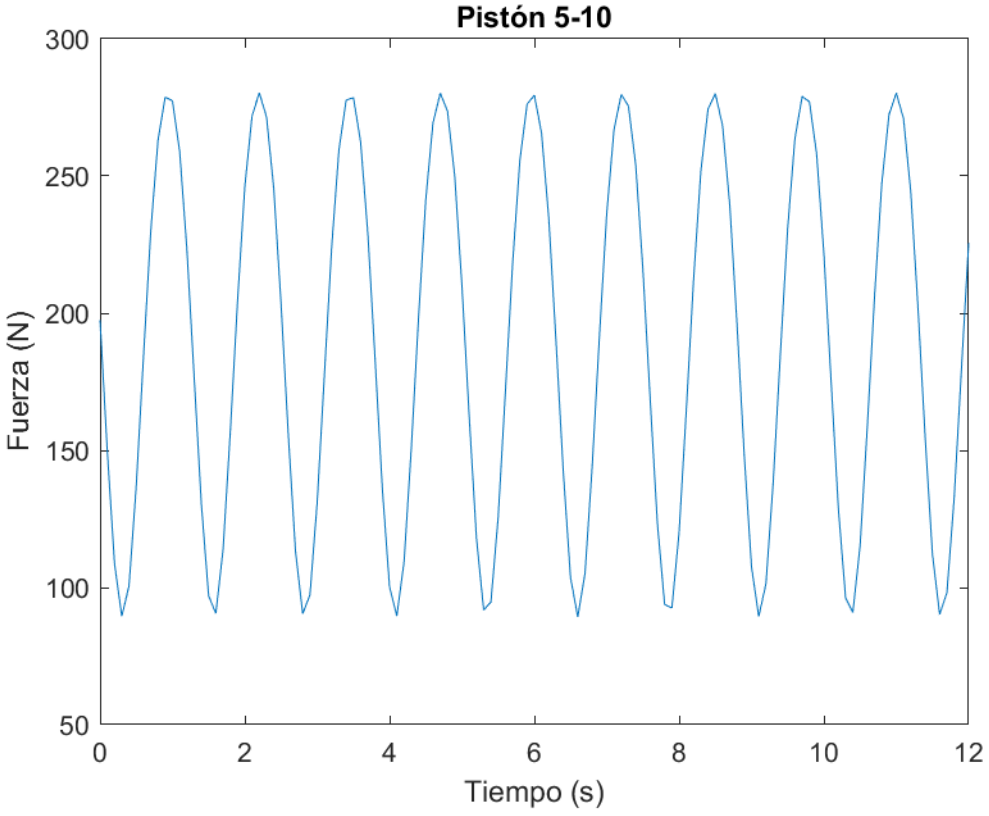
4.3. Resultados fuerza en los pistones.

Se muestran a continuación los resultados de las fuerzas en los pistones para las diferentes maniobras del avión (guiñada, alabeo y cabeceo):






Pistón 4-9
Fuerza (N)
Tiempo (s)
300
250
200
150
100
50
0
2
4
6
8
10
12


Pistón 5-10
Fuerza (N)
Tiempo (s)
300
250
200
150
100
50
0
2
4
6
8
10
12

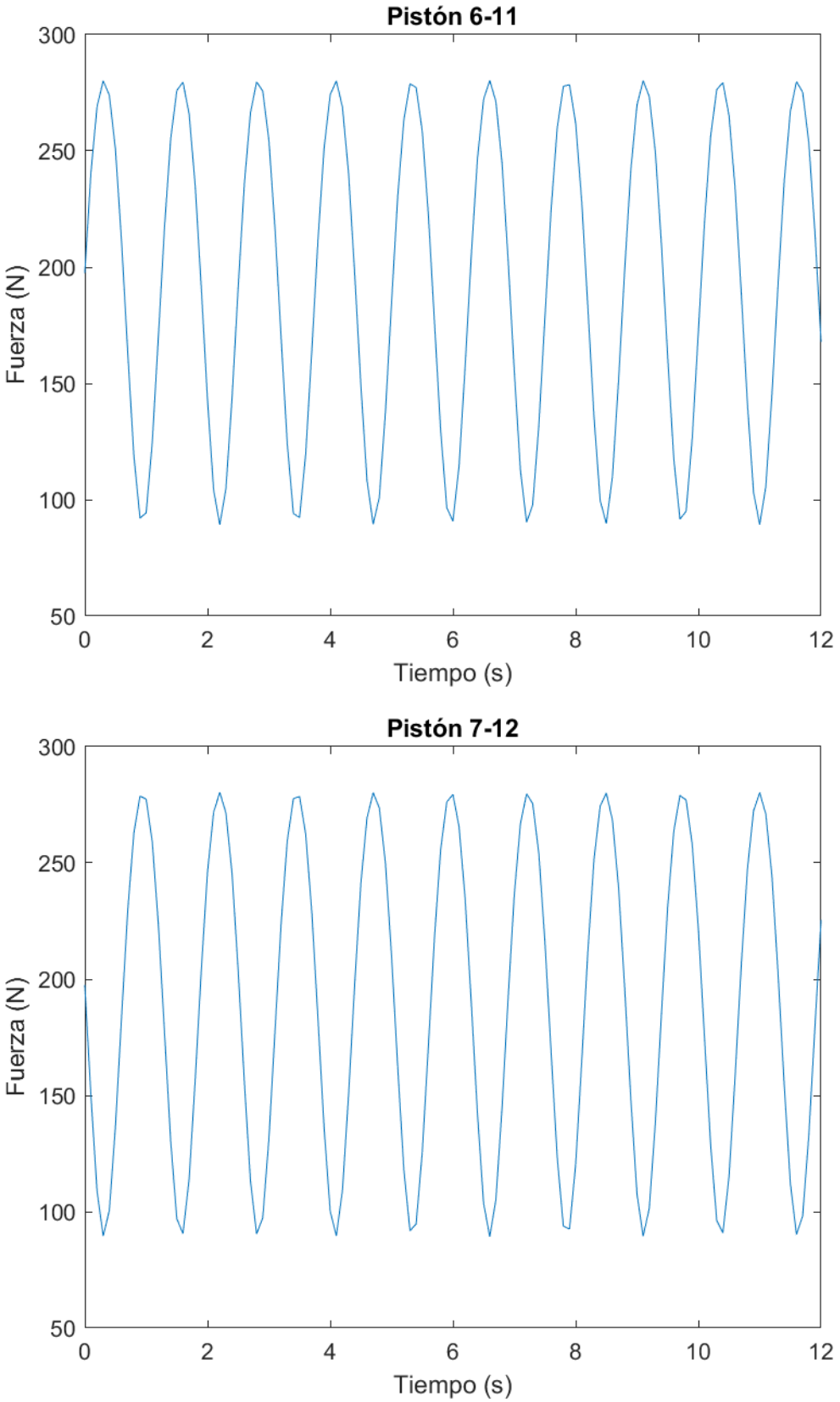
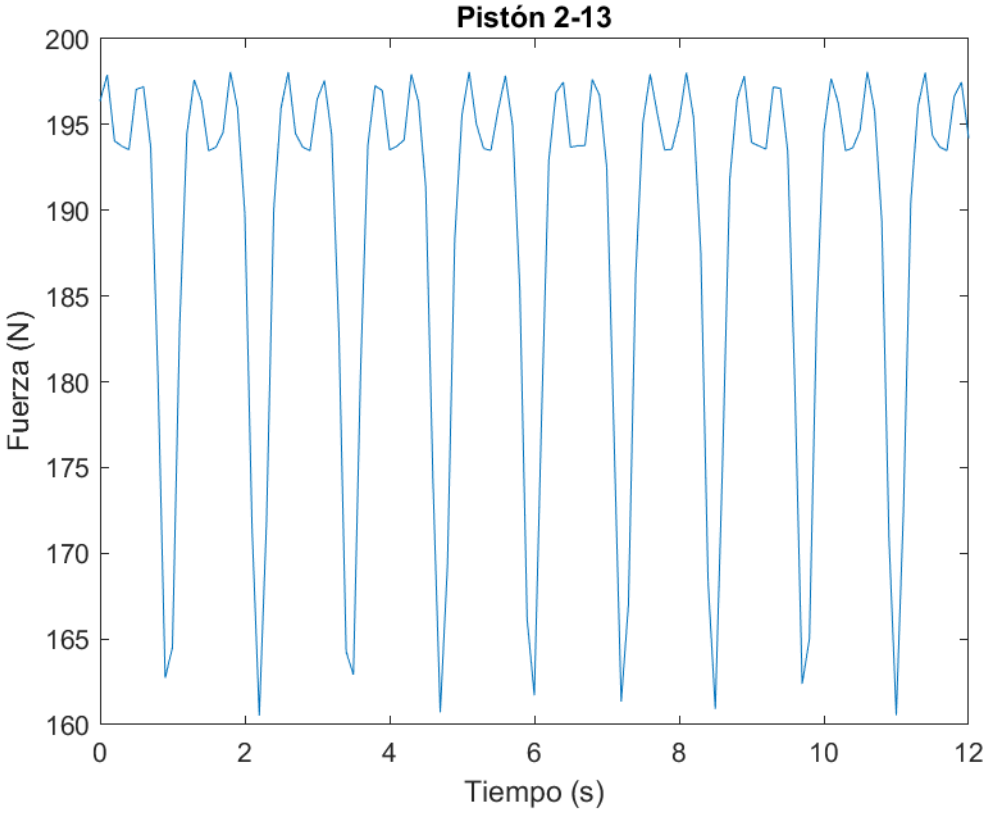
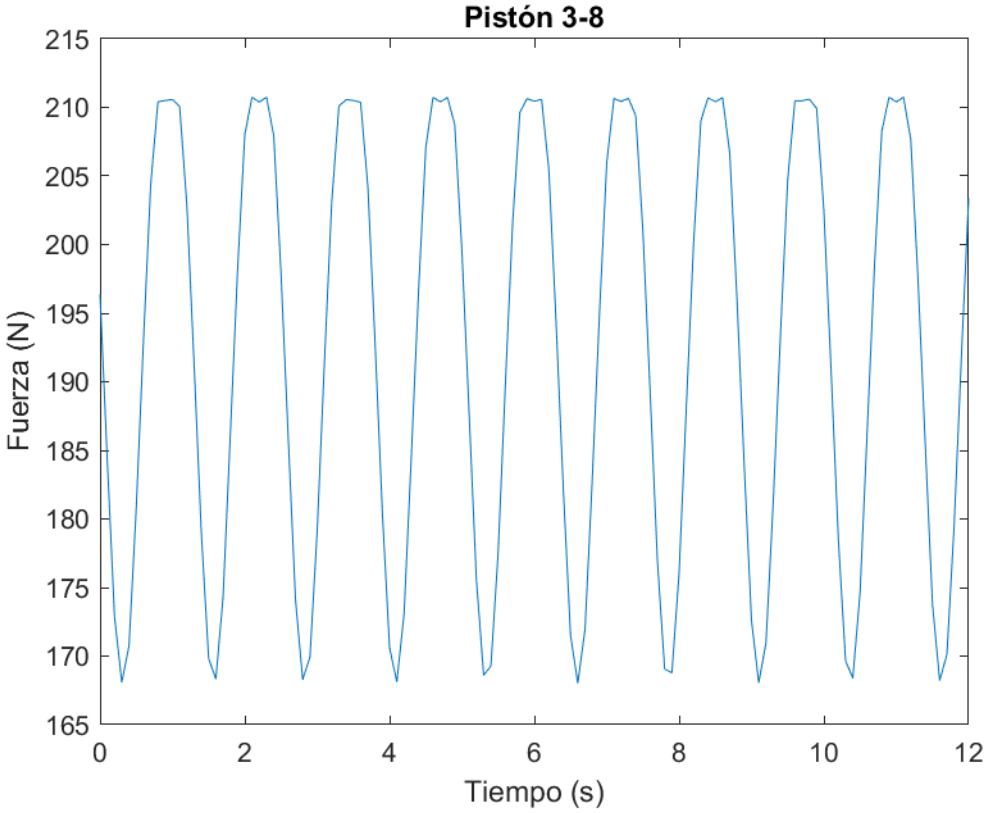
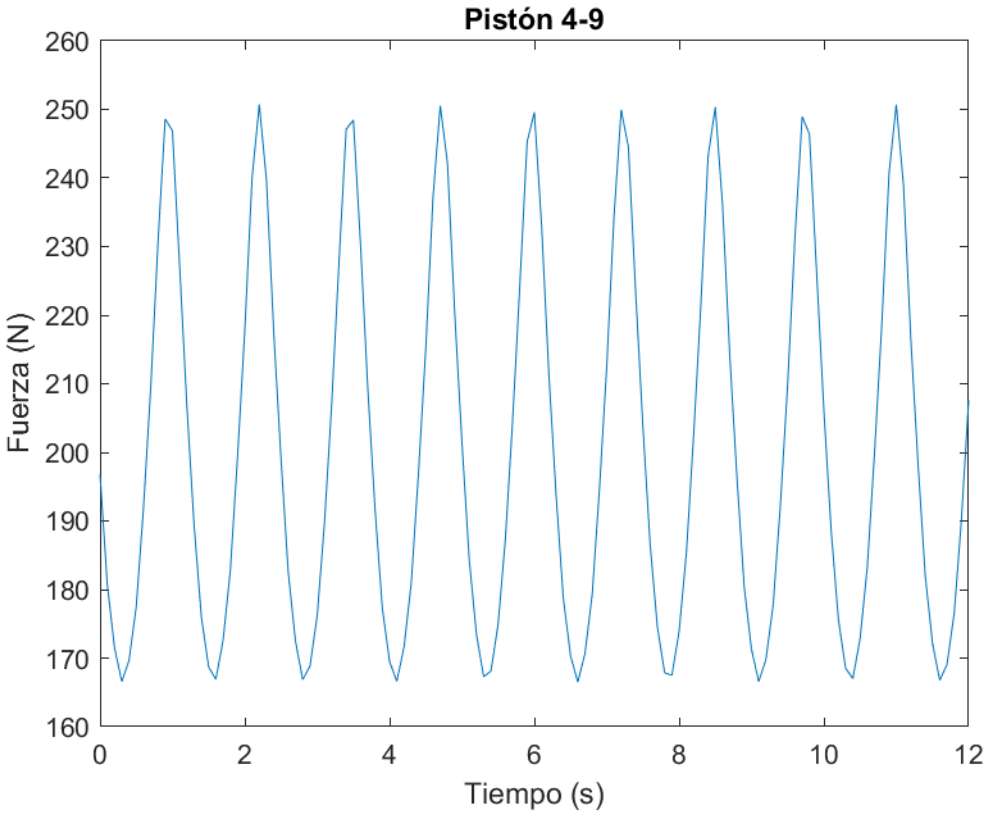


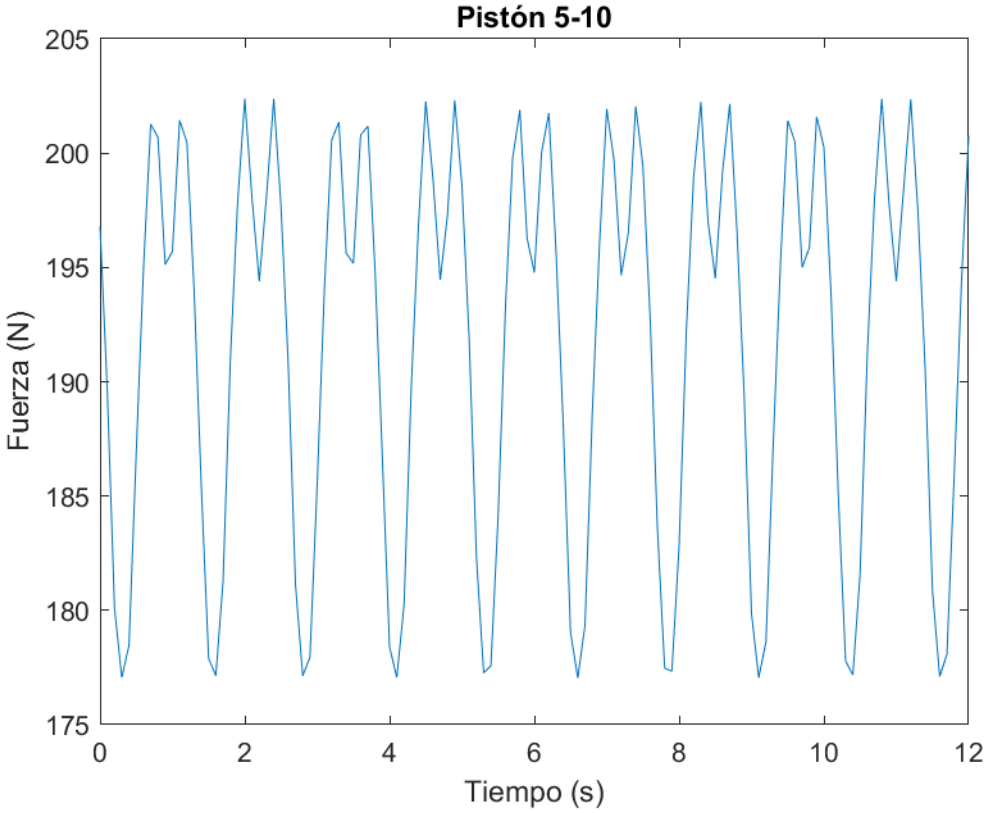
Imagen 3.13: Resultados pistones en guiñada.

Se puede apreciar con facilidad como los pistones, dispuestos de dos en dos, ejercen fuerzas opuestas para el mismo instante de tiempo.


Pistón 2-13
Fuerza (N)
Tiempo (s)
200
195
190
185
180
175
170
165
160
0
2
4
6
8
10
12


Pistón 3-8
Fuerza (N)
Tiempo (s)
215
210
205
200
195
190
185
180
175
170
165
0
2
4
6
8
10
12


Pistón 4-9
Fuerza (N)
Tiempo (s)
260
250
240
230
220
210
200
190
180
170
160
0
2
4
6
8
10
12


Pistón 5-10
Fuerza (N)
Tiempo (s)
205
200
195
190
185
180
175
0
2
4
6
8
10
12

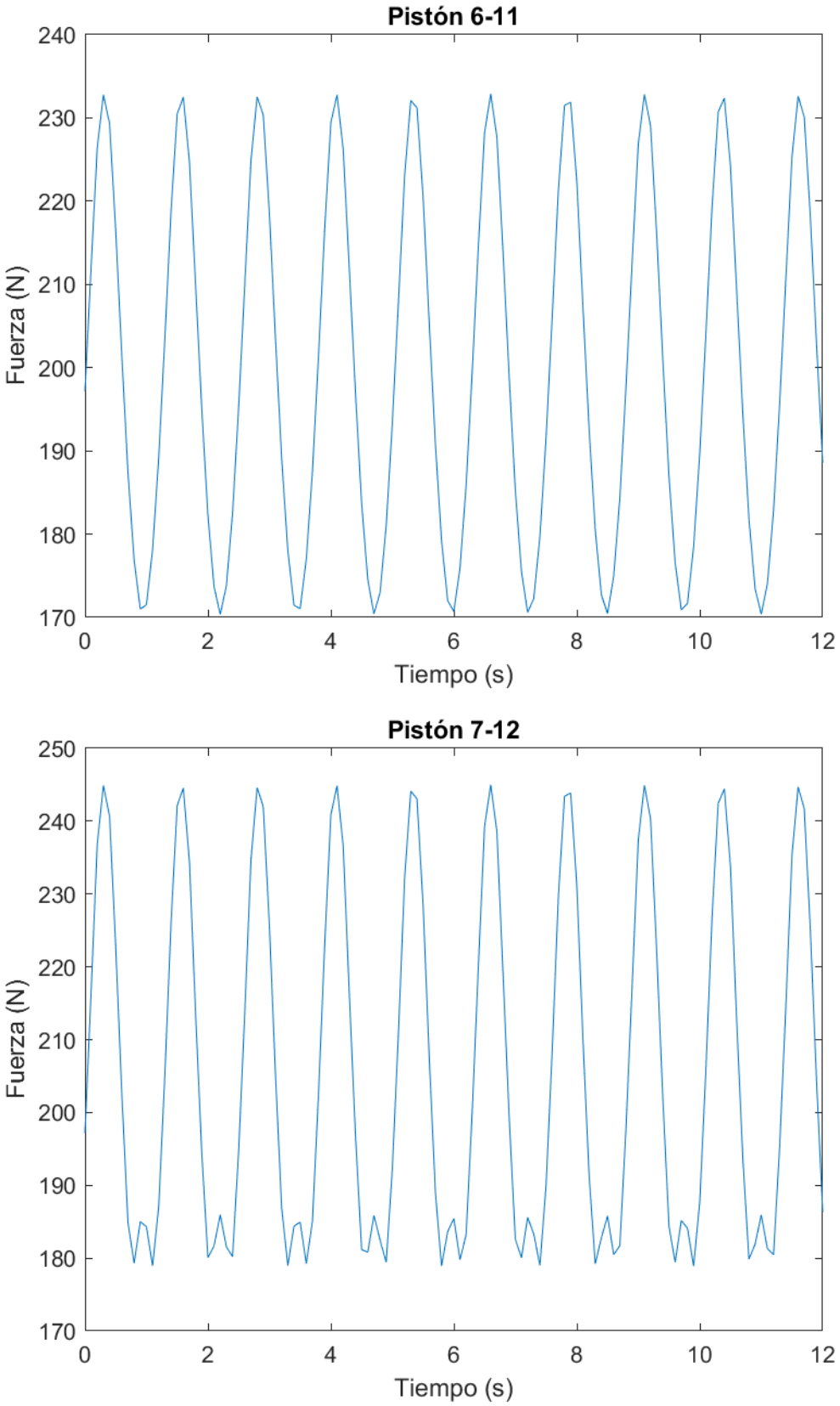
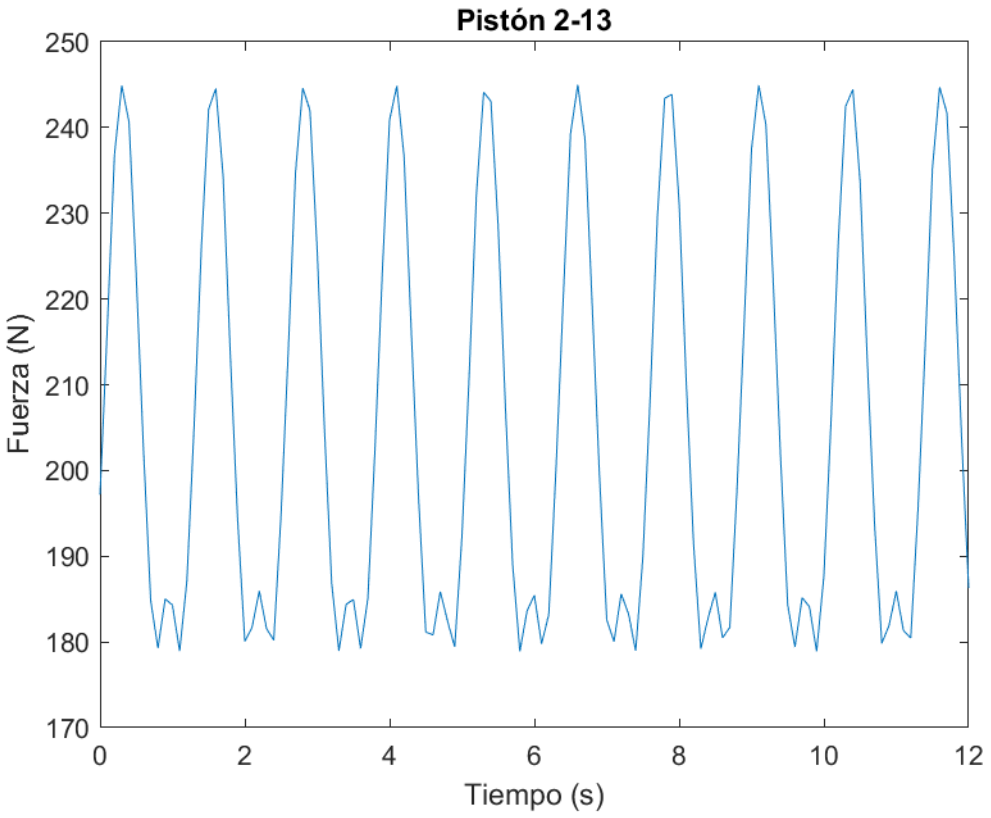
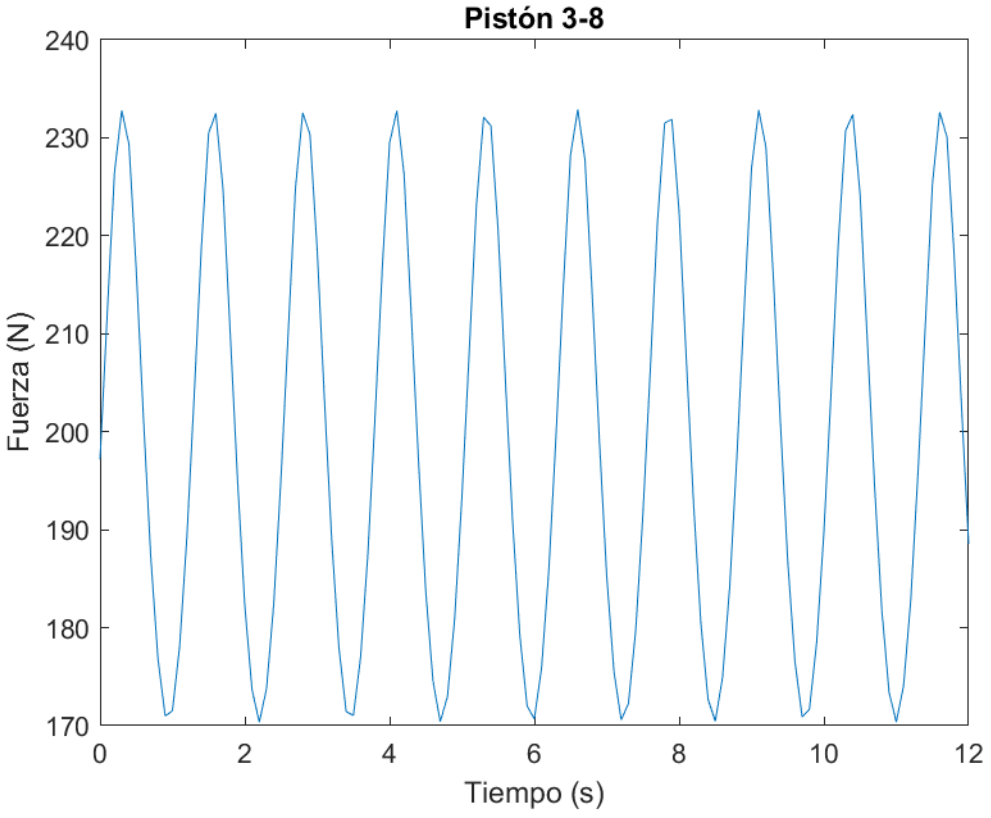
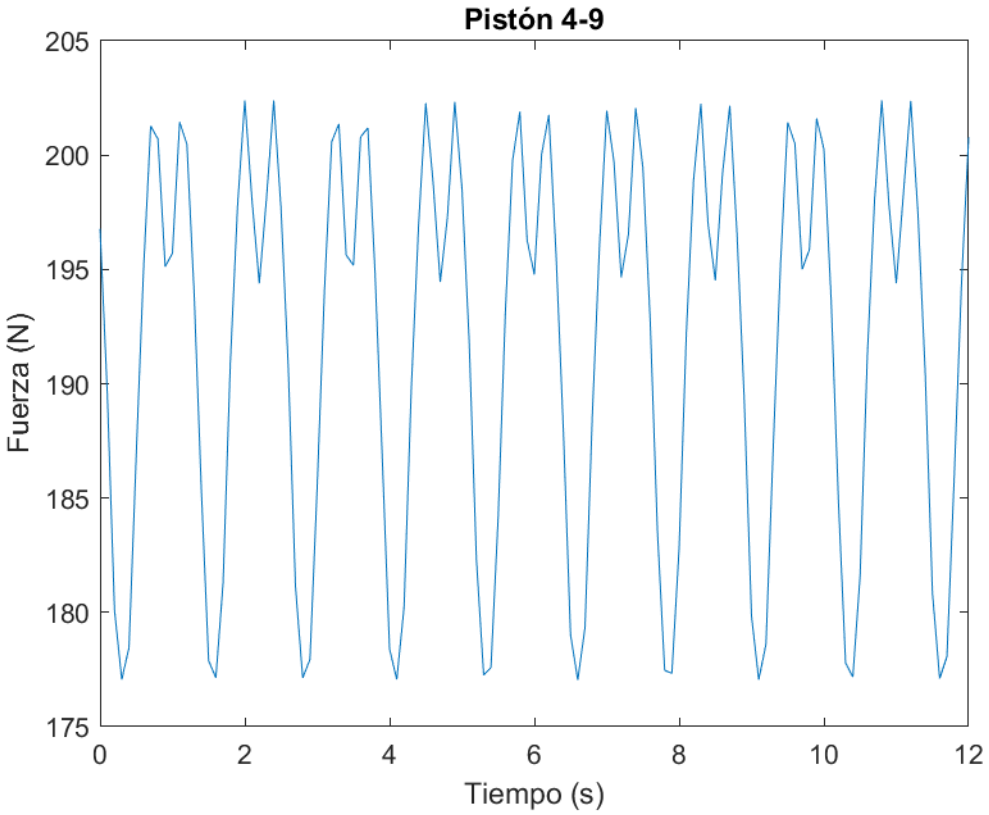


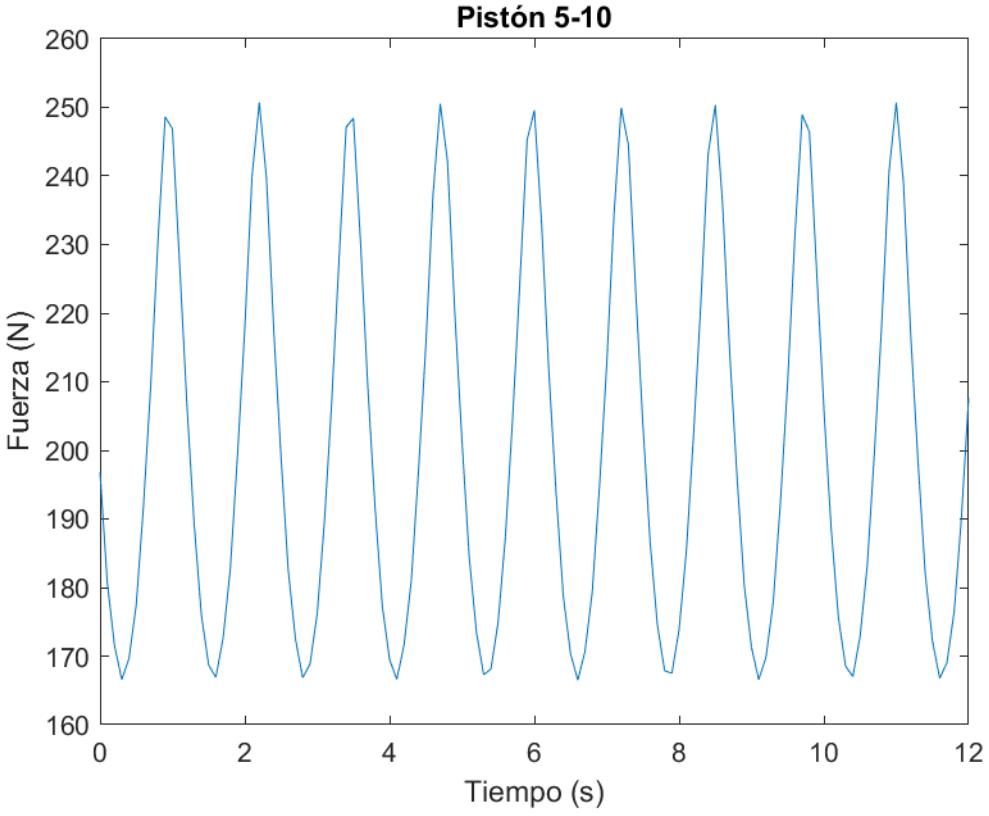
Imagen 3.14: Resultados fuerzas pistones en alabeo.

Por último, para el caso de cabeceo:






Pistón 4-9
Fuerza (N)
Tiempo (s)
205
200
195
190
185
180
175
0
2
4
6
8
10
12


Pistón 5-10
Fuerza (N)
Tiempo (s)
260
250
240
230
220
210
200
190
180
170
160
0
2
4
6
8
10
12

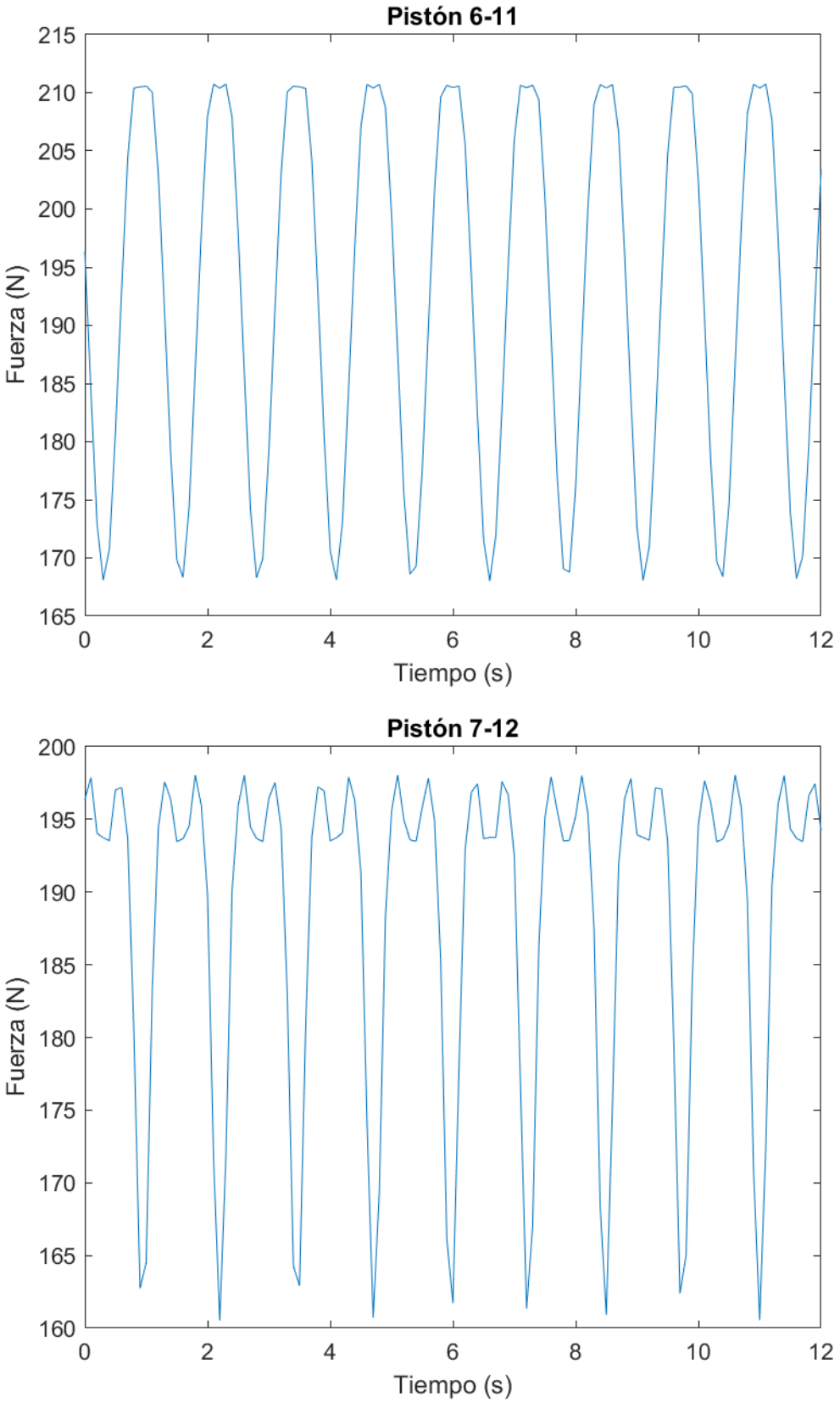


Imagen 3.15: Resultados fuerzas de los pistones en cabeceo.

4.4. Conclusiones.

Referente a la plataforma de Stewart: se han podido corroborar algunas de las características que al principio del presente documento se exponían:

- Gran utilidad a la hora de simular movimientos de 6 grados de libertad, permitiendo un gran rango de desplazamientos. En particular, se ha demostrado la facilidad de dicho mecanismo para representar el movimiento de aeronaves a partir de los tres movimientos básicos de alabeo, cabeceo y guiñada.
- Permite trabajar a altas velocidades. Por ejemplo, se han simulado movimientos de alabeo, guiñada y cabeceo a altas frecuencias y el mecanismo a funcionado correctamente.
- La relación carga/peso efectivamente es alta. Los seis pistones permiten repartirse la carga de manera que las fuerzas ejercidas no son excesivamente altas para las cargas que está soportando la plataforma.
- En definitiva, la gran utilidad de este mecanismo a la hora de aplicarlo a simuladores.
- A pesar del mayor coste computacional y la complejidad del mecanismo, las grandes ventajas que este proporciona hacen de este mal un mal menor.

Referente a la dinámica de sistemas multicuerpos: se ha puesto de manifiesto en este trabajo el beneficio de usar el análisis de sistemas multicuerpos asistidos por ordenador, ya que ha demostrado ser una técnica que puede ser utilizada de manera general y sistemática para cualquier mecanismo. Además, permite evaluar previamente mecanismos antes de que estos se fabriquen, para estudiar su viabilidad, lo que se traduce en una mayor eficiencia y reducción de costes.

En definitiva, todos estos conocimientos adquiridos al profundizar en dicha disciplina han servido al autor de este trabajo para tener una visión más amplia del mundo de la teoría de máquinas.

4.5. Mejoras futuras.

Puesto que esto es solo el principio de algo que podría llevar a un estudio mucho más exhaustivo, las posibles mejoras futuras son numerosas:

- Resolución del problema dinámico directo.
- Representación de movimientos curiosos y más complejos de aeronaves, combinando los tres movimientos básicos de alabeo, guiñada y cabeceo.
- Estudio de singularidades del mecanismo
- Aplicación de control automático a la plataforma de Stewart.

4 CÓDIGOS.

4.1. Códigos mecanismo Biela-Manivela.

4.1.1 Estructura

```
%% Programa que monta la estructura de la Biela-Manivela.

MBody.nombre = "BielaManivela";

%Datos enteros básicos del mecanismo
MBody.nb = 4; %Número de sólidos
MBody.nr = 3; %Número de pares de revolución
MBody.np = 1; %Número de pares prismáticos

%Vector de estructuras de sólidos
MBody.Solidos(1).nombre = "Fija";
MBody.Solidos(2).nombre = "Manivela";
MBody.Solidos(3).nombre = "Biela";
MBody.Solidos(4).nombre = "Corredera";

%Vector de estructuras de pares de revolución
MBody.Pares.Rev(1).I = 1;
MBody.Pares.Rev(1).J = 2;
MBody.Pares.Rev(1).vi = [0 0]';
MBody.Pares.Rev(1).vj = [0 0]';
MBody.Pares.Rev(2).I = 2;
MBody.Pares.Rev(2).J = 3;
MBody.Pares.Rev(2).vi = [1 0]';
MBody.Pares.Rev(2).vj = [0 0]';
MBody.Pares.Rev(3).I = 3;
MBody.Pares.Rev(3).J = 4;
MBody.Pares.Rev(3).vi = [2 0]';
MBody.Pares.Rev(3).vj = [0 0]';

%Vector de estructuras de pares prismáticos
MBody.Pares.Prism(1).I = 1;
MBody.Pares.Prism(1).J = 4;
MBody.Pares.Prism(1).vi = [0 0]';
MBody.Pares.Prism(1).vj = [0 0]';
MBody.Pares.Prism(1).hi = [0 1]';
MBody.Pares.Prism(1).bet = 0.0;
```

4.1.2 Restricciones.

```
%% Función que monta la matriz de ecuaciones de restricción.

function C = RestriccionesParesCinematicos(q)
global MBody
global tiempo
```

```
%% Inicializa vector de restricciones
nrestr = 3 + 2*(MBody.nr + MBody.np);
C = zeros(nrestr,1);
%% Barra fija
C(1,1) = q(1);
C(2,1) = q(2);
C(3,1) = q(3);

%% Pares de revolución
for k = 1:MBody.nr;
i = MBody.Pares.Rev(k).I;
j = MBody.Pares.Rev(k).J;
qi = q(3*(i-1)+1:3*(i-1)+3);
qj = q(3*(j-1)+1:3*(j-1)+3);
Cr = RestriccionParRevolucion(k,qi,qj);
C(3+2*(k-1)+1,1) = Cr(1);
C(3+2*(k-1)+2,1) = Cr(2);
end

%% Pares prismáticos
for k=1:MBody.np;
i = MBody.Pares.Prism(k).I;
j = MBody.Pares.Prism(k).J;
qi = q(3*(i-1)+1:3*(i-1)+3);
qj = q(3*(j-1)+1:3*(j-1)+3);
Cp = RestriccionParPrismatico(k,qi,qj);
C(3+2*MBody.nr+2*(k-1)+1,1) = Cp(1);
C(3+2*MBody.nr+2*(k-1)+2,1) = Cp(2);
end

%% Restricción de movilidad. Hay que modificar en el caso de
otro mecanismo, ya que las restricciónes serán diferentes.
C(3+2*(MBody.nr+MBody.np)+1,1)=q(6)-MBody.w*tiempo;

end

%% Función que monta las ecuaciones del par de revolución que
une el sólido i con el j

function Cr = RestriccionParRevolucion(k,qi,qj)
global MBody

ui = MBody.Pares.Rev(k).vi;
uj = MBody.Pares.Rev(k).vj;
Ai = RotMat(qi(3));
Aj = RotMat(qj(3));
Cr = qi(1:2)+Ai*ui-qj(1:2)-Aj*uj;

end

%% Función que monta las ecuaciones del par de prismático que
une el sólido i con el j
```

```
function Cp = RestriccionParPrismatico(k,qi,qj)
global MBody
ui = MBody.Pares.Prism(k).vi;
uj = MBody.Pares.Prism(k).vj;
hi = MBody.Pares.Prism(k).hi;
bet = MBody.Pares.Prism(k).bet;
Ai = RotMat(qi(3));
Aj = RotMat(qj(3));
Cp(1) = qi(3)-qj(3)-bet;
Cp(2) = (Ai*hi)'*(qi(1:2,1)+Ai*ui-qj(1:2,1)-Aj*uj);
end
```

4.1.3 Jacobiano, matriz DCq y derivadas con respecto al tiempo de las restricciones.

```
%% Función que monta la matriz jacobiana de las restricciones.
Cada fila corresponde a una ecuación de restricción, y cada
columna a la variable con respecto a la cual se está derivando.
Se introducen las submatrices en bloques, primero la del sólido
i, y luego la del j.

function Cq = Jacobiano(q,nrestr)
global MBody

%% Inicializa jacobiano con dimensión (nrest+ngl)x(nrest+ngl)
porque añadimos a las ecuaciones de restricción tantas
ecuaciones de movilidad como gdl tenga el mecanismo.
ngl=1;
Cq=zeros(nrestr+ngl);

%% Barra fija
C=eye(3);
Cq(1:3,1:3)=C;

%% Pares de revolución
for k = 1:MBody.nr;
i = MBody.Pares.Rev(k).I;
j = MBody.Pares.Rev(k).J;
ui = MBody.Pares.Rev(k).vi;
uj = MBody.Pares.Rev(k).vj;
qi = q(3*(i-1)+1:3*(i-1)+3,1);
qj = q(3*(j-1)+1:3*(j-1)+3,1);
Cq(2*k+2:2*k+3 , 3*i-2:3*i)=[ 1 0 -ui(1)*sin(qi(3))-
ui(2)*cos(qi(3)); 0 1 ui(1)*cos(qi(3))-ui(2)*sin(qi(3))];
Cq(2*k+2:2*k+3 , 3*j-2:3*j)=[-1 0
uj(1)*sin(qj(3))+uj(2)*cos(qj(3)); 0 -1 -
uj(1)*cos(qj(3))+uj(2)*sin(qj(3))];
end

%% Pares prismáticos
for k=1:MBody.np;
i = MBody.Pares.Prism(k).I;
j = MBody.Pares.Prism(k).J;
```

```
ui = MBody.Pares.Prism(k).vi;
uj = MBody.Pares.Prism(k).vj;
hi = MBody.Pares.Prism(k).hi;
bet = MBody.Pares.Prism(k).bet;
qi = q(3*(i-1)+1:3*(i-1)+3,1);
qj = q(3*(j-1)+1:3*(j-1)+3,1);
Ai = RotMat(qi(3));
Aj = RotMat(qj(3));
Athetai= RotMatTheta(qi(3));
Athetaj= RotMatTheta(qj(3));


Cq(3+2*MBody.nr+(2*k-1):3+2*MBody.nr+2*k , 3*i-2:3*i)=[0 0
1;(Ai*hi)' (Athetai*hi)'*(qi(1:2)+Ai*ui-qj(1:2)-
Aj*uj)+(Ai*hi)'*(Athetai*ui)];
Cq(3+2*MBody.nr+(2*k-1):3+2*MBody.nr+2*k , 3*j-2:3*j)=[0 0 -1;-
(Ai*hi)' (Ai*hi)'*(-Athetaj*uj)];
end

%% Ecuación de movilidad. Nos queda meter en el jacobiano la
restricción de movilidad que hayamos establecido.

Cq(3+2*MBody.nr+2*MBody.np+1,6)=1;

end

%% Derivada de las ecuaciones de restricción con respecto al
tiempo.

function Ct = dtRestr(q,tspan);
global MBody
ngl=1;
nrestr = 3 + 2*(MBody.nr + MBody.np);
Ct=zeros(nrestr+ngl,1);
Ct(end,1)=-MBody.w;
End


%% Derivada del jacobiano respecto a las coordenadas. No es más
que derivar los jacobianos en cada columna por su variable
correspondiente.

function DCq = DtJacobiano(q,v)
global MBody
%% Inicializa derivada del jacobiano con dimensión
(nrest+ngl)x(nrest+ngl) porque añadimos a las ecuaciones de
restricción tantas ecuaciones de movilidad como gdl tenga el
mecanismo.
ngl=1; nrestr = 3 + 2*(MBody.nr + MBody.np);
DCq=zeros(nrestr+ngl);

%% Barra fija
C=zeros(3);
```

```
DCq(1:3,1:3)=C;

%% Pares de revolución
for k = 1:MBody.nr;
i = MBody.Pares.Rev(k).I;
j = MBody.Pares.Rev(k).J;
ui = MBody.Pares.Rev(k).vi;
uj = MBody.Pares.Rev(k).vj;
qi = q(3*(i-1)+1:3*(i-1)+3,1);
qj = q(3*(j-1)+1:3*(j-1)+3,1);
DCq(2*k+2:2*k+3 , 3*i-2:3*i)=[0 0 -
ui(1)*cos(qi(3))+ui(2)*sin(qi(3)); 0 0 -ui(1)*sin(qi(3))-
ui(2)*cos(qi(3))];
DCq(2*k+2:2*k+3 , 3*j-2:3*j)=[0 0 uj(1)*cos(qj(3))-
uj(2)*sin(qj(3)); 0 0 uj(1)*sin(qj(3))+uj(2)*cos(qj(3))];
end

%% Pares prismáticos
for k=1:MBody.np;
i = MBody.Pares.Prism(k).I;
j = MBody.Pares.Prism(k).J;
ui = MBody.Pares.Prism(k).vi;
uj = MBody.Pares.Prism(k).vj;
hi = MBody.Pares.Prism(k).hi;
bet = MBody.Pares.Prism(k).bet;
qi = q(3*(i-1)+1:3*(i-1)+3,1);
qj = q(3*(j-1)+1:3*(j-1)+3,1);
Ai = RotMat(qi(3));
Aj = RotMat(qj(3));
Athetai= RotMatTheta(qi(3));
Athetaj= RotMatTheta(qj(3));
Atheta2i= RotMatTheta2(qi(3));
Atheta2j= RotMatTheta2(qj(3));
DCq(3+2*MBody.nr+(2*k-1):3+2*MBody.nr+2*k , 3*i-2:3*i)=[0 0 0;0
0 (Atheta2i*hi)'*(qi(1:2)+Ai*ui-qj(1:2)-
Aj*uj)+(Athetai*hi)'*(Athetai*ui)+(Athetai*hi)'*(Athetai*ui)+(Ai
*hi)'*(Atheta2i*ui)];
DCq(3+2*MBody.nr+(2*k-1):3+2*MBody.nr+2*k , 3*j-2:3*j)=[0 0 0;0
0 (Ai*hi)'*(-Atheta2j*uj)];
end

%% Ecuación de movilidad. Nos queda meter en el jacobiano la
restricción de movilidad que hayamos establecido.

DCq(3+2*(MBody.nr+MBody.np)+1,6)=0;

End

%% Derivada segunda de las restricciones respecto al tiempo. En
este caso, todo es 0.

function DCt = DtdtRestr(q,v,tspan)
global MBody
```

```
ngl=1;
nrestr = 3 + 2*(MBody.nr + MBody.np);
DCt=zeros(nrestr+ngl,1);
End
```

4.1.4 Programa principal. Simulación cinemática.

```
%% Programa principal. En el se resuelven las ecuaciones de
restricción para resolver el problema de posición, y
posteriormente se calculan las velocidades y aceleraciones.

close all
clear all
clc
global MBody
global tiempo
BielaManivela;
%%%%%%%%%%%%%%%%%%%%%%%%
% SIMULACIÓN CINEMÁTICA %
%%%%%%%%%%%%%%%%%%%%%%%%

MBody.w = 10*(2*pi/60); %VELOCIDAD ANGULAR (10 rpm)
tspan = 0:0.1:(4*pi/MBody.w); %TIEMPO DE INTEGRACIÓN (2 vueltas)
%Estimacion inicial
q0 = [0 0 0 0 0 0 2 0 0 3 0 0]';
nrestr = 3 + 2*(MBody.nr + MBody.np);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BUCLE DE SOLUCIÓN DEL PROBLEMA DE POSICIÓN %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:length(tspan),
%PROBLEMA DE POSICION
tiempo=tspan(i);
q(:,i) =
fsolve(@RestriccionesParesCinematicos,q0,optimset('Display','off
'));
Cq = Jacobiano(q(:,i),nrestr);
Ct = dtRestr(q(:,i),tspan(i));
%CALCULO DE VELOCIDADES
v(:,i) = -Cq\Ct;
DCq = DtJacobiano(q(:,i),v(:,i));
DCt = DtdtRestr(q(:,i),v(:,i),tspan(i));
%CALCULO DE ACELERACIONES
a(:,i) = -Cq\(DCq*v(:,i).^2+DCt);
% a(:,i) = -Cq\(DCq*v(:,i)+DCt);
q0 = q(:,i);
end

figure(1)
plot(tspan,q(10,:))
```

```
hold on
plot(tspan,q(11,:))
plot(tspan,q(12,:))
xlabel('Tiempo (s)')
ylabel('x, y, \theta')
title('Coordenadas de la corredera.')
legend('x(m)','y(m)','\theta(rad)')
filename=['Pos_Corredera.png'];
saveas([1],filename)

figure(2)
plot(tspan,v(10,:))
hold on
plot(tspan,v(11,:))
plot(tspan,v(12,:))
xlabel('Tiempo (s)')
ylabel('V_x, V_y,\omega_\theta')
title('Velocidades de la corredera.')
legend('V_x (m/s)','V_y(m/s)','\omega_\theta(rad/s)')
filename=['Vel_Corredera.png'];
saveas([2],filename)

figure(3)
plot(tspan,a(10,:))
hold on
plot(tspan,a(11,:))
plot(tspan,a(12,:))
xlabel('Tiempo (s)')
ylabel('V_x, V_y,\omega_\theta')
title('Aceleraciones de la corredera.')
legend('a_x (m/s)','a_y(m/s)','\omega_\theta(rad/s^2)')
filename=['Ac_Corredera.png'];
saveas([2],filename)

figure(4)
plot(tspan,v(10,:))
hold on
plot(tspan,a(10,:))
xlabel('Tiempo (s)')
ylabel('x, y, \theta')
title('Velocidad VS Aceleración corredera.')
legend('V_x(m/s)','a_x(m/s^2)')
filename=['Vel_Ac_Corredera.png'];
saveas([4],filename)

% figure(4)
% anima


%% Animador del mecanismo. Realiza una representación en el
plano de cómo se va moviendo el mecanismo.
```

```
% debe estar como en el programa principal
%global L1 L2 L3 L4

%animacion
nc=1; %numero de ciclos que quiero visualizar
np=length(tspan);
%***********MODIFICAR*************
%coordenadas de los puntos a pintar en la animacion
%O2 y A pueden permanecer como estan
L2=1;
L3=2;
xO2= zeros(1,np);
yO2= zeros(1,np);
xA = L2*cos(q(6,:));
yA = L2*sin(q(6,:));
xB = xA + L3*cos(q(9,:));
yB = yA + L3*sin(q(9,:));
xO4= q(10,:);
yO4= q(11,:);


%definicion de las dos lineas a dibujar
x1 = [xO2; xA; xB; xO4];
y1 = [yO2; yA; yB; yO4];

%************************************

for j=1:nc,
for i=2:(np),
   plot(x1(:,i),y1(:,i))



   %para animar necesito fijar los ejes, eliminarlos y
ralentizar la animacion
   %correr una vez y modificar luego para que se ajuste la
pantalla al
   %dibujo
   axis([-3 3 -3 3])

   %poner el mismo rango para que no se distorsione figura
(circulo y no elipse)
   axis square off
   pause(.1)
end
end
```

4.1.5 Matrices de rotación.

```
function A = RotMat(x)
A(1,1) = cos(x);
A(1,2) = -sin(x);
A(2,1) = sin(x);
A(2,2) = cos(x);
end

function A=RotMatTheta(x)
A(1,1) = -sin(x);
A(1,2) = -cos(x);
A(2,1) = cos(x);
A(2,2) = -sin(x);
end

function A=RotMatTheta2(x)
A(1,1) = -cos(x);
A(1,2) =  sin(x);
A(2,1) = -sin(x);
A(2,2) = -cos(x);
end
```

4.1.6 Dinámica inversa Biela- Manivela.

```
%% Dinámica inversa: Calcular el par motor de la manivela.
close all
global MBody
ngl=1;
nrestr=3+2*(MBody.nr+MBody.np);

%% Se monta la matriz de masa y la matriz de acciones
exteriores.
M2=1;
M3=2;
M4=3;
g=9.81; %Aceleración de la gravedad.
g0=[0 0 0 0 -M2*g 0 0 -M3*g 0 0 -M4*g 0];
% g0=[0 0 0 0 0 0 0 0 0 2 0 0];

lambda=zeros(nrestr+ngl,length(tspan));
lambda1=zeros(nrestr+ngl,length(tspan));
b=0.5; c=0.2; %Dimensiones del rectángulo de la corredera
g1=zeros(nrestr+ngl,length(tspan));
M=zeros(12);
M(1:3,1:3)=eye(3);
M(4:6,4:6)=M2*eye(3)+L2^2/12*[0 0 0; 0 0 0; 0 0 1];
M(7:9,7:9)=M3*eye(3)+L3^2/12*[0 0 0; 0 0 0; 0 0 1];
M(10:12,10:12)= [M4 0 0;0 M4 0; 0 0 8*M4*(b^2+c^2)];
for i = 1:length(tspan),
%PROBLEMA DE POSICION
tiempo=tspan(i);
Cq = Jacobiano(q(:,i),nrestr);
```

```
% Se calcula Lambda. Vector de reacciones.
lambda(:,i)=Cq'\(M*a(:,i)-g0');

%Una vez se tiene lambda, se procede a calcular el par motor de
la manivela
%Para ello, se hace cero todo el vector lambda excepto lambda12

lambda1(12,i)=lambda(12,i);
g1(:,i)=Cq'*lambda1(:,i); %Este vector me da las reacciones en
la componente sexta, que es la que da el par motor.
end

figure
plot(tspan,g1(6,:))
% hold on
% plot(tspan,-2*v(10,:)./v(6,:))
title('Par Motor manivela')
xlabel('Tiempo (s)')
ylabel('Par Motor')
filename=['Par_Motor.png'];
saveas([1],filename)
```

4.2. Códigos de la plataforma de Stewart.

4.2.1 Estructura.

```
%% Montaje de la estructura del mecanismo.

MBody.nombre = "StewartPlatform";

%% Datos enteros básicos del mecanismo
MBody.nb = 14; %Número de sólidos
MBody.ncar = 12; %Número de pares cardan
MBody.ncil = 6; %Número de pares cilíndricos

%% Vector de estructuras de sólidos
MBody.Solidos(1).nombre = "Plataforma fija";
MBody.Solidos(2).nombre = "Barra 1";
MBody.Solidos(3).nombre = "Barra 2";
MBody.Solidos(4).nombre = "Barra 3";
MBody.Solidos(5).nombre = "Barra 4";
MBody.Solidos(6).nombre = "Barra 5";
MBody.Solidos(7).nombre = "Barra 6";
MBody.Solidos(8).nombre = "Barra 7";
MBody.Solidos(9).nombre = "Barra 8";
MBody.Solidos(10).nombre = "Barra 9";
MBody.Solidos(11).nombre = "Barra 10";
MBody.Solidos(12).nombre = "Barra 11";
MBody.Solidos(13).nombre = "Barra 12";
MBody.Solidos(14).nombre = "Plataforma móvil";
```

```
%% Vector de estructuras de pares cardan
    %Pares cardan de la plataforma fija
MBody.Pares.Car(1).I = 1;
MBody.Pares.Car(1).J = 2;
MBody.Pares.Car(1).uPi = [50 0 0]';
MBody.Pares.Car(1).uPj = [0 0 0]';
MBody.Pares.Car(1).si = [0 1 0]';
MBody.Pares.Car(1).sj = [1 0 0]';
MBody.Pares.Car(2).I = 1;
MBody.Pares.Car(2).J = 3;
MBody.Pares.Car(2).uPi = [50*cos(pi/6) 50*sin(pi/6) 0 ]';
MBody.Pares.Car(2).uPj = [0 0 0]';
MBody.Pares.Car(2).si = [-sin(pi/6) cos(pi/6) 0]';
MBody.Pares.Car(2).sj = [1 0 0]';
MBody.Pares.Car(3).I = 1;
MBody.Pares.Car(3).J = 4;
MBody.Pares.Car(3).uPi = [50*cos(2*pi/3) 50*sin(2*pi/3) 0 ]';
MBody.Pares.Car(3).uPj = [0 0 0]';
MBody.Pares.Car(3).si = [-sin(2*pi/3) cos(2*pi/3) 0]';
MBody.Pares.Car(3).sj = [1 0 0]';
MBody.Pares.Car(4).I = 1;
MBody.Pares.Car(4).J = 5;
MBody.Pares.Car(4).uPi = [50*cos(5*pi/6) 50*sin(5*pi/6) 0 ]';
MBody.Pares.Car(4).uPj = [0 0 0]';
MBody.Pares.Car(4).si = [-sin(5*pi/6) cos(5*pi/6) 0]';
MBody.Pares.Car(4).sj = [1 0 0]';
MBody.Pares.Car(5).I = 1;
MBody.Pares.Car(5).J = 6;
MBody.Pares.Car(5).uPi = [50*cos(4*pi/3) 50*sin(4*pi/3) 0 ]';
MBody.Pares.Car(5).uPj = [0 0 0]';
MBody.Pares.Car(5).si = [-sin(4*pi/3) cos(4*pi/3) 0]';
MBody.Pares.Car(5).sj = [1 0 0]';
MBody.Pares.Car(6).I = 1;
MBody.Pares.Car(6).J = 7;
MBody.Pares.Car(6).uPi = [50*cos(3*pi/2) 50*sin(3*pi/2) 0 ]';
MBody.Pares.Car(6).uPj = [0 0 0]';
MBody.Pares.Car(6).si = [-sin(3*pi/2) cos(3*pi/2) 0]';
MBody.Pares.Car(6).sj = [1 0 0]';

%Pares Cardan de la plataforma móvil. Girada pi/3.
MBody.Pares.Car(7).I = 14;
MBody.Pares.Car(7).J = 8;
MBody.Pares.Car(7).uPi = [50*cos(pi/3) 50*sin(pi/3) 0]';
MBody.Pares.Car(7).uPj = [0 0 0]';
MBody.Pares.Car(7).si = [-sin(pi/3) cos(pi/3) 0]';
MBody.Pares.Car(7).sj = [1 0 0]';
MBody.Pares.Car(8).I = 14;
MBody.Pares.Car(8).J = 9;
MBody.Pares.Car(8).uPi = [50*cos(pi/6+pi/3) 50*sin(pi/6+pi/3)
0]';
MBody.Pares.Car(8).uPj = [0 0 0]';
MBody.Pares.Car(8).si = [-sin(pi/6+pi/3) cos(pi/6+pi/3) 0]';
```

```
MBody.Pares.Car(8).sj = [1 0 0]';
MBody.Pares.Car(9).I = 14;
MBody.Pares.Car(9).J = 10;
MBody.Pares.Car(9).uPi = [50*cos(2*pi/3+pi/3)
50*sin(2*pi/3+pi/3) 0 ]';
MBody.Pares.Car(9).uPj = [0 0 0]';
MBody.Pares.Car(9).si = [-sin(2*pi/3+pi/3) cos(2*pi/3+pi/3) 0]';
MBody.Pares.Car(9).sj = [1 0 0]';
MBody.Pares.Car(10).I = 14;
MBody.Pares.Car(10).J = 11;
MBody.Pares.Car(10).uPi = [50*cos(5*pi/6+pi/3)
50*sin(5*pi/6+pi/3) 0 ]';
MBody.Pares.Car(10).uPj = [0 0 0]';
MBody.Pares.Car(10).si = [-sin(5*pi/6+pi/3) cos(5*pi/6+pi/3)
0]';
MBody.Pares.Car(10).sj = [1 0 0]';
MBody.Pares.Car(11).I = 14;
MBody.Pares.Car(11).J = 12;
MBody.Pares.Car(11).uPi = [50*cos(4*pi/3+pi/3)
50*sin(4*pi/3+pi/3) 0]';
MBody.Pares.Car(11).uPj = [0 0 0]';
MBody.Pares.Car(11).si = [-sin(4*pi/3+pi/3) cos(4*pi/3+pi/3)
0]';
MBody.Pares.Car(11).sj = [1 0 0]';
MBody.Pares.Car(12).I = 14;
MBody.Pares.Car(12).J = 13;
MBody.Pares.Car(12).uPi = [50*cos(3*pi/2+pi/3)
50*sin(3*pi/2+pi/3) 0 ]';
MBody.Pares.Car(12).uPj = [0 0 0]';
MBody.Pares.Car(12).si = [-sin(3*pi/2+pi/3) cos(3*pi/2+pi/3)
0]';
MBody.Pares.Car(12).sj = [1 0 0]';


%% Vector de estructuras de pares cilíndricos.
MBody.Pares.Cil(1).I = 3;
MBody.Pares.Cil(1).J = 8;
MBody.Pares.Cil(1).vi = [0 0 1]';
MBody.Pares.Cil(1).vj = [0 0 1]';
MBody.Pares.Cil(1).ui = [0 0 0]';
MBody.Pares.Cil(1).uj = [0 0 0]';
MBody.Pares.Cil(2).I = 4;
MBody.Pares.Cil(2).J = 9;
MBody.Pares.Cil(2).vi = [0 0 1]';
MBody.Pares.Cil(2).vj = [0 0 1]';
MBody.Pares.Cil(2).ui = [0 0 0]';
MBody.Pares.Cil(2).uj = [0 0 0]';
MBody.Pares.Cil(3).I = 5;
MBody.Pares.Cil(3).J = 10;
MBody.Pares.Cil(3).vi = [0 0 1]';
MBody.Pares.Cil(3).vj = [0 0 1]';
MBody.Pares.Cil(3).ui = [0 0 0]';
```

```
MBody.Pares.Cil(3).uj = [0 0 0]';
MBody.Pares.Cil(4).I = 6;
MBody.Pares.Cil(4).J = 11;
MBody.Pares.Cil(4).vi = [0 0 1]';
MBody.Pares.Cil(4).vj = [0 0 1]';
MBody.Pares.Cil(4).ui = [0 0 0]';
MBody.Pares.Cil(4).uj = [0 0 0]';
MBody.Pares.Cil(5).I = 7;
MBody.Pares.Cil(5).J = 12;
MBody.Pares.Cil(5).vi = [0 0 1]';
MBody.Pares.Cil(5).vj = [0 0 1]';
MBody.Pares.Cil(5).ui = [0 0 0]';
MBody.Pares.Cil(5).uj = [0 0 0]';
MBody.Pares.Cil(6).I = 2;
MBody.Pares.Cil(6).J = 13;
MBody.Pares.Cil(6).vi = [0 0 1]';
MBody.Pares.Cil(6).vj = [0 0 1]';
MBody.Pares.Cil(6).ui = [0 0 0]';
MBody.Pares.Cil(6).uj = [0 0 0]';
```

4.2.2 Restricciones.

```
%% Función que monta la matriz de restricciones.

function C=Restricciones(q)

global MBody
global tiempo
%% Inicializa vector de restricciones.
nrestr=7+4*(MBody.ncil+MBody.ncar)+(MBody.nb-1); %La barra fija
son 7, MÁS 13 DE LA ORIENTACIÓN DE LOS 13 SOLIDOS MÓVILES.
C=zeros(nrestr,1);
ngl=6;
%% Barra fija.
for i=1:7
C(i,1)=q(i);
end
C(4,1)=q(4)-1;

%% Pares Cardan
for k=1:MBody.ncar
i = MBody.Pares.Car(k).I;
j = MBody.Pares.Car(k).J;
qi = q(7*(i-1)+1:7*(i-1)+7);
qj = q(7*(j-1)+1:7*(j-1)+7);
Ccar = RestriccionParCardan(k,qi,qj);
C(7+4*(k-1)+1,1) = Ccar(1);
C(7+4*(k-1)+2,1) = Ccar(2);
C(7+4*(k-1)+3,1) = Ccar(3);
C(7+4*(k-1)+4,1) = Ccar(4);
```

```
end

%% Pares Cilíndricos.
for k=1:MBody.ncil
i = MBody.Pares.Cil(k).I;
j = MBody.Pares.Cil(k).J;
qi = q(7*(i-1)+1:7*(i-1)+7);
qj = q(7*(j-1)+1:7*(j-1)+7);
Ccil = RestriccionParCilindrico(k,qi,qj);
C(7+4*MBody.ncar+4*(k-1)+1,1) = Ccil(1);
C(7+4*MBody.ncar+4*(k-1)+2,1) = Ccil(2);
C(7+4*MBody.ncar+4*(k-1)+3,1) = Ccil(3);
C(7+4*MBody.ncar+4*(k-1)+4,1) = Ccil(4);
end

%% Restricciones de orientación.
for k=2:MBody.nb-1
C(7+4*(MBody.ncar+MBody.ncil)+(k-1),1)=q(7*k-3:7*k)'*q(7*k-
3:7*k)-1; %No entra ni la plataforma fija, ni la móvil.
end

%% Restricciones de movilidad. Aquí deberían ir las 7 ecuaciones
de movilidad.
% Cuatro de ellas deben ser los parametros de Euler de la
plataforma móvil,
% que que hemos eliminado la condición de módulo unitario en
esta
% plataforma.

% Comentar los que no procedan y descomentar el que se quiera.

% if Ac==1
% Prueba
C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= q(92); %Primera
ecuación
C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= q(93);  %Segunda
ecuación
C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= q(94)-50-
4*sin(5*tiempo); %Tercera ecuación
C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=  q(95)-1;
%Cuarta ecuación
C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)=  q(96); %Quinta
ecuación
C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  q(97); %Sexta
ecuación
C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  q(98); %Séptima
ecuación

% elseif Ac==2

% % Guiñada
```

```
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= q(92); %Primera
ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= q(93);
%Segunda ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= q(94)-50;
%Tercera ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=  q(95)-
cos(pi/18/2*sin(5*tiempo)); %Cuarta ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)=  q(96);
%Quinta ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  q(97); %Sexta
ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  q(98)-
sin(pi/18/2*sin(5*tiempo)); %Séptima ecuación

% elseif Ac==3

% Alabeo. Giro alrededor de eje x
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= q(92); %Primera
ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= q(93);
%Segunda ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= q(94)-50;
%Tercera ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=  q(95)-
cos(pi/18/2*sin(5*tiempo)); %Cuarta ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)=  q(96)-
sin(pi/18/2*sin(5*tiempo)); %Quinta ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  q(97); %Sexta
ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  q(98);
%Séptima ecuación

% elseif Ac==4

% Cabeceo. Giro alrededor de eje y
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= q(92); %Primera
ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= q(93);
%Segunda ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= q(94)-50;
%Tercera ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=  q(95)-
cos(pi/18/2*sin(5*tiempo)); %Cuarta ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)=  q(96);
%Quinta ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  q(97)-
sin(pi/18/2*sin(5*tiempo)); %Sexta ecuación
% C(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  q(98);
%Séptima ecuación

% end
```

```
End

function Ccil=RestriccionParCilindrico(k,qi,qj)
global MBody
vi=MBody.Pares.Cil(k).vi;
vj=MBody.Pares.Cil(k).vj;
ui=MBody.Pares.Cil(k).ui;
uj=MBody.Pares.Cil(k).uj;
Ai=RotMat(qi(4:7));
Aj=RotMat(qj(4:7));

%% Restricción de movilidad.
vec=[1 0 0]';
vi1=cross(vec,vi);
vi2=cross(vi,vi1);
Ccil(1,1)=(Ai*vi1)'*Aj*vj;
Ccil(2,1)=(Ai*vi2)'*Aj*vj;
Ccil(3,1)=(Ai*vi1)'*(qi(1:3)+Ai*ui-qj(1:3)-Aj*uj);
Ccil(4,1)=(Ai*vi2)'*(qi(1:3)+Ai*ui-qj(1:3)-Aj*uj);

%% Restricción de orientación.
% Ccil(5,1)=qi(4:7)'

End

function Ccar=RestriccionParCardan(k,qi,qj)
global MBody
uPi=MBody.Pares.Car(k).uPi;
uPj=MBody.Pares.Car(k).uPj;
si=MBody.Pares.Car(k).si;
sj=MBody.Pares.Car(k).sj;
Ai=RotMat(qi(4:7));
Aj=RotMat(qj(4:7));

%% Restricción de movilidad
Ccar(1:3,1)=qi(1:3)+Ai*uPi-qj(1:3)-Aj*uPj;
Ccar(4,1)=(Ai*si)'*(Aj*sj);

%% Restricción de orientación
% Ccar(5,1)=q(4:7)'*q(4:7)-1;

End
```

4.2.3 Jacobiano, DCq y derivadas de las restricciones respecto al tiempo.

```
function Cq = Jacobiano(q,nrestr,Ac)

global MBody
```

```
%% Inicializa jacobiano con dimensión (nrest+ngl)x(nrest+ngl)
porque añadimos a las ecuaciones de restricción tantas
ecuaciones de movilidad como gdl tenga el mecanismo.
ngl=6;
Cq=zeros(nrestr+ngl);

%% Barra fija
C=eye(7);
Cq(1:7,1:7)=C;

%% Pares Cardan

for k=1:MBody.ncar

i=MBody.Pares.Car(k).I;
j=MBody.Pares.Car(k).J;
uPi=MBody.Pares.Car(k).uPi;
uPj=MBody.Pares.Car(k).uPj;
si=MBody.Pares.Car(k).si;
sj=MBody.Pares.Car(k).sj;
qi = q(7*(i-1)+1:7*(i-1)+7,1);
qj = q(7*(j-1)+1:7*(j-1)+7,1);
Ai =RotMat(qi(4:7));
Aj =RotMat(qj(4:7));
Ae0i = RotMat0(qi(4:7));
Ae1i = RotMat1(qi(4:7));
Ae2i = RotMat2(qi(4:7));
Ae3i = RotMat3(qi(4:7));
Ae0j = RotMat0(qj(4:7));
Ae1j = RotMat1(qj(4:7));
Ae2j = RotMat2(qj(4:7));
Ae3j = RotMat3(qj(4:7));
Cq(4*(k-1)+8:4*(k-1)+8+3,7*(i-1)+1:7*(i-1)+7)=[eye(3)
Ae0i*uPi           Ae1i*uPi           Ae2i*uPi
Ae3i*uPi;
                                                zeros(1,3)
(Ae0i*si)'*(Aj*sj) (Ae1i*si)'*(Aj*sj) (Ae2i*si)'*(Aj*sj)
(Ae3i*si)'*(Aj*sj)];

Cq(4*(k-1)+8:4*(k-1)+8+3,7*(j-1)+1:7*(j-1)+7)=[-eye(3)     -
Ae0j*uPj          -Ae1j*uPj          -Ae2j*uPj          -
Ae3j*uPj;
                                                 zeros(1,3)
(Ai*si)'*(Ae0j*sj) (Ai*si)'*(Ae1j*sj) (Ai*si)'*(Ae2j*sj)
(Ai*si)'*(Ae3j*sj)];
end

%% Pares Cilíndricos.

for k=1:MBody.ncil

i=MBody.Pares.Cil(k).I;
```

```
j=MBody.Pares.Cil(k).J;
ui=MBody.Pares.Cil(k).ui;
uj=MBody.Pares.Cil(k).uj;
vi=MBody.Pares.Cil(k).vi;
vj=MBody.Pares.Cil(k).vj;
vec=[1 0 0]';
vi1=cross(vec,vi);
vi2=cross(vi,vi1);
qi = q(7*(i-1)+1:7*(i-1)+7,1);
qj = q(7*(j-1)+1:7*(j-1)+7,1);
Ai =RotMat(qi(4:7,1));
Aj =RotMat(qj(4:7,1));
Ae0i = RotMat0(qi(4:7,1));
Ae1i = RotMat1(qi(4:7,1));
Ae2i = RotMat2(qi(4:7,1));
Ae3i = RotMat3(qi(4:7,1));
Ae0j = RotMat0(qj(4:7,1));
Ae1j = RotMat1(qj(4:7,1));
Ae2j = RotMat2(qj(4:7,1));
Ae3j = RotMat3(qj(4:7,1));
Cq(4*MBody.ncar+4*(k-1)+8:4*MBody.ncar+4*(k-1)+8+3,7*(i-
1)+1:7*(i-1)+7)=[zeros(1,3), (Ae0i*vi1)'*(Aj*vj),
(Ae1i*vi1)'*(Aj*vj),
(Ae2i*vi1)'*(Aj*vj),
(Ae3i*vi1)'*(Aj*vj);

zeros(1,3), (Ae0i*vi2)'*(Aj*vj),
(Ae1i*vi2)'*(Aj*vj),
(Ae2i*vi2)'*(Aj*vj),
(Ae3i*vi2)'*(Aj*vj);

(Ai*vi1)',  ((Ae0i*vi1)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj))+((Ai*vi1)'*(Ae0i*ui)), ((Ae1i*vi1)'*(qi(1:3)+Ai*ui-
qj(1:3)-Aj*uj))+((Ai*vi1)'*(Ae1i*ui)),
((Ae2i*vi1)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj))+((Ai*vi1)'*(Ae2i*ui)), ((Ae3i*vi1)'*(qi(1:3)+Ai*ui-
qj(1:3)-Aj*uj))+((Ai*vi1)'*(Ae3i*ui));

(Ai*vi2)',  ((Ae0i*vi2)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj))+((Ai*vi2)'*(Ae0i*ui)), ((Ae1i*vi2)'*(qi(1:3)+Ai*ui-
qj(1:3)-Aj*uj))+((Ai*vi2)'*(Ae1i*ui)),
((Ae2i*vi2)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj))+((Ai*vi2)'*(Ae2i*ui)), ((Ae3i*vi2)'*(qi(1:3)+Ai*ui-
qj(1:3)-Aj*uj))+((Ai*vi2)'*(Ae3i*ui))];


Cq(4*MBody.ncar+4*(k-1)+8:4*MBody.ncar+4*(k-1)+8+3,7*(j-
1)+1:7*(j-1)+7)=[zeros(1,3) (Ai*vi1)'*(Ae0j*vj)
(Ai*vi1)'*(Ae1j*vj)  (Ai*vi1)'*(Ae2j*vj)  (Ai*vi1)'*(Ae3j*vj);

zeros(1,3) (Ai*vi2)'*(Ae0j*vj)  (Ai*vi2)'*(Ae1j*vj)
(Ai*vi2)'*(Ae2j*vj)  (Ai*vi2)'*(Ae3j*vj);
```

```
-(Ai*vi1)'  (Ai*vi1)'*(-Ae0j*uj) (Ai*vi1)'*(-Ae1j*uj)
(Ai*vi1)'*(-Ae2j*uj) (Ai*vi1)'*(-Ae3j*uj);

-(Ai*vi2)'  (Ai*vi2)'*(-Ae0j*uj) (Ai*vi2)'*(-Ae1j*uj)
(Ai*vi2)'*(-Ae2j*uj) (Ai*vi2)'*(-Ae3j*uj)];
end

%% Añadimos el jacobiano de las restricciones de orientación.
for k=2:MBody.nb-1

Cq(4*(MBody.ncar+MBody.ncil)+(k-2)+7+1,7*(k-1)+1:7*(k-
1)+7)=[zeros(1,3) 2*q(7+4+(k-2)*7:7+4+(k-2)*7+3,1)'];

end

%% Por último se añaden las ecuaciones de movilidad.

if Ac==1
% % Prueba
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,92)= 1; %Primera
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,93)= 1;  %Segunda
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,94)= 1; %Tercera
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,95)= 1; %Cuarta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,96)= 1; %Quinta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,97)= 1; %Sexta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,98)= 1; %Séptima
ecuación

elseif Ac==2

% Guiñada
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,92)= 1; %Primera
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,93)= 1;  %Segunda
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,94)= 1; %Tercera
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,95)= 1; %Cuarta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,96)= 1; %Quinta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,97)= 1; %Sexta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,98)= 1; %Séptima
ecuación
```

```
elseif Ac==3

% Alabeo
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,92)= 1; %Primera
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,93)= 1;  %Segunda
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,94)= 1; %Tercera
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,95)= 1; %Cuarta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,96)= 1; %Quinta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,97)= 1; %Sexta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,98)= 1; %Séptima
ecuación

elseif Ac==4
% Cabeceo
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,92)= 1; %Primera
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,93)= 1;  %Segunda
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,94)= 1; %Tercera
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,95)= 1; %Cuarta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,96)= 1; %Quinta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,97)= 1; %Sexta
ecuación
Cq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,98)= 1; %Séptima
ecuación
end
end

function Ct = dtRestr(q,Ac);
global tiempo
global MBody
ngl=6;
nrestr = 7 + 4*(MBody.ncil + MBody.ncar) + (MBody.nb-1);
Ct=zeros(ngl+nrestr,1);

if Ac==1
% Prueba
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= 0; %Primera
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= 0;  %Segunda
ecuación
```

```
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= -
20*cos(5*tiempo); %Tercera ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=  0; %Cuarta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)=  0; %Quinta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  0; %Sexta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  0; %Séptima
ecuación

elseif Ac==2

% Guiñada
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= 0; %Primera
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= 0;  %Segunda
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= 0; %Tercera
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=
sin(pi/18/2*sin(5*tiempo))*pi/18/2*cos(5*tiempo)*5; %Cuarta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)=  0; %Quinta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  0; %Sexta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  -
cos(pi/18/2*sin(5*tiempo))*pi/18/2*cos(5*tiempo)*5; %Séptima
ecuación

elseif Ac==3

% Alabeo
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= 0; %Primera
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= 0;  %Segunda
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= 0; %Tercera
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=
sin(pi/18/2*sin(5*tiempo))*pi/18/2*cos(5*tiempo)*5; %Cuarta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)= -
cos(pi/18/2*sin(5*tiempo))*pi/18/2*cos(5*tiempo)*5; %Quinta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  0; %Sexta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  0; %Séptima
ecuación

elseif Ac==4
```

```
% Cabeceo
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= 0; %Primera
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= 0;  %Segunda
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= 0; %Tercera
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=
sin(pi/18/2*sin(5*tiempo))*pi/18/2*cos(5*tiempo)*5; %Cuarta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)= 0; %Quinta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)= -
cos(pi/18/2*sin(5*tiempo))*pi/18/2*cos(5*tiempo)*5; %Sexta
ecuación
Ct(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  0; %Séptima
ecuación

end

end



function DCq = DtJacobiano(q,v,nrestr,Ac);
global MBody
%% Inicializa jacobiano con dimensión (nrest+ngl)x(nrest+ngl)
porque añadimos a las ecuaciones de restricción tantas
ecuaciones de movilidad como gdl tenga el mecanismo.
ngl=6;
DCq=zeros(nrestr+ngl);

%% Barra fija
C=zeros(7);
DCq(1:7,1:7)=C;

%% Pares Cardan

for k=1:MBody.ncar

i=MBody.Pares.Car(k).I;
j=MBody.Pares.Car(k).J;
uPi=MBody.Pares.Car(k).uPi;
uPj=MBody.Pares.Car(k).uPj;
si=MBody.Pares.Car(k).si;
sj=MBody.Pares.Car(k).sj;
qi = q(7*(i-1)+1:7*(i-1)+7,1);
qj = q(7*(j-1)+1:7*(j-1)+7,1);
Ai =RotMat(qi(4:7));
Aj =RotMat(qj(4:7));
Ae0i = RotMat0(qi(4:7));
```

```
Ae1i = RotMat1(qi(4:7));
Ae2i = RotMat2(qi(4:7));
Ae3i = RotMat3(qi(4:7));
Ae0j = RotMat0(qj(4:7));
Ae1j = RotMat1(qj(4:7));
Ae2j = RotMat2(qj(4:7));
Ae3j = RotMat3(qj(4:7));

Ae00i = RotMat00(qi(4:7));
Ae11i = RotMat11(qi(4:7));
Ae22i = RotMat22(qi(4:7));
Ae33i = RotMat33(qi(4:7));
Ae00j = RotMat00(qj(4:7));
Ae11j = RotMat11(qj(4:7));
Ae22j = RotMat22(qj(4:7));
Ae33j = RotMat33(qj(4:7));

DCq(4*(k-1)+8:4*(k-1)+8+3,7*(i-1)+1:7*(i-1)+7)=[zeros(3)
Ae00i*uPi           Ae11i*uPi           Ae22i*uPi
Ae33i*uPi;
                                                zeros(1,3)
(Ae00i*si)'*(Aj*sj) (Ae11i*si)'*(Aj*sj) (Ae22i*si)'*(Aj*sj)
(Ae33i*si)'*(Aj*sj)];

DCq(4*(k-1)+8:4*(k-1)+8+3,7*(j-1)+1:7*(j-1)+7)=[-zeros(3)     -
Ae00j*uPj          -Ae11j*uPj          -Ae22j*uPj          -
Ae33j*uPj;
                                                 zeros(1,3)
(Ai*si)'*(Ae00j*sj) (Ai*si)'*(Ae11j*sj) (Ai*si)'*(Ae22j*sj)
(Ai*si)'*(Ae33j*sj)];
end

%% Pares Cilíndricos.

for k=1:MBody.ncil

i=MBody.Pares.Cil(k).I;
j=MBody.Pares.Cil(k).J;
ui=MBody.Pares.Cil(k).ui;
uj=MBody.Pares.Cil(k).uj;
vi=MBody.Pares.Cil(k).vi;
vj=MBody.Pares.Cil(k).vj;
vec=[1 0 0]';
vi1=cross(vec,vi);
vi2=cross(vi,vi1);
qi = q(7*(i-1)+1:7*(i-1)+7,1);
qj = q(7*(j-1)+1:7*(j-1)+7,1);
Ai =RotMat(qi(4:7,1));
Aj =RotMat(qj(4:7,1));
Ae0i = RotMat0(qi(4:7,1));
Ae1i = RotMat1(qi(4:7,1));
Ae2i = RotMat2(qi(4:7,1));
```

```
Ae3i = RotMat3(qi(4:7,1));
Ae0j = RotMat0(qj(4:7,1));
Ae1j = RotMat1(qj(4:7,1));
Ae2j = RotMat2(qj(4:7,1));
Ae3j = RotMat3(qj(4:7,1));

Ae00i = RotMat00(qi(4:7));
Ae11i = RotMat11(qi(4:7));
Ae22i = RotMat22(qi(4:7));
Ae33i = RotMat33(qi(4:7));
Ae00j = RotMat00(qj(4:7));
Ae11j = RotMat11(qj(4:7));
Ae22j = RotMat22(qj(4:7));
Ae33j = RotMat33(qj(4:7));

DCq(4*MBody.ncar+4*(k-1)+8:4*MBody.ncar+4*(k-1)+8+3,7*(i-
1)+1:7*(i-1)+7)=[zeros(1,3), (Ae00i*vi1)'*(Aj*vj),
(Ae11i*vi1)'*(Aj*vj),
(Ae22i*vi1)'*(Aj*vj),
(Ae33i*vi1)'*(Aj*vj);

zeros(1,3), (Ae00i*vi2)'*(Aj*vj),
(Ae11i*vi2)'*(Aj*vj),
(Ae22i*vi2)'*(Aj*vj),
(Ae33i*vi2)'*(Aj*vj);

zeros(1,3), (Ae00i*vi1)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj)+2*((Ae0i*vi1)'*(Ae0i*ui))+(Ai*vi1)'*(Ae00i*ui),
(Ae11i*vi1)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj)+2*((Ae1i*vi1)'*(Ae1i*ui))+((Ai*vi1)'*(Ae11i*ui)),
(Ae22i*vi1)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj)+2*((Ae2i*vi1)'*(Ae2i*ui))+((Ai*vi1)'*(Ae22i*ui)),
(Ae33i*vi1)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj)+2*((Ae3i*vi1)'*(Ae3i*ui))+((Ai*vi1)'*(Ae33i*ui));

zeros(1,3), (Ae00i*vi2)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj)+2*((Ae0i*vi2)'*(Ae0i*ui))+(Ai*vi2)'*(Ae00i*ui),
(Ae11i*vi2)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj)+2*((Ae1i*vi2)'*(Ae1i*ui))+((Ai*vi2)'*(Ae11i*ui)),
(Ae22i*vi2)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj)+2*((Ae2i*vi2)'*(Ae2i*ui))+((Ai*vi2)'*(Ae22i*ui)),
(Ae33i*vi2)'*(qi(1:3)+Ai*ui-qj(1:3)-
Aj*uj)+2*((Ae3i*vi2)'*(Ae3i*ui))+((Ai*vi2)'*(Ae33i*ui))];


DCq(4*MBody.ncar+4*(k-1)+8:4*MBody.ncar+4*(k-1)+8+3,7*(j-
1)+1:7*(j-1)+7)=[zeros(1,3) (Ai*vi1)'*(Ae00j*vj)
(Ai*vi1)'*(Ae11j*vj)  (Ai*vi1)'*(Ae22j*vj)
(Ai*vi1)'*(Ae33j*vj);

zeros(1,3) (Ai*vi2)'*(Ae00j*vj)  (Ai*vi2)'*(Ae11j*vj)
(Ai*vi2)'*(Ae22j*vj)  (Ai*vi2)'*(Ae33j*vj);
```

```
zeros(1,3) (Ai*vi1)'*(-Ae00j*uj) (Ai*vi1)'*(-Ae11j*uj)
(Ai*vi1)'*(-Ae22j*uj) (Ai*vi1)'*(-Ae33j*uj);

zeros(1,3) (Ai*vi2)'*(-Ae00j*uj) (Ai*vi2)'*(-Ae11j*uj)
(Ai*vi2)'*(-Ae22j*uj) (Ai*vi2)'*(-Ae33j*uj)];
end

%% Añadimos el jacobiano de las restricciones de orientación.
for k=2:MBody.nb-1

DCq(4*(MBody.ncar+MBody.ncil)+(k-2)+7+1,7*(k-1)+1:7*(k-
1)+7)=[zeros(1,3) 2 2 2 2];

end

%% Por último se añaden las ecuaciones de movilidad.

if Ac==1
% % Prueba
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,92)= 0; %Primera
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,93)= 0;  %Segunda
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,94)= 0; %Tercera
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,95)= 0; %Cuarta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,96)= 0; %Quinta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,97)= 0; %Sexta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,98)= 0; %Séptima
ecuación

elseif Ac==2

% Guiñada
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,92)= 0; %Primera
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,93)= 0;  %Segunda
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,94)= 0; %Tercera
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,95)= 0; %Cuarta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,96)= 0; %Quinta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,97)= 0; %Sexta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,98)= 0; %Séptima
ecuación
```

```
elseif Ac==3

% Alabeo
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,92)= 0; %Primera
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,93)= 0;  %Segunda
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,94)= 0; %Tercera
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,95)= 0; %Cuarta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,96)= 0; %Quinta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,97)= 0; %Sexta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,98)= 0; %Séptima
ecuación

elseif Ac==4
% Cabeceo
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,92)= 0; %Primera
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,93)= 0;  %Segunda
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,94)= 0; %Tercera
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,95)= 0; %Cuarta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,96)= 0; %Quinta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,97)= 0; %Sexta
ecuación
DCq(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,98)= 0; %Séptima
ecuación
end
end

function DCt = DtdtRestr(q,v,nrestr,Ac);
global tiempo
global MBody
ngl=6;
DCt=zeros(nrestr+ngl,1);
A=pi/18/2;
if Ac==1

% Prueba
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= 0; %Primera
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= 0;  %Segunda
ecuación
```

```
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)=
100*sin(5*tiempo); %Tercera ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=  0; %Cuarta
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)=  0; %Quinta
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  0; %Sexta
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  0; %Séptima
ecuación

elseif Ac==2

% Guiñada
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= 0; %Primera
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= 0;  %Segunda
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= 0; %Tercera
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=
5*A*(cos(A*sin(5*tiempo))*5*A*cos(5*tiempo)^2-
5*sin(5*tiempo)*sin(A*sin(5*tiempo))); %Cuarta ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)=  0; %Quinta
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  0; %Sexta
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=
5*A*(sin(A*sin(5*tiempo))*5*A*cos(5*tiempo)^2+5*sin(5*tiempo)*co
s(A*sin(5*tiempo))); %Séptima ecuación

elseif Ac==3

% Alabeo
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= 0; %Primera
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= 0;  %Segunda
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= 0; %Tercera
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=
5*A*(cos(A*sin(5*tiempo))*5*A*cos(5*tiempo)^2-
5*sin(5*tiempo)*sin(A*sin(5*tiempo))); %Cuarta ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)=
5*A*(sin(A*sin(5*tiempo))*5*A*cos(5*tiempo)^2+5*sin(5*tiempo)*co
s(A*sin(5*tiempo))); %Quinta ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=  0; %Sexta
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  0; %Séptima
ecuación

elseif Ac==4
```

```
% Cabeceo
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+8,1)= 0; %Primera
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+9,1)= 0;  %Segunda
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+10,1)= 0; %Tercera
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+11,1)=
5*A*(cos(A*sin(5*tiempo))*5*A*cos(5*tiempo)^2-
5*sin(5*tiempo)*sin(A*sin(5*tiempo))); %Cuarta ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+12,1)= 0; %Quinta
ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+13,1)=
5*A*(sin(A*sin(5*tiempo))*5*A*cos(5*tiempo)^2+5*sin(5*tiempo)*co
s(A*sin(5*tiempo))); %Sexta ecuación
DCt(4*(MBody.ncar+MBody.ncil)+(MBody.nb-2)+14,1)=  0; %Séptima
ecuación

end


end
```

4.2.4 Función estimación inicial.

```
%% Función que calcula una estimación inicial decente para
empezar a resolver el mecanismo.

function q0=EstimacionInicial(q)

%% Plataforma fija (Barra fija)
q(1:3)=0;
q(4)=1;
q(5:7)=0;

%% Barras pegadas a la plataforma fija.
vx=0;
vy=0;
vz=1;
k=2;
for i=[0 pi/6 2*pi/3 5*pi/6 4*pi/3 3*pi/2]
q(7*k-6:7*k)=[50*cos(i) 50*sin(i) 0 cos(i/2) vx*sin(i/2)
vy*sin(i/2) vz*sin(i/2)];
k=k+1;
end

%% Barras pegadas a la plataforma móvil.
vx=0;
vy=0;
```

```
vz=1;
% k=8; La primera barra pegada a la plataforma movil es el
sólido 8. Ya
% viene del bucle anterior.
for i=[0+pi/3 pi/6+pi/3 2*pi/3+pi/3 5*pi/6+pi/3 4*pi/3+pi/3
3*pi/2+pi/3] %Cuando la giremos, aquí habrá que poner 60º mas.
q(7*k-6:7*k)=[50*cos(i) 50*sin(i) 50 cos(i/2) vx*sin(i/2)
vy*sin(i/2) vz*sin(i/2)];
k=k+1;
end

%% Plataforma móvil.
vx=0;
vy=0;
vz=1;
i=0; %El Sistema de Referencia no está girado respecto a la fija
q(7*k-6:7*k)=[0 0 50 cos(i/2) vx*sin(i/2) vy*sin(i/2)
vz*sin(i/2)];

%% Lo metemos todo en el vector q0
q0=q;
end
```

4.2.5 Programa principal. Simulación cinemática.

```
%% Programa principal


clear all
close all
clc
global MBody
global tiempo
global tspan
StewartPlatform;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SIMULACIÓN CINEMÁTICA PLATAFORMA STEWART %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

MBody.w = 10*(2*pi/60); %VELOCIDAD ANGULAR (10 rpm)
tspan = 0:0.1:(4*pi/MBody.w); %TIEMPO DE INTEGRACIÓN (2 vueltas)

%% Estimacion inicial
Ac=input('Actuación= ');
q0 = zeros(98,1);
q0 = EstimacionInicial(q0);
nrestr =7+4*(MBody.ncil+MBody.ncar)+(MBody.nb-1);

%% Bucle para resolver el problema de posición
```

```
for i = 1:length(tspan)
%PROBLEMA DE POSICION
tiempo=tspan(i);
q(:,i) = fsolve(@Restricciones,q0,optimset('Display','on'));
Cq = Jacobiano(q(:,i),nrestr,Ac);
Ct = dtRestr(q(:,i),Ac);%,tspan(i));
%CALCULO DE VELOCIDADES
v(:,i) = -Cq\Ct;
DCq = DtJacobiano(q(:,i),v(:,i),nrestr,Ac);
DCt = DtdtRestr(q(:,i),v(:,i),nrestr,Ac);
% %CALCULO DE ACELERACIONES
a(:,i) = -Cq\(DCq*v(:,i)+DCt);
q0 = q(:,i);
end

p=1;
for k=1:MBody.nb
figure(p)
plot(tspan,v(1+7*(k-1),:),'g')
hold on
plot(tspan,v(2+7*(k-1),:),'b')
plot(tspan,v(3+7*(k-1),:),'r')
xlabel('t(s)')
ylabel('v(cm/s)')
title(['Velocidades solido ', num2str(k)])
legend('Eje x','Eje y','Eje z')
% filename=['Vel_' num2str(k) '.png'];
% saveas([p],filename)
p=p+1;
end


for k=1:MBody.nb
figure(p)
plot(tspan,a(1+7*(k-1),:),'g')
hold on
plot(tspan,a(2+7*(k-1),:),'b')
plot(tspan,a(3+7*(k-1),:),'r')
xlabel('t(s)')
ylabel('a(cm/s^2)')
title(['Aceleraciones solido ', num2str(k)])
legend('Eje x','Eje y','Eje z')
% filename=['Ac_' num2str(k) '.png'];
% saveas([p],filename)
p=p+1;
end

for k=8:MBody.nb %Para ver el desfase de aceleración y
velocidad.
figure(p)
plot(tspan,v(3+7*(k-1),:),'r')
hold on
```

```
plot(tspan,a(3+7*(k-1),:),'b')
xlabel('t(s)')
ylabel('a(cm/s^2),v(cm/s)')
title(['Desfases solido ', num2str(k)])
legend('Velocidad (cm/s)','Aceleración (cm/s^2)')
% filename=['Des_' num2str(k) '.png'];
% saveas([p],filename)
p=p+1;
end

% figure
% AnimaStewart(q)
```

4.2.6 Animación del mecanismo.

```
%% Programa de animación del mecanismo.

function AnimaStewart(q)

global tspan
%% Animación Plataforma de Stewart.
np=length(tspan);
nc=1;
%% Definición de los puntos notables del mecanismo.

% Barras pegadas a la plataforma fija
x02 = q(8,:);
y02 = q(9,:);
z02 = q(10,:);
x03 = q(15,:);
y03 = q(16,:);
z03 = q(17,:);
x04 = q(22,:);
y04 = q(23,:);
z04 = q(24,:);
x05 = q(29,:);
y05 = q(30,:);
z05 = q(31,:);
x06 = q(36,:);
y06 = q(37,:);
z06 = q(38,:);
x07 = q(43,:);
y07 = q(44,:);
z07 = q(45,:);

% Barras pegadas a la plataforma móvil.

x08 = q(50,:);
y08 = q(51,:);
z08 = q(52,:);
x09 = q(57,:);
```

```
y09 = q(58,:);
z09 = q(59,:);
x10 = q(64,:);
y10 = q(65,:);
z10 = q(66,:);
x11 = q(71,:);
y11 = q(72,:);
z11 = q(73,:);
x12 = q(78,:);
y12 = q(79,:);
z12 = q(80,:);
x13 = q(85,:);
y13 = q(86,:);
z13 = q(87,:);

%% Definicion de las lineas a dibujar.

% Barras.
x1 = [x03; x08];
y1 = [y03; y08];
z1 = [z03; z08];

x2 = [x04; x09];
y2 = [y04; y09];
z2 = [z04; z09];

x3 = [x05; x10];
y3 = [y05; y10];
z3 = [z05; z10];

x4 = [x06; x11];
y4 = [y06; y11];
z4 = [z06; z11];

x5 = [x07; x12];
y5 = [y07; y12];
z5 = [z07; z12];

x6 = [x02; x13];
y6 = [y02; y13];
z6 = [z02; z13];

% Plataforma fija.

x7 = [x02; x03; x04; x05; x06; x07; x02];
y7 = [y02; y03; y04; y05; y06; y07; y02];
z7 = [z02; z03; z04; z05; z06; z07; z02];

% Plataforma móvil.

x8 = [x08; x09; x10; x11; x12; x13; x08];
y8 = [y08; y09; y10; y11; y12; y13; y08];
```

```
z8 = [z08; z09; z10; z11; z12; z13; z08];
%************************************

for j=1:nc,
    for i=2:(np),

        % Se pintan las seis barras

        plot3(x1(:,i),y1(:,i),z1(:,i),'b')
        hold on
        plot3(x2(:,i),y2(:,i),z2(:,i),'b')
        plot3(x3(:,i),y3(:,i),z3(:,i),'b')
        plot3(x4(:,i),y4(:,i),z4(:,i),'b')
        plot3(x5(:,i),y5(:,i),z5(:,i),'b')
        plot3(x6(:,i),y6(:,i),z6(:,i),'b')

        % Se pintan la plataforma fija y móvil

        plot3(x7(:,i),y7(:,i),z7(:,i),'k')
        plot3(x8(:,i),y8(:,i),z8(:,i),'r')
        hold off

        % Fijamos los ejes para evitar la distorsión y
establecemos un pause
        % para que de tiempo a verlo. Poner el mismo rango para
que no se
        % distorsione figura (circulo y no elipse)
        axis([-75 75 -75 75 0 150])
        axis square off
        pause(.1)
    end
end
end
```

4.2.7 Matrices de rotación.

```
function A=RotMat(x)
I=eye(3);
theta=[0 -x(4) x(3);x(4) 0 -x(2);-x(3) x(2) 0];
A=I+2*theta*(I*x(1)+theta);
end


function Ae0=RotMat0(x)

Ae0=[0     -2*x(4) 2*x(3);
     2*x(4) 0     -2*x(2);
    -2*x(3) 2*x(2)     0];

end
```

```
function Ae1=RotMat1(x)

Ae1=[0       2*x(3)   2*x(4);
     2*x(3) -4*x(2)  -2*x(1);
     2*x(4)  2*x(1)  -4*x(2)];

end


function Ae2=RotMat2(x)

Ae2=[-4*x(3)  2*x(2)   2*x(1);
      2*x(2)       0   2*x(4);
     -2*x(1)  2*x(4)  -4*x(3)];

end


function Ae3=RotMat3(x)

Ae3=[-4*x(4) -2*x(1)   2*x(2);
      2*x(1) -4*x(4)   2*x(3);
      2*x(2)  2*x(3)       0];

end


function Ae00=RotMat00(x)
Ae00=zeros(3);
end


function Ae11=RotMat11(x)

Ae11=[0  0  0;
      0 -4  0;
      0  0 -4];
end


function Ae22=RotMat22(x)

Ae22=[-4  0  0;
       0  0  0;
       0  0 -4];

end

function Ae33=RotMat33(x)
```

```
Ae33=[-4  0  0;
       0 -4  0;
       0  0  0];

end
```

4.2.8 Dinámica inversa plataforma de Stewart.

```
%% Programa que calcula la dinámica inversa. Método 2, sin cambiar los
4 parámetros de Euler por los 3 ángulos.
global MBody tspan
ngl=6;
nrestr =7+4*(MBody.ncil+MBody.ncar)+(MBody.nb-1);
rho=2700; %Densidad aluminio (Kg/m^3)
h=0.05; %Espesor de la plataforma (m)
d=0.015; %Diámetro de las barras (m)
g=9.81; %Gravedad (m/s^2)
Ac=input('¿En qué actuación estamos?: ');

%% Longitudes de las barras. (En m)
D1=1; L2=0.3; L3=0.3; L4=0.3; L5=0.3; L6=0.3; L7=0.3;
L8=0.3;  L9=0.3; L10=0.3;L11=0.3;L12=0.3;L13=0.3;D14=1;

%% Cálculos de masas (En kg)
M1=pi*D1^2/4*h*rho; M2=pi*d^2/4*L2*rho; M3=pi*d^2/4*L3*rho;
M4=pi*d^2/4*L4*rho;
M5=pi*d^2/4*L5*rho; M6=pi*d^2/4*L6*rho; M7=pi*d^2/4*L7*rho;
M8=pi*d^2/4*L8*rho;
M9=pi*d^2/4*L9*rho; M10=pi*d^2/4*L10*rho; M11=pi*d^2/4*L11*rho;
M12=pi*d^2/4*L12*rho;
M13=pi*d^2/4*L13*rho; M14=pi*D14^2/4*h*rho;

%% Cálculos de inercias. (Kg m^2 )
I1_x=1/4*M1*(D1/2)^2; I1_y=1/4*M1*(D1/2)^2; I1_z=1/2*M1*(D1/2)^2;
I2_x=1/12*M2*(3*(d/2)^2+L2^2); I2_y=1/12*M2*(3*(d/2)^2+L2^2);
I2_z=1/2*M2*(d/2)^2;
I3_x=1/12*M3*(3*(d/2)^2+L3^2); I3_y=1/12*M3*(3*(d/2)^2+L3^2);
I3_z=1/2*M3*(d/2)^2;
I4_x=1/12*M4*(3*(d/2)^2+L4^2); I4_y=1/12*M4*(3*(d/2)^2+L4^2);
I4_z=1/2*M4*(d/2)^2;
I5_x=1/12*M5*(3*(d/2)^2+L5^2); I5_y=1/12*M5*(3*(d/2)^2+L5^2);
I5_z=1/2*M5*(d/2)^2;
I6_x=1/12*M6*(3*(d/2)^2+L6^2); I6_y=1/12*M6*(3*(d/2)^2+L6^2);
I6_z=1/2*M6*(d/2)^2;
I7_x=1/12*M7*(3*(d/2)^2+L7^2); I7_y=1/12*M7*(3*(d/2)^2+L7^2);
I7_z=1/2*M7*(d/2)^2;
I8_x=1/12*M8*(3*(d/2)^2+L8^2); I8_y=1/12*M8*(3*(d/2)^2+L8^2);
I8_z=1/2*M8*(d/2)^2;
I9_x=1/12*M9*(3*(d/2)^2+L9^2); I9_y=1/12*M9*(3*(d/2)^2+L9^2);
I9_z=1/2*M9*(d/2)^2;
I10_x=1/12*M10*(3*(d/2)^2+L10^2); I10_y=1/12*M10*(3*(d/2)^2+L10^2);
I10_z=1/2*M10*(d/2)^2;
```

```
I11_x=1/12*M11*(3*(d/2)^2+L11^2); I11_y=1/12*M11*(3*(d/2)^2+L11^2);
I11_z=1/2*M11*(d/2)^2;
I12_x=1/12*M12*(3*(d/2)^2+L12^2); I12_y=1/12*M12*(3*(d/2)^2+L12^2);
I12_z=1/2*M12*(d/2)^2;
I13_x=1/12*M13*(3*(d/2)^2+L13^2); I13_y=1/12*M13*(3*(d/2)^2+L13^2);
I13_z=1/2*M13*(d/2)^2;
I14_x=1/4*M14*(D14/2)^2; I14_y=1/4*M14*(D14/2)^2;
I14_z=1/2*M14*(D14/2)^2;

I=eye(3); %Matriz identidad de orden 3
%% Vector con todas las masas e inercias del mecanismo. En orden. Las
6 primeras corresponden al sólido 1, 3 masas y 3 inercias x, y y z
MI=[M1 M1 M1 I1_x I1_y I1_z M2 M2 M2 I2_x I2_y I2_z M3 M3 M3 I3_x I3_y
I3_z M4 M4 M4 I4_x I4_y I4_z M5 M5 M5 I5_x I5_y I5_z M6 M6 M6 I6_x
I6_y I6_z M7 M7 M7 I7_x I7_y I7_z M8 M8 M8 I8_x I8_y I8_z M9 M9 M9
I9_x I9_y I9_z M10 M10 M10 I10_x I10_y I10_z M11 M11 M11 I11_x I11_y
I11_z M12 M12 M12 I12_x I12_y I12_z M13 M13 M13 I13_x I13_y I13_z M14
M14 M14 I14_x I14_y I14_z]';

%% Una vez se tiene el vector, se monta la matriz M y la g.
M=zeros(7*MBody.nb); %La matriz tendrá 7 filas y columnas por cada
sólido.
g0=zeros(7*MBody.nb,1);


%% Comenzamos el bucle del tiempo, donde para cada instante de tiempo
se calculan las matrices B,h,b y gamma.
lambda=zeros(98,length(tspan));
lambda1=zeros(98,length(tspan));

F_x=zeros(98,length(tspan));
F_y=zeros(98,length(tspan));
F_z=zeros(98,length(tspan));
F_e0=zeros(98,length(tspan));
F_e1=zeros(98,length(tspan));
F_e2=zeros(98,length(tspan));
F_e3=zeros(98,length(tspan));
for i=1:length(tspan)
    for k=1:MBody.nb
    M(1+7*(k-1):3+7*(k-1),1+7*(k-1):3+7*(k-1))=MI(1+6*(k-1):3+6*(k-
1)).*I;
    J=(MI(4+6*(k-1):6+6*(k-1)).*I);
    L=MatL(q(4+7*(k-1):7+7*(k-1),1));
    M(4+7*(k-1):7+7*(k-1),4+7*(k-1):7+7*(k-1))=4*L'*J*L;
if k==1 || k==14 % Este if se puede hacer así porque las masas de
todas las barras son iguales, y las de la plataforma también.
g0(3+7*(k-1),1)=-M1*g;
else
g0(3+7*(k-1),1)=-M2*g;
end

L_punto=MatL(v(4+7*(k-1):7+7*(k-1),1));
g0(4+7*(k-1):7+7*(k-1),1)=8*L_punto'*J*L_punto*q(4+7*(k-1):7+7*(k-
1),1);

end
```

```
Cq=Jacobiano(q(:,i),nrestr,Ac);
lambda(:,i)=Cq'\(g0-M*a(:,i));

%% Ahora se va a calcular la fuerza en los pistones aplicando el TPV.

% Sacar las fuerzas de la plataforma móvil (Fx,Fy,Fz,Fe0,Fe1,Fe2,Fe3).
% Fx
lambda1(92,i)=lambda(92,i);
F_x(:,i)=Cq'*lambda1(:,i);
lambda1(92,i)=0;
% Fy
lambda1(93,i)=lambda(93,i);
F_y(:,i)=Cq'*lambda1(:,i);
lambda1(93,i)=0;
% Fz
lambda1(94,i)=lambda(94,i);
F_z(:,i)=Cq'*lambda1(:,i);
lambda1(94,i)=0;
% Fe0
lambda1(95,i)=lambda(95,i);
F_e0(:,i)=Cq'*lambda1(:,i);
lambda1(95,i)=0;
% Fe1
lambda1(96,i)=lambda(96,i);
F_e1(:,i)=Cq'*lambda1(:,i);
lambda1(96,i)=0;
% Fe2
lambda1(97,i)=lambda(97,i);
F_e2(:,i)=Cq'*lambda1(:,i);
lambda1(97,i)=0;
% Fe3
lambda1(98,i)=lambda(98,i);
F_e3(:,i)=Cq'*lambda1(:,i);
lambda1(98,i)=0;

end

% figure
% plot(tspan,F_x(92,:))
% hold on
% plot(tspan,F_y(93,:))
% plot(tspan,F_z(94,:))
% xlabel('t(s)')
% ylabel('F(N)')
% title('Fuerzas plataforma móvil.')
% legend('F_x (N)','F_y (N)','F_z (N)')
% filename=['Fuerza_Fija.png'];
% saveas([1],filename)
% plot(tspan,F_e0(95,:))
% plot(tspan,F_e1(96,:))
% plot(tspan,F_e2(97,:))
% plot(tspan,F_e3(98,:))
%
% title('Fuerza')
% xlabel('Tiempo (s)')
% ylabel('Fuerza (N)')
% legend('F_x','F_y','F_z','F_e0','F_e1','F_e2','F_e3')
```

```
% % filename=['Fuerza_Z.png'];
% % saveas([1],filename)

% Calculamos ahora los "deltas" que multiplican a las fuerzas de los
pistones.

F213=zeros(1,91);
F38=zeros(1,91);
F49=zeros(1,91);
F510=zeros(1,91);
F611=zeros(1,91);
F712=zeros(1,91);

Res_213=zeros(7,length(tspan));
Res_38=zeros(7,length(tspan));
Res_49=zeros(7,length(tspan));
Res_510=zeros(7,length(tspan));
Res_611=zeros(7,length(tspan));
Res_712=zeros(7,length(tspan));

for i=1:length(tspan)

    Cq=Jacobiano(q(:,i),nrestr,Ac);
    Cqd=Cq(1:91,1:91);
    Cqi=Cq(1:91,92:98);
    deltaqd=-(Cqd)\Cqi;
    %Pistón 2-13
    At=RotMat(q(11:14,i));
    r1=q(8,i);
    r2=q(9,i);
    r3=q(10,i);
    e0=q(11,i);
    e1=q(12,i);
    e2=q(13,i);
    e3=q(14,i);
    p1=q(85,i);
    p2=q(86,i);
    p3=q(87,i);
    F213(1,8:10)=[At(3,1) At(3,2) At(3,3)];
    F213(1,85:87)=[-At(3,1) -At(3,2) -At(3,3)];
    F213(1,11:14)=2*[At(3,1)*(-e2*(p3-r3)+e3*(p2-r2))+At(3,2)*(e1*(p3-
r3)-e3*(p1-r1))+At(3,3)*(-e1*(p2-r2)+e2*(p1-r1));
                     At(3,1)*(e3*(p3-r3)-e2*(p2-r2))+At(3,2)*(-e0*(p3-
r3)-e2*(p1-r1))+At(3,3)*(e0*(p2-r2)-e3*(p1-r1));
                     At(3,1)*(e0*(p3-r3)-e1*(p2-r2))+At(3,2)*(-e3*(p3-
r3)+e1*(p1-r1))+At(3,3)*(-e3*(p2-r2)+e0*(p1-r1));
                     At(3,1)*(-e1*(p3-r3)+e0*(p2-r2))+At(3,2)*(-e2*(p3-
r3)+e0*(p1-r1))+At(3,3)*(e2*(p2-r2)+e1*(p1-r1))]';

    Res_213(:,i)=(F213*deltaqd)'; %Vector de 7*1 que multiplica a las
"deltas" independientes. Sale de multiplicar 1*91 x 91*7 y trasponer
    %Pistón 3-8
    At=RotMat(q(18:21,i));
    r1=q(15,i);
    r2=q(16,i);
    r3=q(17,i);
    e0=q(18,i);
```

```
    e1=q(19,i);
    e2=q(20,i);
    e3=q(21,i);
    p1=q(50,i);
    p2=q(51,i);
    p3=q(52,i);
    F38(1,15:17)=[At(3,1) At(3,2) At(3,3)];
    F38(1,50:52)=[-At(3,1) -At(3,2) -At(3,3)];
    F38(1,18:21)=2*[At(3,1)*(-e2*(p3-r3)+e3*(p2-r2))+At(3,2)*(e1*(p3-
r3)-e3*(p1-r1))+At(3,3)*(-e1*(p2-r2)+e2*(p1-r1));
                    At(3,1)*(e3*(p3-r3)-e2*(p2-r2))+At(3,2)*(-e0*(p3-
r3)-e2*(p1-r1))+At(3,3)*(e0*(p2-r2)-e3*(p1-r1));
                    At(3,1)*(e0*(p3-r3)-e1*(p2-r2))+At(3,2)*(-e3*(p3-
r3)+e1*(p1-r1))+At(3,3)*(-e3*(p2-r2)+e0*(p1-r1));
                    At(3,1)*(-e1*(p3-r3)+e0*(p2-r2))+At(3,2)*(-e2*(p3-
r3)+e0*(p1-r1))+At(3,3)*(e2*(p2-r2)+e1*(p1-r1))]';

    Res_38(:,i)=(F38*deltaqd)';
    %Pistón 4-9
    At=RotMat(q(25:28,i));
    r1=q(22,i);
    r2=q(23,i);
    r3=q(24,i);
    e0=q(25,i);
    e1=q(26,i);
    e2=q(27,i);
    e3=q(28,i);
    p1=q(57,i);
    p2=q(58,i);
    p3=q(59,i);
    F49(1,22:24)=[At(3,1) At(3,2) At(3,3)];
    F49(1,57:59)=[-At(3,1) -At(3,2) -At(3,3)];
    F49(1,25:28)=2*[At(3,1)*(-e2*(p3-r3)+e3*(p2-r2))+At(3,2)*(e1*(p3-
r3)-e3*(p1-r1))+At(3,3)*(-e1*(p2-r2)+e2*(p1-r1));
                    At(3,1)*(e3*(p3-r3)-e2*(p2-r2))+At(3,2)*(-e0*(p3-
r3)-e2*(p1-r1))+At(3,3)*(e0*(p2-r2)-e3*(p1-r1));
                    At(3,1)*(e0*(p3-r3)-e1*(p2-r2))+At(3,2)*(-e3*(p3-
r3)+e1*(p1-r1))+At(3,3)*(-e3*(p2-r2)+e0*(p1-r1));
                    At(3,1)*(-e1*(p3-r3)+e0*(p2-r2))+At(3,2)*(-e2*(p3-
r3)+e0*(p1-r1))+At(3,3)*(e2*(p2-r2)+e1*(p1-r1))]';
    Res_49(:,i)=(F49*deltaqd)';

    %Pistón 5-10
    At=RotMat(q(32:35,i));
    r1=q(29,i);
    r2=q(30,i);
    r3=q(31,i);
    e0=q(32,i);
    e1=q(33,i);
    e2=q(34,i);
    e3=q(35,i);
    p1=q(64,i);
    p2=q(65,i);
    p3=q(66,i);
    F510(1,29:31)=[At(3,1) At(3,2) At(3,3)];
    F510(1,64:66)=[-At(3,1) -At(3,2) -At(3,3)];
```

```
    F510(1,32:35)=2*[At(3,1)*(-e2*(p3-r3)+e3*(p2-r2))+At(3,2)*(e1*(p3-
r3)-e3*(p1-r1))+At(3,3)*(-e1*(p2-r2)+e2*(p1-r1));
                    At(3,1)*(e3*(p3-r3)-e2*(p2-r2))+At(3,2)*(-e0*(p3-
r3)-e2*(p1-r1))+At(3,3)*(e0*(p2-r2)-e3*(p1-r1));
                    At(3,1)*(e0*(p3-r3)-e1*(p2-r2))+At(3,2)*(-e3*(p3-
r3)+e1*(p1-r1))+At(3,3)*(-e3*(p2-r2)+e0*(p1-r1));
                    At(3,1)*(-e1*(p3-r3)+e0*(p2-r2))+At(3,2)*(-e2*(p3-
r3)+e0*(p1-r1))+At(3,3)*(e2*(p2-r2)+e1*(p1-r1))]';

    Res_510(:,i)=(F510*deltaqd)';

    %Pistón 6-11
    At=RotMat(q(39:42,i));
    r1=q(36,i);
    r2=q(37,i);
    r3=q(38,i);
    e0=q(39,i);
    e1=q(40,i);
    e2=q(41,i);
    e3=q(42,i);
    p1=q(71,i);
    p2=q(72,i);
    p3=q(73,i);
    F611(1,36:38)=[At(3,1) At(3,2) At(3,3)];
    F611(1,71:73)=[-At(3,1) -At(3,2) -At(3,3)];
    F611(1,39:42)=2*[At(3,1)*(-e2*(p3-r3)+e3*(p2-r2))+At(3,2)*(e1*(p3-
r3)-e3*(p1-r1))+At(3,3)*(-e1*(p2-r2)+e2*(p1-r1));
                    At(3,1)*(e3*(p3-r3)-e2*(p2-r2))+At(3,2)*(-e0*(p3-
r3)-e2*(p1-r1))+At(3,3)*(e0*(p2-r2)-e3*(p1-r1));
                    At(3,1)*(e0*(p3-r3)-e1*(p2-r2))+At(3,2)*(-e3*(p3-
r3)+e1*(p1-r1))+At(3,3)*(-e3*(p2-r2)+e0*(p1-r1));
                    At(3,1)*(-e1*(p3-r3)+e0*(p2-r2))+At(3,2)*(-e2*(p3-
r3)+e0*(p1-r1))+At(3,3)*(e2*(p2-r2)+e1*(p1-r1))]';

    Res_611(:,i)=(F611*deltaqd)';

    %Pistón 7-12
    At=RotMat(q(46:49,i));
    r1=q(43,i);
    r2=q(44,i);
    r3=q(45,i);
    e0=q(46,i);
    e1=q(47,i);
    e2=q(48,i);
    e3=q(49,i);
    p1=q(78,i);
    p2=q(79,i);
    p3=q(80,i);
    F712(1,43:45)=[At(3,1) At(3,2) At(3,3)];
    F712(1,78:80)=[-At(3,1) -At(3,2) -At(3,3)];
    F712(1,46:49)=2*[At(3,1)*(-e2*(p3-r3)+e3*(p2-r2))+At(3,2)*(e1*(p3-
r3)-e3*(p1-r1))+At(3,3)*(-e1*(p2-r2)+e2*(p1-r1));
                    At(3,1)*(e3*(p3-r3)-e2*(p2-r2))+At(3,2)*(-e0*(p3-
r3)-e2*(p1-r1))+At(3,3)*(e0*(p2-r2)-e3*(p1-r1));
                    At(3,1)*(e0*(p3-r3)-e1*(p2-r2))+At(3,2)*(-e3*(p3-
r3)+e1*(p1-r1))+At(3,3)*(-e3*(p2-r2)+e0*(p1-r1));
```

```
                    At(3,1)*(-e1*(p3-r3)+e0*(p2-r2))+At(3,2)*(-e2*(p3-
r3)+e0*(p1-r1))+At(3,3)*(e2*(p2-r2)+e1*(p1-r1))]';

    Res_712(:,i)=(F712*deltaqd)';

    %% Se crea el sistema a resolver.

    A=zeros(7,6); %Matriz que contiene los "deltas" multiplicando a
las fuerzas de los pistones.
    A(:,1)=Res_213(:,i);
    A(:,2)=Res_38(:,i);
    A(:,3)=Res_49(:,i);
    A(:,4)=Res_510(:,i);
    A(:,5)=Res_611(:,i);
    A(:,6)=Res_712(:,i);

    b=zeros(7,1);
    b(1,1)=F_x(92,i);
    b(2,1)=F_y(93,i);
    b(3,1)=F_z(94,i);
    b(4,1)=F_e0(95,i);
    b(5,1)=F_e1(96,i);
    b(6,1)=F_e2(97,i);
    b(7,1)=F_e3(98,i);

%     if (A(3,1)==A(3,3)) && (A(3,1)==A(3,5)) && (A(3,2)==A(3,4)) &&
(A(3,1)==A(3,6)) %& (A(3,1)==A(3,2))
%
%         x(:,i)=b(3,1)/6/A(3,1)*ones(6,1);
%
%     end
%     diff=A*x(:,i)-b;
%     max(diff)

    x(:,i)=A\b; %NO SE PUEDE PORQUE TIENE ECUACIONES REDUNTANTES. X
sería el vector donde están las fuerzas (en la dirección Z) de los
pistones para el instante i de tiempo.
end
figure(1)
plot(tspan,x(1,:))
title('Pistón 2-13')
xlabel('Tiempo (s)')
ylabel('Fuerza (N)')
filename=['Piston_2_13_Guiñada.png'];
saveas([1],filename)
figure(2)
plot(tspan,x(2,:))
title('Pistón 3-8')
xlabel('Tiempo (s)')
ylabel('Fuerza (N)')
filename=['Piston_3_8_Guiñada.png'];
saveas([2],filename)
figure(3)
plot(tspan,x(3,:))
title('Pistón 4-9')
xlabel('Tiempo (s)')
ylabel('Fuerza (N)')
```

```
filename=['Piston_4_9_Guiñada.png'];
saveas([3],filename)
figure(4)
plot(tspan,x(4,:))
title('Pistón 5-10')
xlabel('Tiempo (s)')
ylabel('Fuerza (N)')
filename=['Piston_5_10_Guiñada.png'];
saveas([4],filename)
figure(5)
plot(tspan,x(5,:))
title('Pistón 6-11')
xlabel('Tiempo (s)')
ylabel('Fuerza (N)')
filename=['Piston_6_11_Guiñada.png'];
saveas([5],filename)
figure(6)
plot(tspan,x(6,:))
title('Pistón 7-12')
xlabel('Tiempo (s)')
ylabel('Fuerza (N)')
filename=['Piston_7_12_Guiñada.png'];
saveas([6],filename)
```

REFERENCIAS

[1] Carlos Borrás P., Katherin Duarte B.: Generalidades de robots paralelos.

[2] Edward J. Haug, COMPUTER AIDED KINEMATICS AND DYNAMICS OF MECHANICAL SYSTEMS

[3] Isidro Zabalza, Javier Ros: Aplicaciones acutales de los robots paralelos.
http://congreso.pucp.edu.pe/cibim8/pdf/19/19-02.pdf

[4] José L. Escalona: Apuntes de Dinámica de Sistemas Multicuerpos.

[5] Parviz E. Nikravesh COMPUTER AIDED ANALYSIS OF MECHANICAL SYSTEMS.