E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

Identificación de parámetros dinámicos del robot KUKA LBR IIWA

Máster Universitario
en Ingeniería Industrial

Trabajo Fin de Máster

Julio Cesar Ruiz Ramirez

Xabier Iriarte Goñi

Pamplona, 21 de febrero 2022

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

.

Enseñar es aprender dos veces.
Joseph Joubert

.

Agradecimientos

Con este trabajo cierro una pequeña etapa en términos de tiempo de mi aprendizaje, pero de las más enriquecedoras en conocimiento de mi vida. Ha sido una bonita experiencia que he tratado de aprovechar al máximo.

Le doy las gracias a mi tutor del trabajo, Xabier Iriarte. Por su gran ayuda, por la enorme paciencia que ha tenido en responder las innumerables preguntas que le hice, así como sus acertadas correcciones a mis errores y en fin por todo lo que me ha enseñado.

Agradecer a todo el personal docente del máster, los cuales me han brindado gran parte del conocimiento empleado para llevar a cabo este trabajo, siendo promotores en primera fila de mi despertar e interés en diversas áreas de estudio.

Agradecer a mi familia y amigos, quienes me animaron a la distancia y fueron partícipe de largas conversaciones en donde les expresaba mis grandes emociones y retos durante esta etapa.

Especialmente a Francisco y Marcela, quienes han sido pilar fundamental durante este viaje, que estando a miles de kilómetros, no han dejado de empujarme hacia adelante, desde el comienzo.

En fin, agradecerles a todos. Ya que sin ustedes este trabajo no hubiese sido pasible.

.

Resumen

Este trabajo trata sobre la identificación de los parámetros dinámicos del robot manipulador KUKA LBR iiwa 14 R820 (robot "asistente de trabajo industrial inteligente"), comenzando por la construcción de un modelo dinámico del robot que sea lineal en los parámetros de inercia y de fricción. Para la validación se aplica la reducción de los parámetros dinámicos a parámetros base para poder garantizar su estimación. Los experimentos del robot serán basados en trayectorias de excitación optimizadas. Una vez realizados los experimentos se realizará una validación cruzada con otra trayectoria optimizada para medir la validez del modelo.

Palabras clave: Identificación de Parámetros, Parametrización, Optimización, Modelado, Reducción.

Índice

Índice de figuras

Índice de cuadros

1. INTRODUCCIÓN

1.1. Motivación

Una de las motivaciones principales de este trabajo, es continuar con el estudio y análisis del modelado de sistemas multi-cuerpo (Multibody) e identificación de sistemas dinámicos comenzados en los estudios durante el máster.

La idea de llevar a cabo el experimento y los retos que ello conlleva, desde las pruebas previas, la instrumentación a utilizar, adquisición y tratamiento de los datos así como la validación del modelo construido con el real.

Me parece interesante la manipulación matemática requerida y apropiada para encontrar la única, o las mejores soluciones al sistema, como ser, reducciones, cálculo vectorial, técnicas estadística y optimizaciones.

El presente trabajo se ha basado principalmente en la aplicación del conocimiento aprendido por el autor en las asignaturas del primer curso del máster, especialmente en la de Dinámica de Sistemas Multicuerpo e Identificación de Sistemas Mecánicos, y en el acompañamiento del tutor, como una guía continua hasta su finalización.

1.2. Objetivos

El Objetivo principal de este trabajo de fin de máster es la identificación de los parámetros dinámicos del Robot KUKA LBR iiwa R14 820 de la empresa ALDAKIN. Así, los objetivos específicos han sido:

- Construir el modelado dinámico del robot que incluya todos los efectos dinámicos significativos.
- La reducción del modelo dinámico a parámetros base, construyendo un modelo lineal en los parámetros dinámicos.
- Realizar la optimización de las trayectorias con las que se comanda al robot en los experimentos utilizando dos criterios.
- Llevar a cabo los experimentos con el robot de estudio utilizando una de las trayectorias ya optimizadas y obtener los valores medidos de las posiciones y torques ejercidos por los motores, los cuales son elementos fundamentales para la estimación de los parámetros.
- Realizar la estimación de los parámetros a partir del modelo creado y los datos medidos de los experimentos.
- Validar el modelo construido y los parámetros estimados con trayectorias distintas

1.3. Robot en serie Kuka LBR iiwa

El robot utilizado para el presente trabajo es el robot en serie LBR iiwa 14 R820 de la marca KUKA Roboter GmbH. El LBR iiwa es el primer robot fabricado en serie con capacidad sensitiva, y el primero preparado para la colaboración entre personas y robots. LBR es la abreviatura utilizada para "robot de estructura liviana", mientras que iiwa es la abreviatura de "intelligent industrial work assistant ", el 14 corresponde a la capacidad de 14 Kg y el 820 el alcance máximo en mm de la zona de trabajo.

Figura 1.1: Robot KUKA LBR iiwa[1]

Descripción del LBR iiwa

El LBR iiwa está clasificado como un robot ligero y es un robot de brazo articulado con 7 ejes. Todas las unidades de motor y los cables que transportan corriente están protegidos por placas de cubierta.

Cada eje está protegido por medio de sensores de rango de eje y se puede ajustar mediante sensores internos. Cada articulación está equipada con un sensor de posición en el lado de entrada, sensores de par en el lado de salida y sensores de temperatura. Por lo tanto, el robot puede ser operado con control de posición e impedancia. Los sensores de temperatura evitan la sobrecarga térmica del robot.

El manual de especificaciones provisto por el fabricante indica que el robot está equipado de forma redundante y consta de los siguientes componentes principales [2]:

- **Muñeca en línea:** El robot está equipado con una muñeca en línea de 2 ejes. Los motores están ubicados en los ejes A6 y A7.
- **Módulo de unión:** Los módulos de unión están formados por una estructura de aluminio. Las unidades de accionamiento están situadas dentro de estos módulos. De esta manera, las unidades de accionamiento están conectadas entre sí a través de las estructuras de aluminio.

- **Bastidor base:** El bastidor base, es la base del robot. La interfaz A1 se encuentra en la parte trasera de el marco base. Constituye la interfaz para los cables de conexión entre el robot, el controlador y el sistema de suministro de energía.

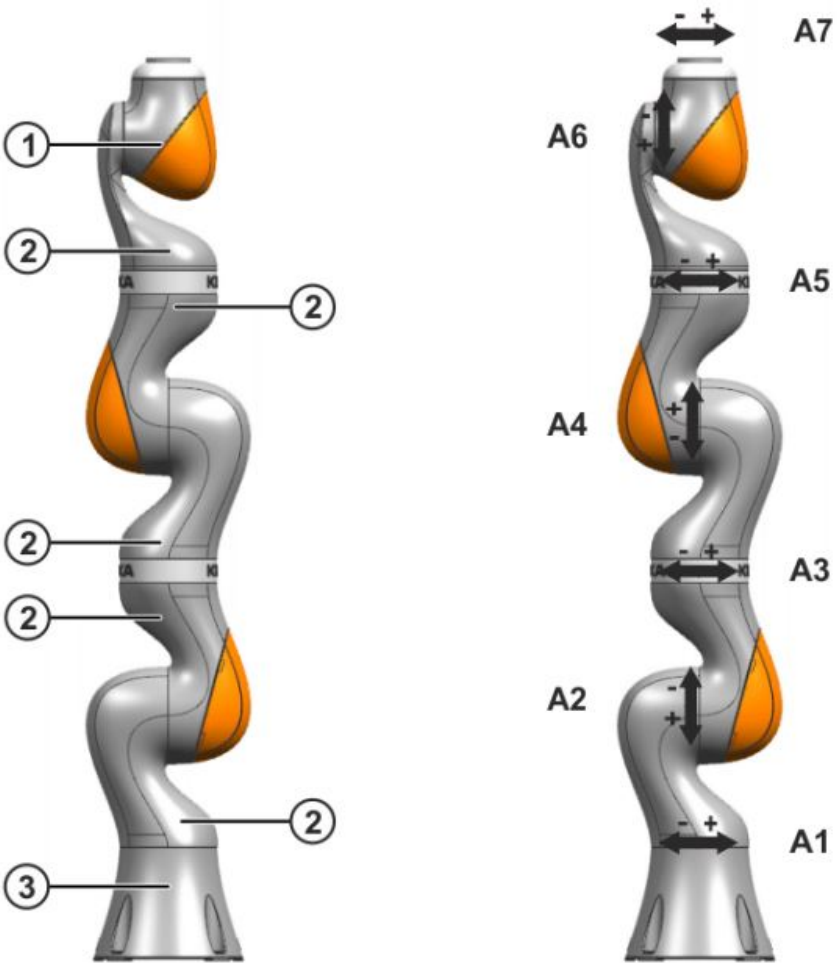


Figura 1.2: Conjuntos principales y ejes de robot [2]

1. Muñeca en linea.
2. Módulo de unión.
3. Bastidor base.

Instalaciones eléctricas: Las instalaciones eléctricas incluyen todos los cables de alimentación y control de los motores de los ejes A1 (J1) a A7 (J7). Todas las conexiones en los motores son conexiones macho y hembra. Todo el cableado se enruta internamente en el robot.

Ventajas principales:

- Tiempo de reacción mínimo
- Con capacidad de Aprendizaje
- Sensitivo
- Autónomo

El LBR iiwa es ligero y recomendable para tareas de montaje sensibles en donde se podrá eliminar las vallas protectoras del espacio de trabajo y allanar el camino a la colaboración entre personas y robots [1].

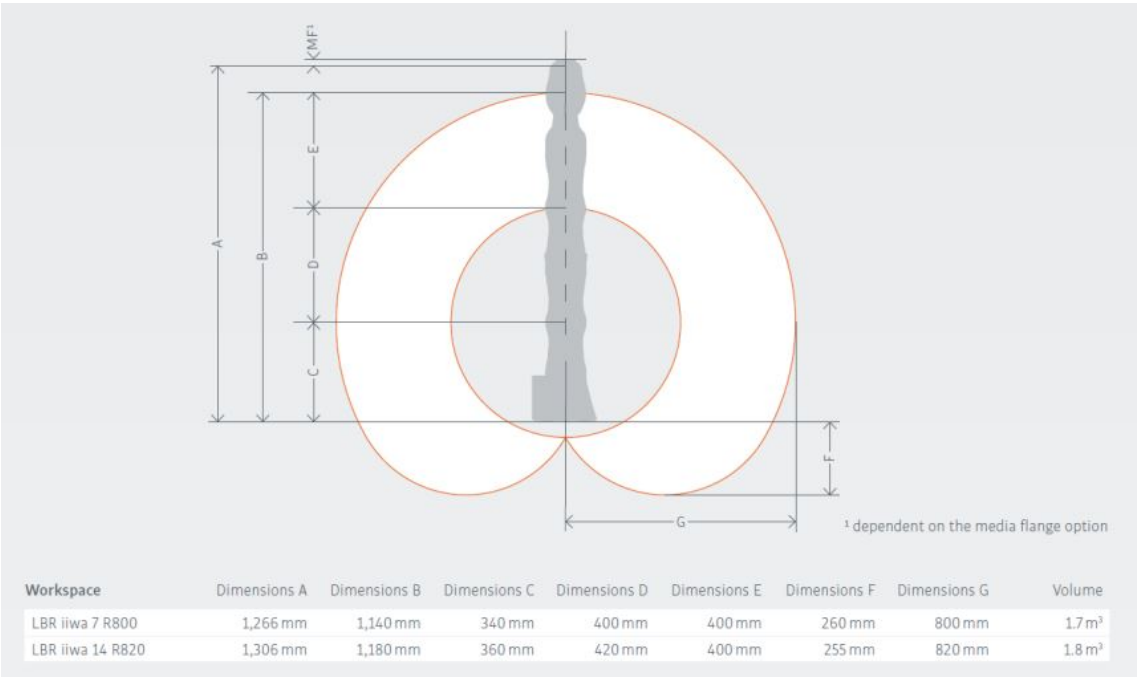
Figura 1.3: Entorno de trabajo colaborativo [1]

El LBR iiwa tiene un uso extendido como asistente en la producción debido a su gran versatilidad de trabajar de forma colaborativa con el operador, lo hace óptimo para actividades que requieren mucha precisión y que se repiten muchas veces al día. Al final se logra una productividad optimizada.

Tal como se muestra en la figura 1.3 y 1.4, el robot puede trabajar en un entorno colaborativo en las tareas exclusivas en las que se requiera asistencia o en todo el proceso de desearlo.

Figura 1.4: Robot KUKA LBR iiwa R14 820 [1].

En las figuras 1.5 y 1.6 se muestran las especificaciones generales en cuanto a dimensiones, peso, limitación de los actuadores, conexión del suministro eléctrico y más.



Workspace	Dimensions A	Dimensions B	Dimensions C	Dimensions D	Dimensions E	Dimensions F	Dimensions G	Volume
LBR iiwa 7 R800	1,266 mm	1,140 mm	340 mm	400 mm	400 mm	260 mm	800 mm	1.7 m³
LBR iiwa 14 R820	1,306 mm	1,180 mm	360 mm	420 mm	400 mm	255 mm	820 mm	1.8 m³

Figura 1.5: Dimensiones del LBR iiwa.[1]

Este conjunto de especificaciones geométricas y de limitación de los ejes tanto en posición como velocidad y torque serán empleadas a lo largo del trabajo en las diferentes etapas.

LBR iiwa	LBR iiwa 7 R800	LBR iiwa 14 R820
Rated payload	7 kg	14 kg
Number of axes	7	7
Wrist variant	In-line wrist	In-line wrist
Mounting flange A7	DIN ISO 9409-1-A50	DIN ISO 9409-1-A50
Installation position	any	any
Positioning accuracy (ISO 9283)	± 0.1 mm	± 0.1 mm
Axis-specific torque accuracy	± 2 %	± 2 %
Weight	23.9 kg	29.9 kg
Protection rating	IP 54	IP 54

Axis data / Range of motion		LBR iiwa 7 kg Maximum torque	LBR iiwa 7 kg Maximum velocity	LBR iiwa 14 kg Maximum torque	LBR iiwa 14 kg Maximum velocity
Axis 1 (A1)	± 170°	176 Nm	98°/s	320 Nm	85°/s
Axis 2 (A2)	± 120°	176 Nm	98°/s	320 Nm	85°/s
Axis 3 (A3)	± 170°	110 Nm	100°/s	176 Nm	100°/s
Axis 4 (A4)	± 120°	110 Nm	130°/s	176 Nm	75°/s
Axis 5 (A5)	± 170°	110 Nm	140°/s	110 Nm	130°/s
Axis 6 (A6)	± 120°	40 Nm	180°/s	40 Nm	135°/s
Axis 7 (A7)	± 175°	40 Nm	180°/s	40 Nm	135°/s

Programmable Cartesian stiffness		
Min. (X, Y, Z)	0.0 N/m	0.0 N/m
Max. (X, Y, Z)	5,000 N/m	5,000 N/m
Min. (A, B, C)	0.0 N/rad	0.0 N/rad
Max. (A, B, C)	300 Nm/rad	300 Nm/rad

KUKA Sunrise Cabinet	
Processor	Quad-core processor
Hard drive	SSD
Interfaces	USB, EtherNet, DVI-I
Protection rating	IP20
Dimensions (D x W x H)	500 mm x 483 mm x 190 mm
Weight	23 kg

Power supply connection	
Rated supply voltage	AC 110 V to 230 V
Permissible tolerance of rated voltage	± 10 %
Mains frequency	50 Hz ± 1 Hz or 60 Hz ± 1 Hz
Mains-side fusing	2 x 16 A slow-blowing

30,000 operating hours

Figura 1.6: Ficha técnica del LBR iiwa[1]

1.4. Organización del Trabajo de Fin de Máster

El contenido de este trabajo de fin de máster se ha organizado según las siguientes secciones.

En la sección 2, Modelado de Sistemas Mecánicos. Se describen los métodos utilizados para el Modelado Cinemático y Dinámico de sistemas mecánicos. Se comenta la manipulación simbólica empleada para las ecuaciones de la simulación directa, ecuaciones de estimación de parámetros y se construyen las Ecuaciones Dinámicas de Movimiento del Robot KUKA LBR iiwa R14 820 en las que se basa este Trabajo de Fin de Máster.

En la sección 3, Reducción del Modelo Dinámico. Se describe la manera en que se lleva a cabo la reducción del modelo dinámico a parámetros base a partir de la ecuación de la dinámica inversa. Se explica de forma muy general pero suficiente para los alcances de este trabajo el método seleccionado para realizar dicha reducción y la forma en que se llevó a cabo su implementación en el robot de estudio.

La sección 4 está dedicada al Diseño de Experimentos. En él se describen los dos criterios utilizados para la optimización de la trayectoria, así como la parametrización de trayectorias con funciones que sean permanentemente excitantes y derivables. La manera de abordar las restricciones presentes en los actuadores del mecanismo. Se muestra en qué manera se realiza la optimización de la trayectoria a través de las funciones especiales de MATLAB y se implementa la parametrización y optimización de la trayectoria en el robot LBR iiwa.
En un apartado especial de la sección 4 se detalla la simulación del experimento, creación de un modelo virtual experimental y el conjunto de consideraciones y herramientas empleadas para lograrlo.

En la sección 5 se describe el proceso de validación, se llevan a cabo las simulaciones para sustituir los experimentos. La estimación de los parámetros y el vector de pares de actuación.
De cara a verificar la calidad de los resultados, se calculan los residuos y se evalúan en términos de la desviación típica de la estimación de los parámetros y en términos del error de estimación de las fuerzas externas.
Los intervalos de confianza del vector de pares de actuación estimado y el rango de error de los parámetros base estimados son calculados y visualizados de forma previa a la aplicación de la validación cruzada al modelo y las estimaciones.

En la sección 6 se presentan el conjunto de Conclusiones obtenidas de este trabajo de fin de máster.

Por último, en las sección 7 se detalla la Bibliografía, Referencias y al final los anexos del documento.

Cabe mencionar que de las secciones anteriores, a excepción de la 1, 6 y 7 el resto se estructuraron de la siguiente manera:

- Marco teórico o referencial del tema.
- Implementación de lo mencionado en dicha sección en el robot de estudio.
- Referencias al código de MATLAB y otras herramientas utilizadas por el autor.

De esta forma se facilitará la compresión lectora y mantendrá homogeneidad estructural en el documento según el autor.

2. MODELADO DE SISTEMAS MECÁNICOS

En esta sección se detallará el procedimiento empleado para el modelado del robot. Hemos aplicado los conocimientos y métodos aprendidos en la asignatura de Dinámica de sistemas Multicuerpo del máster.

Este modelado se divide en dos sub-secciones, la primera sub-sección trata del modelado cinemático que consiste en construir un modelo que represente la cinemática del sistema. Es decir que se cumplan las posiciones de los sólidos involucrados, la definición de las juntas cinemáticas, la relación del movimiento entre los sólidos, la identificación de los grados de liberta del sistema y la realización de una simulación que permita verificar que el modelo satisface todos los aspectos antes mencionados.

La segunda sub-sección aborda el modelado dinámico del sistema, en el que se identifican y caracterizan los elementos que afectan el comportamiento de la dinámica del sistema como son los efectos debidos a los parámetros inerciales como ser la masa, primer momento, tensor de inercia que son inherentes a los sólidos.

Los efectos debidos a las acciones constitutivas, únicamente el efecto de la fricción entre los sólidos, acciones de inercia, gravedad y externas que no provienen de la caracterización del sistema, es decir aquellas aplicadas por los actuadores para generar el movimiento.

El principio de formulación seleccionado para obtener el conjunto de ecuaciones dinámicas del modelo analítico del sistema es el Principio de las Potencias Virtuales. Esta formulación plantea que, sí se cumplen ciertas condiciones *el modelo será lineal en los parámetros dinámicos*, es decir los parámetros inerciales y de fricción [3].

Una vez caracterizado el sistema por medio de torsores que representen los efectos de las acciones antes mencionadas y seleccionando el principio de formulación que convenientemente nos permite plantearlo como un sumatorio de productos de torsores por velocidades virtuales para obtener las ecuaciones dinámicas del sistema, se procede al cálculo de las ecuaciones dinámicas de movimiento.

Una vez obtenidas las ecuaciones dinámicas, es necesario llevar a cabo simulaciones que nos permitan visualizar el verdadero comportamiento dinámico del modelo construido y ser capaces de identificar de forma intuitiva que lo que se muestra en la simulación está cerca de lo que se espera de las acciones consideradas en el modelo, por ejemplo debido a la gravedad, inercia y fuerzas externas.

Esta simulación se logra resolviendo el problema de la dinámica directa planteado y utilizando el método de Euler mejorado para generar la simulación a lo largo del tiempo deseado [4].
Todo lo antes mencionado se ha implementado en MATLAB, empleando como plantilla uno de los ejemplos explicados en la asignatura DSM, y lógicamente empleando la misma librería.

Con respecto a lo que al código se refiere, para todo el modelado del sistema se ha empleado como herramienta principal la librería de 3D_MEC_MATLAB creada por el Dr. Javier Ros [5]. Esta librería permite de forma completa el modelado de cualquier sistema mecánico. Está formada por un conjunto de funciones que se pueden clasificar principalmente en simbólicas y numéricas. Distinción importante que se destaca debido a que la formulación simbólica es la que nos permite obtener el conjunto de ecuaciones dinámicas requeridas para llevar a cabo la estimación de parámetros y todo lo que ello implique. Y la formulación numérica es la encargada de permitirnos visualizar tanto el modelo cinemático como el modelo dinámico construido, así como la obtención de gráficas que nos permiten conocer el comportamiento del sistema.

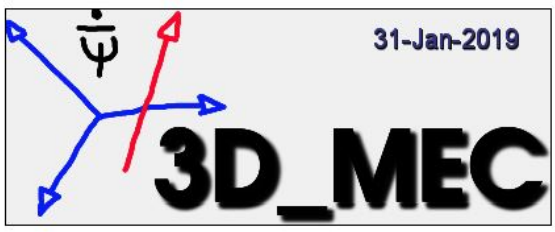


Figura 2.1: Librería 3D MEC[5]

2.1. Modelado Cinemático

En esta sub-sección se aborda el modelado cinemático, en el cual se establecen los movimientos permitidos del sistema, es decir se construye el modelo analítico que cumpla las condiciones a las que se encuentra sometido el mecanismo, ya sean grados de libertad, restricciones cinemáticas, de velocidad y aceleraciones.

Dado que el enfoque seleccionado es analítico, viene bien trabajar de forma directa en un espacio vectorial construyendo vectores, que serán manipulados en un espacio cartesiano.

Para llevar a cabo lo antes mencionado se requiere de la definición de un conjunto de elementos básicos que permitan una distribución espacial de los sólidos involucrados como pueden ser puntos de interés (en este caso las posiciones de las juntas cinemáticas), bases de referencia para cada sólido, transformación de bases debido a los movimientos y la exportación de la geometría de cada uno de los sólidos.

La parametrización trata la distinción entre las coordenadas variables del sistema, que se conocen como coordenadas generalizadas, que son aquellas que varían a lo largo del tiempo en el sistema normalmente asociadas a un grado de libertad, y aquellas coordenadas constantes que se denominan parámetros y que normalmente se asocian a las longitudes y orientaciones fijas de un sólido.

Una vez que se tienen definidos los aspectos antes mencionados sólo queda realizar el ensamblaje del sistema, y para ello es necesario resolver el problema de la posición, velocidad y aceleración que se comentarán en el siguiente apartado

La cinemática del robot es abordada con todo detalle en la sub-sección 2.4 de este documento.

2.1.1. Descripción de cómo se calculan las posiciones, velocidades y aceleraciones

La cinemática es la rama de la mecánica que nos permite describir el movimiento de los sólidos sin considerar las causas de su origen. Para esa descripción es necesario obtener un conjunto de posiciones, velocidades y aceleraciones de los puntos de interés.

Como se ha mencionado en la introducción de la presente sección, el enfoque que se aborda es analítico, gracias a la definición de la posición del sistema en términos de coordenadas constantes en el tiempo, que llamaremos parámetros y coordenadas variables que pueden variar con el tiempo, denominadas coordenadas generalizadas [4], nos centraremos en estas últimas y se describirá la manera empleada para obtener a partir de ellas las velocidades y aceleraciones generalizadas.

Una vez realizada la parametrización antes mencionada, podemos definir las coordenadas generalizadas como aquellos movimientos permitidos de traslación o rotación a los que son sometidos los sólidos del sistema.

Es decir aquellas que pueden llegar a variar con el tiempo, por lo que de forma directa, aplicando la derivada a esa variable es posible obtener las velocidades generalizadas y nuevamente aplicando derivación con respecto al tiempo es posible obtener las aceleraciones generalizadas.

Las velocidades y aceleraciones calculadas en los puntos de interés, o en los objetos son relativas al marco de referencia u observador. Por lo que podemos definir que para representar matemáticamente el marco de referencia será necesario en principio definir previamente la pareja (Punto, Base). El punto hace referencia al “Lugar del espacio desde donde se observa” y la base a la “orientación en la que se proyectan los vectores” de movimiento [4].

Cabe resaltar que en el presente modelo no se han utilizado Ecuaciones Cinemáticas de enlace. No obstante, para interés del lector son explicadas en la siguiente referencia [4].

2.2. Modelado Dinámico

En esta sección se aborda el estudio del modelado dinámico del sistema desde una visión analítica empleando nuevamente vectores que representarán las magnitudes físicas involucradas.

El objetivo del modelo dinámico que se pretende construir es servir para la identificación del propio sistema que representa. Una vez que hemos construido nuestro modelo dinámico, este nos permitirá estimar un conjunto de parámetros desconocidos a priori, de manera que permita que el modelo sea capaz de predecir el comportamiento dinámico del sistema [6, pág 67].

Estos parámetros mencionados los nombraremos *parámetros dinámicos*, los cuales a la vez se clasifican según su naturaleza en *parámetros inerciales* y *parámetros de fricción*.

Los parámetros inerciales son aquellos que están relacionados con los propios sólidos, como pueden ser las masas de cada sólido, las 3 componentes de la posición a su centro de gravedad en el espacio cartesiano y las 6 componentes diferentes del tensor de inercia formado por sus 3 componentes principales y 3 productos de inercia.

Los parámetros de fricción, son los coeficientes de fricción existentes entre dos sólidos en movimiento debido a la resistencia que se presenta por la rugosidad de ambas superficies y dependen del modelo de fricción seleccionado.

Para llegar a cumplir los objetivos propuestos en este trabajo es necesario ser capaz de escribir las ecuaciones dinámicas obtenidas del modelo construido, de tal forma que estas *sean lineales en los parámetros inerciales y de fricción*. Para que lo antes mencionado sea posible, respecto a los parámetros inerciales es necesario cumplir las siguientes condiciones [6, pág 67-68]:

- La información de la posición del centro de gravedad de cada sólido tendrá que estimarse a partir de los 3 primeros momentos de inercia, que no es más que el producto de la masa del sólido por sus tres coordenadas de posición al centro de gravedad. Luego una vez estimada la masa es podrá calcularse la posición de dicho centro de gravedad dividiendo entre la masa ya estimada de dicho sólido.
- La referencia en la que se defina el tensor de inercia del sólido, deberá pertenecer al sólido mismo. Por lo mencionado en la condición anterior, el único punto en el que no se puede definir la referencia inercial es el centro de gravedad del propio sólido, ya que es uno de los parámetros a estimar, y es desconocido a priori.

Ahora los parámetros inerciales de cada sólido que se deberán estimar son los siguientes:

$$(m, m_x, m_y, m_z, I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz}, I_{zz}) \tag{2.1}$$

donde m representa la masa del sólido, (m_x, m_y, m_z) representan el producto de la masa por la posición al centro de gravedad respecto a una referencia solidaria al sólido e (I_{xx}, I_{yy}, I_{zz}) representan los momentos de inercia y (I_{xy}, I_{xz}, I_{yz}) los tres productos de inercia con respecto a la misma referencia.

Respecto a los modelos de fricción en el presente trabajo se han considerado dos, el modelo de *Fricción Viscosa* ($\phi^F_{k,v}$) y el modelo de *Fricción de Coulumb* ($\phi^F_{k,c}$), ambos son capaces de definirse *de manera lineal respecto a los parámetros de fricción*, punto fundamental al momento de calcular las ecuaciones dinámicas del modelo.

Tanto los parámetros inerciales como los parámetros de fricción se agrupan en el conjunto que llamaremos vector de los parámetros dinámicos que representaremos en el trabajo como ϕ.

$$\phi = (m, m_x, m_y, m_z, I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz}, I_{zz}, \phi^F_{k,v}, \phi^F_{k,c}) \tag{2.2}$$

En las sub-secciones siguientes se definen los torsores, se caracterizan según la naturaleza de la que provienen y aplican al mecanismo de estudio. Se selecciona el formalismo a utilizar para la obtención de las ecuaciones dinámicas, sus condiciones y ventajas de utilizar dicho formalismo. Así como lo pertinente sobre la simulación dinámica del modelo.

Una sub-sección se ha dedicado a la manipulación simbólica, la manera de representarlas y el uso de las funciones especiales de MATLAB [7] que como se detallará en la misma toma un papel fundamental a lo largo de este trabajo.

2.2.1. Torsores

Los torsores (Wrench) son un concepto mecánico, que permite la agrupación del conjunto de fuerzas y momentos relativos a una acción concreta. Para la definición de dichos torsores es necesario identificar el punto de aplicación de las fuerzas de cada sólido y el punto con respecto al que se considera el momento aplicado. Estas fuerzas pueden ser de origen externo al sistema o interno [4, pág. 31].

Estos Torsores actúan sobre todos los sólidos y se caracterizan según la naturaleza de que provienen y los efectos sobre el mecanismo, entre los que encontramos los torsores de la inercia, gravedad, externos y de fricción que se describirán en los apartados siguientes.

Existen los torsores correspondientes a las restricciones de movimiento del sistema denominados torsores de restricción. Estos se encuentran asociados con la determinación de las fuerzas y momentos de enlace en los pares cinemáticos, es decir aquellas fuerzas y momentos necesarios para que el conjunto de sólidos permanezcan juntos y solamente realice los movimientos permitidos [4].

Los torsores de restricción no se han planteado es este trabajo, dado que no es del interés del autor obtener dichas fuerzas y momentos de enlace. No obstante, las restricciones de movimiento del sistema van implícitas en la parametrización realizada en donde se han definido tanto las coordenadas generalizadas como los movimientos permitidos.

El empleo de los torsores es útil tanto con la formulación de Newton-Euler como en la de Potencias Virtuales. Normalmente son aplicados para obtener el conjunto de las ecuaciones dinámicas del modelo dinámico como se mencionó en las secciones anteriores.

En los apartados siguientes se caracterizan los torsores considerados en el sistema, los cuales contienen la pareja de fuerza y momento.

Torsores de Inercia:
La acción de la inercia sobre todos los sólidos la podemos definir de la siguiente manera [3, pág. 29-34]:

$$\mathcal{F}_i = -m_i a(P_i) \tag{2.3}$$

$$\mathcal{M}_i^{P_i} = -I_{P_i}(Sol_i)\alpha(Sol_i) - m_i\overline{P_iG_i} \times a(P_i) \tag{2.4}$$

Torsores de Gravedad
La acción de la gravedad sobre todos los sólidos la podemos definir de la siguiente manera [3, pág. 29-34]:

$$F_g = m_i g \tag{2.5}$$

$$M_g^{P_i} = m_i\overline{P_iG_i} \times g \tag{2.6}$$

Torsores externos
La acción de las fuerzas y momentos externos sobre todos los sólidos la podemos definir de la siguiente manera [3, pág. 29-34]:

$$F_{e,i} = F_e \tag{2.7}$$

$$M_g^{P_i} = \overline{P_iQ_i} \times F_e + M_e \tag{2.8}$$

Torsores de fricción viscosa
La acción de las fuerzas y momentos por fricción viscosa sobre todos los sólidos la podemos definir de la siguiente manera [3, pág. 29-34]:

$$F_{v,i} = -\phi_{k,v}^F \dot{q}_{k,s} v_k \tag{2.9}$$

$$M_{v,i}^{P_i} = -\overline{P_iQ_i} \times F_{v,i} - \phi_{k,v}^M \dot{q}_{k,\theta} w_k \tag{2.10}$$

Torsores de Fricción de Coulomb

La acción de las fuerzas y momentos por fricción de coulomb sobre todos los sólidos la podemos definir de la siguiente manera [3, pág. 29-34]:

$$F_{c,i} = -\phi^{F}_{k,c} F_c sign(\dot{q}_{k,s}) u_k \tag{2.11}$$

$$M^{P_i}_{c,i} = -\overline{P_i Q_i} \times F_{c,i} - \phi^{M}_{k,c} M_c sign(\dot{q}_{k,\theta}) u_k \tag{2.12}$$

De las expresiones anteriores, a excepción de los torsores externos que se definen directamente en las ecuaciones 2.7 y 2.8 el resto de torsores son lineales en los parámetros dinámicos. Las fuerzas en los torsores se visualizan como el producto entre un parámetro dinámico 2.2 y una magnitud cinemática que no depende ni de los parámetros de inercia ni de fricción.

Para el caso de los momentos en los torsores, se observa que están formados por dos monomios. para el primer término tendremos magnitudes cinemáticas o fuerzas que ya sabemos que son lineal en los parámetros dinámicos y el segundo término es poco más complejo de visualizarlo, la explicación del desarrollo de lo mencionado se encuentra en la siguiente referencia [3, pág. 30-32].

Una vez definidos todos los torsores de interés en las ecuaciones 2.5 a la 2.12, y verificar que son lineales en los parámetros dinámicos se procede a aplicar el formalismo seleccionado para la determinación de las ecuaciones dinámicas de movimiento.

2.2.2. Ecuaciones Dinámicas

El conjunto de ecuaciones dinámicas se pueden determinar empleando dos formalismos conocidos, el de Newton Euler y el de las Potencias Virtuales. Aunque ambos permiten escribir las ecuaciones dinámicas lineales en los parámetros, cumpliendo ciertas condiciones, el autor ha seleccionado el Principio de las Potencias Virtuales [4, pág. 37].

Previo a la definición del formalismo seleccionado conviene introducir brevemente el concepto de Rotor (Twist), al vector que contiene el par de velocidad angular de un cuerpo y velocidad de uno de sus puntos, para mayor información ver referencia [4].

Cabe destacar que tanto los Torsores como los Rotores, son formados por dos vectores 3D, Torsores (Fuerza y Momento) y Rotores (Velocidad angular y Velocidad Lineal) lo que resulta vectores 6D (6 dimensiones) para los Torsores y Rotores.

Principio de las Potencias Virtuales

El principio de las potencias virtuales se enuncia como la suma de la potencia virtual de las acciones actuantes sobre un sistema es igual a cero. Que, no es otra cosa que la versión analítica de la segunda Ley de Newton [3, pág. 25].

La potencia virtual de las acciones de enlace actuantes sobre el sistema es cero para velocidades virtuales compatibles con los enlaces. La propiedad anterior, no es otra cosa que la versión analítica de la tercera ley de Newton, puede ser utilizada para la caracterización analítica de las acciones de enlace.

Las ecuaciones dinámicas a través del Principio de las Potencias Virtuales, pueden escribirse de la siguiente manera [3, pág. 25]:

$$\Gamma_k = \sum_{i=1}^{N_{sol}} (\mathcal{F}_i \mathcal{V}^*_{ik} + \mathcal{M}_i \Omega^*_{ik}) + \sum_{i=1}^{N_{sol}} \sum_{j=1}^{N_{Acc}} (F_{ij} V^*_{ik} + M_{ij} \Omega^*_{ik}) = 0 \tag{2.13}$$

donde F_{ij} y M_{ij} representan las fuerzas y momentos exteriores (acciones) j-ésimos aplicados sobre el sólido i, N_{Sol} representa el número de sólidos del sistema, y N_{Acc} el número de acciones que se ejercen sobre cada sólido. Los vectores V^*_{ik} o Ω^*_{ik}, representan los vectores por los que multiplicamos las fuerzas ejercidas sobre el sólido i que nos darán la ecuación dinámica.

Si a los vectores V^* y Ω^* se les define el significado de las velocidades virtuales, de forma que el producto resultantes son las Potencias Virtuales, de donde toman su nombre.

El Principio de potencias virtuales aplicado a un sistema multicuerpo quedaría de la siguiente manera [3, pág. 25]:

$$\sum_{i=1}^{N_{sol}} \dot{\mathcal{W}}_i + \sum_{i=1}^{N_{sol}} \sum_{j=1}^{N_{Acc}} \dot{\mathcal{W}}_{ij} = 0 \tag{2.14}$$

donde $\dot{\mathcal{W}}_i$ es la potencia virtual del sólido i debido a su torsor de inercia, y $\dot{\mathcal{W}}_{ij}$ es la potencia virtual del sólido i debido al torsor de las fuerzas j aplicado sobre el sólido i.

2.2.3. Simulación Dinámica

En esta sub-sección se describe el método empleado para llevar a cabo la simulación dinámica.

El algoritmo usado es el método de Euler mejorado, que hace uso de la serie de Taylor con una aproximación de segundo orden ya que disponemos del valor de $\ddot{\mathbf{q}}$ para realizar la integración de $\mathbf{q}$. Quedando el algoritmo empleado de la siguiente manera [4, pág. 27]:

$$\mathbf{q} = \mathbf{q} + (\dot{q}\Delta t + \frac{1}{2}\ddot{q})\Delta t \tag{2.15}$$

donde Δt es el incremento de tiempo entre cada instante, q el valor de la posición calculado. El procedimiento es iterativo, y consiste en resolver el problema de la posición y velocidad del mecanismo en el instante inicial (problema de montaje ,t=0), obtención de las aceleraciones generalizadas resolviendo el problema de la dinámica directa, luego a partir de las aceleraciones calculadas se realiza una doble integración para obtener $\mathbf{q}$, utilizando la ecuación 2.15, se calcula $\dot{\mathbf{q}}$ y se pasa al instante siguiente (t=t+Δt), resolviendo el problema de la nueva posición y velocidad calculada y se repite nuevamente los pasos a partir de la aplicación de la dinámica directa.

2.3. Manipulación Simbólica

El enfoque del cálculo simbólico, es de relevante importancia ya que se emplea en múltiples ocasiones en este trabajo. Primero para el modelado del sistema que ya se menciono en al inicio de esta sección en donde la librería 3D_MEC_MATLAB [5] emplea el cálculo simbólico y numérico. Segundo, para el manejo, simplificación de las ecuaciones dinámicas, la construcción de los vectores y matrices del sistema. Y tercero, para la creación de las ecuaciones (series finitas de Fourier) empleadas para la excitación de la trayectoria y la optimización de la misma.

En la segunda y tercera ocasión mencionada, en el **main_simbolic** ver Anexo B se han creado funciones a través del matlabFuncion [7], las cuales son empleadas más adelante en este trabajo de forma muy recurrente para la optimización de los parámetros de trayectoria, para generar la estimación y simulación, dichas expresiones simbólicas son evaluadas de forma numérica y permiten dar solución numérica a lo solicitado.

En las secciones anteriores se ha mencionado el procedimiento empleado para la creación del modelado cinemático y dinámico del robot. De donde se obtiene la simulación para verificar la cinemática del robot y las ecuaciones dinámicas que rigen el comportamiento dinámico del robot empleando el Principio de las Potencias Virtuales, así cómo la comprobación del modelo a través de la simulación dinámica en varias posiciones.

Luego de haber obtenido las ecuaciones dinámicas se puede llevar a cabo el paso siguiente, que consiste en la obtención del modelo lineal en los parámetros dinámicos. Para ello se ha empleado la ecuación de la dinámica inversa, formada por las matriz de observación K para cada instante de tiempo $K(q_i, \dot{q}_i, \ddot{q}_i)$, el vector de parámetros dinámicos ϕ vistos en la ecuación 2.2, (parámetros inerciales $\phi_{inercia}$ y de fricción $\phi_{friccion}$), y el vector de pares de actuación que se obtendrá de forma experimental del robot en cada instante de tiempo τ_i.

$$K\phi = \left[K(q_i, \dot{q}_i, \ddot{q}_i)\right] \phi = \left\{\tau_i\right\} = Y \tag{2.16}$$

donde K es la matriz de observaciones, ϕ el vector de parámetros dinámicos y τ o Y el vector de los pares de los actuadores. Como se puede observar el sistema anterior es lineal en los parámetros dinámicos, y para realizar la estimación se requiere la evaluación de la matriz de observación K en los n instantes de tiempo del experimento, que resulta en la creación de la matriz de observación W. Dicha matriz es detallada en la sección 3.

2.3.1. Nombramiento de las variables

Desde el inicio del trabajo, por instrucciones del tutor se consideró el nombramiento de todas las variables con nombres simples, cortos y que presten idea de lo que representan.

Esto se debe principalmente a que el conjunto de ecuaciones dinámicas obtenido es muy extenso debido al número de sólidos involucrados y sus movimientos relativos. También, como se trabaja de forma simbólica el número de variables es mayor.

El nombramiento de los elementos, puntos, bases, sólidos, y demás se ha llevado a cabo empleando la letra inicial del elemento en mayúscula y con un sufijo numérico para indicar a que sólido, o grado de libertad (GDL) corresponde.

Para los parámetros geométricos, dinámicos y coordenadas generalizadas, se ha empleado la letra inicial del elemento en minúscula y se ha incluido un sufijo adicional al numérico en letras minúsculas que indicará la orientación en un espacio cartesiano cuando corresponda (Parámetros Inerciales).

Se han empleado los nombres siguientes:

- Puntos: P(i)
- Bases: B(i)
- Marcos de Referencia: F(i)
- Sólidos: S(i)
- Coordenadas generalizadas: t(i)

- Velocidades generalizadas: dt(i)
- Aceleraciones generalizadas: ddt(i)
- Parámetros geométricos: h(i), L(i)
- Parámetros inerciales: m(i),mx(i),my(i),mz(i),Ixx(i),Iyy(i),Izz(i),Ixy(i),Ixz,(i),Iyz(i)
- Parámetros de fricción: c(i),cc(i)

Cabe mencionar que a excepción del punto O y base del origen xyz, lo anterior se cumple. De cara al análisis dinámico, se menciona que ambos elementos no están presentes en las ecuaciones dinámicas.

Aun realizando los nombramientos mencionados las ecuaciones dinámicas son bastante extensas, por lo que se han guardado en archivo ".mat" para su manipulación.

2.3.2. Ecuaciones de simulación dinámica directa

Como se ha mencionado al inicio de la sub-sección, para realizar las simulación dinámica, es necesario definir de forma simbólica el formalismo empleado como se muestra en la ecuación 2.17, de la cual se obtienen las matriz de masa por medio del jacobiano de la ecuación respecto al vector de las aceleraciones generalizadas [4, pág. 33].

Es decir, resolver el problema de la dinámica directa de forma simbólica. Para luego evaluar de forma numérica la expresión resultante [5].

$$\mathbf{e}^{Dyn,VP} = \mathbf{M}_{\dot{q}\dot{q}}\ddot{q} - \mathbf{f}^{C}_{\dot{q}\varepsilon}\varepsilon - \delta = 0 \tag{2.17}$$

donde $e^{Dyn,VP}$ representa el conjunto de las ecuaciones dinámicas, $\mathbf{M}$ la matriz de masas, $\ddot{q}$ el vector de las aceleraciones generalizadas. Para el cálculo de la simulación dinámica directa empleando el formalismo de las Potencias Virtuales, el cálculo de la matriz de masa es regida por la ecuación 2.18.

$$\mathbf{M}_{\dot{q}\dot{q}}(q,t) = \frac{\partial \mathbf{e}^{Dyn,VP}}{\partial \ddot{\mathbf{q}}} \tag{2.18}$$

2.3.3. Ecuaciones para estimación de parámetros:

De la misma manera en que se han descrito las ecuaciones dinámicas en la ecuación 2.16, se puede obtener los valores de la matriz de observación K, para n instante de tiempo con el jacobiano del conjunto de ecuaciones dinámicas respecto a los parámetros dinámicos.

$$\mathbf{K}(q,\dot{q},\ddot{q},t) = \frac{\partial \mathbf{e}^{Dyn,VP}}{\partial \phi} \tag{2.19}$$

$$\tau(q,t) = K\phi - \mathbf{e}^{Dyn,VP} \tag{2.20}$$

donde K es la matriz de observaciones, $e^{Dyn,VP}$ representa el conjunto de las ecuaciones dinámicas, ϕ el vector de parámetros dinámicos y τ el vector de los pares de los actuadores.

En este trabajo se ha optado por determinar K, de forma diferente. En vez de emplear las derivadas parciales de las ecuaciones dinámicas respecto a los parámetros dinámicos como se observa en la ecuación 2.19, se utiliza un método de sustitución, representado en la ecuación 2.21. Dicha selección fue basada en que, él cambio *resulta en una reducción del costo computacional*, que se traduce en menos tiempo para la creación de la matriz de observación.

$$\mathbf{K}_i(q,\dot{q},\dot{q},t) = \mathbf{e}^{Dyn,VP}|\phi_{\mathbf{j}} = \mathbf{0},\ \ \phi_{\mathbf{i}} = \mathbf{1},\ \ \tau_{\mathbf{n}} = \mathbf{0} \tag{2.21}$$

donde ϕ_j representa todos los elementos del vector de parámetros con excepción del elemento ϕ_i. El procedimiento es descrito a continuación.

- Sustituir por cero todos los elementos del vector de parámetros dinámicos con excepción del í-ésimo parámetro.
- Sustituir por uno, el i-ésimo parámetro (equivalente a la derivada con respecto al parámetro).

- Sustituir por cero todos los elementos del vector de Fuerzas y momentos (que al ser término independiente su derivada respecto a los parámetros es cero).
- Construcción de la matriz K, mostrada en la ecuación 2.22, que de forma posterior será evaluada en cada instante de tiempo para la creación de la matriz de observaciones W.

$$K_{n_q \times n_\phi} \phi_{n_\phi \times 1} = \tau_{n_q \times 1} \tag{2.22}$$

donde K representa la matriz de observaciones en el instante n, ϕ el vector de parámetros dinámicos, n_q representa el número de coordenadas generalizadas, n_ϕ representa el número de parámetros inerciales del sistema y τ el vector de pares externos.

2.4. Modelado del Robot iiwa

En este sub-sección se describirá el procedimiento empleado para llevar a cabo la construcción del modelo cinemático del robot, la selección de variables y parámetros, la construcción de los puntos, marcos de referencia y bases asociados con los puntos de interés del robot. Así como lo necesario para la creación del modelo dinámico y poder simular el comportamiento del sistema con la aplicación de los diferentes torsores considerados y descritos en las secciones anteriores.

Para llevar a cabo el modelado completo se ha empleado la librería de 3D_MEC_MATLAB [5] a lo largo de la descripción se indicará la referencia al fragmento de código que representa dicho apartado.

2.4.1. Cinemática del Robot

El KUKA LBR iiwa es un robot en serie con $n = 7$ grados de libertad en rotaciones y sin grados de libertad de traslaciones. Su estructura es representada en la figura 2.2. Entre cada acople de sólidos se encuentran las cajas de engranajes, juntas, sensores de par y esta equipado con codificadores de posición.

Tanto la medición del par de cada motor, como la posición de dicha articulación son mediciones fundamentales de cara a realizar el experimento y la estimación de los parámetros dinámicos que tiene como finalidad este trabajo.

Para el robot KUKA LBR iiwa, se ha realizado la acotación de los sólidos mediante vectores de posición y cambio de base, conocimiento aplicado en la asignatura de Dinámica de Sistemas Multicuerpo del máster.

Acotar un sistema significa describir la posición, a lo largo del tiempo de los puntos asociados al conjunto del mecanismo [3].

La acotación de un sólido en el espacio requiere de 6 coordenadas que representen su posición y orientación. Así que para la representación de las coordenadas de posición se han utilizado el vector de posición del punto final (sujeta herramienta) y el punto central de cada una de las juntas (p_x, p_y, p_z) que corresponden a las coordenadas en x, y, z de un vector en el espacio cartesiano, valores conocidos y de interés ya que los puntos funcionan de unión entre dos sólidos, y hemos definido el movimiento de cada sólido respecto de otro sólido. Para las orientaciones, dado que en cada una de las juntas esta presente un grado de libertad de rotación, es posible utilizar puntos para la creación de las diferentes bases y marcos de referencia del modelo.

Las orientaciones son representadas por las diferentes bases creadas en donde se definen $(\gamma_x, \gamma_y, \gamma_z)$, que son a las componentes del vector de orientación en un espacio cartesiano, que corresponde a la orientación de un sólido respecto al otro, en este caso en particular, las orientaciones en rotación permitidas de cada sólido corresponden con los grados de libertad del mismo.

Como se ha mencionado anteriormente, esta definición y parametrización es de suma importancia ya que es fundamental para la definición del modelo, porque son el conjunto de variables que se manipulan para definir los movimientos permitidos en el robot y una elección incorrecta de la parametrización puede ser funcional pero complicar la obtención de las ecuaciones dinámicas, lo que se traduce en aumentar el coste computacional en el análisis y simulación del sistema.

Una vez realizada la acotación, estas coordenadas las clasificamos según sus características, si la coordenada representa una longitud o una orientación fija de un sólido respecto al otro se denomina

parámetros geométricos (constantes), si la coordenada representa una capacidad de movimiento entre los dos sólidos se denomina coordenada generalizada.

2.4.2. Selección de coordenadas generalizadas

En la sección anterior se comento que las Coordenadas Generalizadas las definimos como aquellas que nos representan el movimiento relativo de un sólido respecto al otro, que para el robot son representados por los giros generados por los motores que son los que introducen el movimiento al robot.

Las hemos denominado t_i, que representan los grados de libertad del robot, movimiento rotacional de cada uno de los motores, cabe destacar que en este robot en serie todas las coordenadas generalizadas seleccionadas son independientes. Luego a partir de las coordenadas generalizadas somos capaces de obtener las velocidad y aceleraciones generalizadas aplicando la primera y segunda derivada con respecto al tiempo.

En el caso de este robot de cadena abierta, no hay ecuaciones de enlace geométrico, y sabiendo que por la naturaleza de los movimientos las coordenadas generalizas son independientes y los grados de libertad del robot podemos comprobar:

$$m = n - n_{q_{ind}} \qquad m = 7 - 7 = 0 \tag{2.23}$$

donde m representa el número de ecuaciones de enlace geométrico, n los grados de libertad del robot y n_{qind} las coordenadas generalizadas independientes. Se ha verificado la no existencias de las ecuaciones de enlace geométrico con sus respectivas derivadas.

Para le robot de estudio se representan las coordenadas, velocidades y aceleraciones generalizadas empleadas.

$$q = \begin{bmatrix} t1 \\ t2 \\ t3 \\ t4 \\ t5 \\ t6 \\ t7 \end{bmatrix} \qquad \dot{q} = \begin{bmatrix} dt1 \\ dt2 \\ dt3 \\ dt4 \\ dt5 \\ dt6 \\ dt7 \end{bmatrix} \qquad \ddot{q} = \begin{bmatrix} ddt1 \\ ddt2 \\ ddt3 \\ ddt4 \\ ddt5 \\ ddt6 \\ ddt7 \end{bmatrix} \tag{2.24}$$

donde q representa las coordenadas generalizadas, $\dot{q}$ las velocidades generalizadas y $\ddot{q}$ las aceleraciones generalizadas.

En la figura 2.2, se muestra el robot en posición vertical identificando los sólidos del robot y los giros de las coordenadas generalizadas.

2.4.3. Selección de parámetros

Denominamos por parámetros a aquellas coordenadas que son constante a lo largo del tiempo, como longitudes de los sólidos y orientaciones fijas, la información de los valores de dichas constantes se extrajeron del manual de KUKA [2], así como las características mostradas en el apartado anterior.

El vector de parámetros geométricos del modelo esta formado de la siguiente manera:

$$param = \begin{bmatrix} h0 \\ h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ L6 \end{bmatrix} \tag{2.25}$$

donde h_i representa la componente en vertical en el espacio cartesiano de la unión de los sólidos, dicho de otra forma la “altura” de los sólidos del robot.

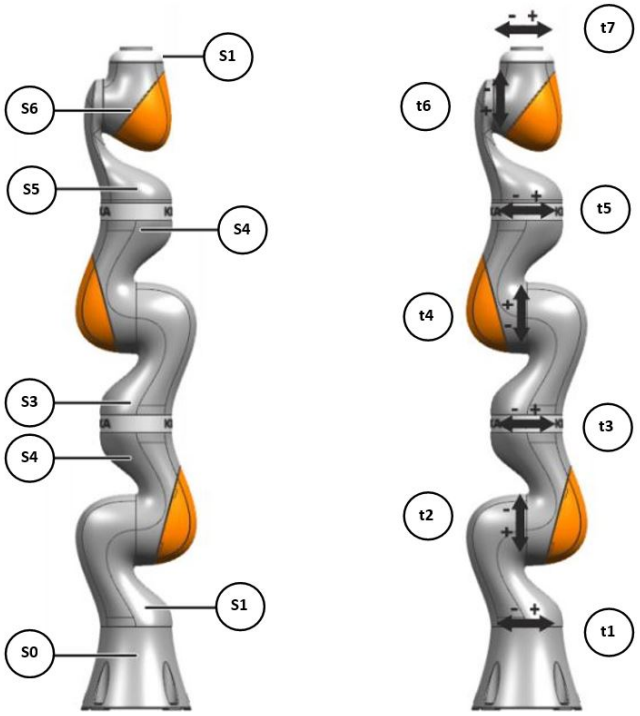


Figura 2.2: Sólidos del robot y grados de libertad

Como se menciono al inicio de esta sub-sección, el resto de posiciones y orientaciones que no se han incluido como coordenadas generalizadas ni en el vector de parámetros geométricos, son parámetros que toman el valor de cero, los cuales se toman en cuenta al momento de crear los puntos de interés.

A este conjunto de parámetros geométricos se incluye la gravedad que siempre es constante y será necesaria para el análisis dinámico del modelo.

2.4.4. Definición de los puntos, marcos de referencia y bases de interés

Dentro del análisis cinemático, se han realizado la selección de coordenadas generalizadas, parámetros geométricos, selección de los puntos de interés, marcos de referencia inercial para cada sólido y las bases para las rotaciones de Euler.

- Puntos

  Los puntos de interés, como se ha repetido en reiteradas ocasiones se han considerado las uniones entre los sólidos, (juntas cinemáticas), así como los puntos de la base inercial (suelo) del robot y el punto final (ubicación de la herramienta).

- Marcos de referencia

  Cabe mencionar que los marcos de referencia se han creado en cada una de las bases, solidarias a las juntas, los cuales sirven de referencia de un sólido al sólido siguiente.

- Bases

  Transformación de las bases

$$xyz \quad \xrightarrow{(e_3^{xyz}, t1)} \quad B1$$

$$B1 \quad \xrightarrow{(e_2^{B1}, t2)} \quad B2$$

$$B2 \quad \xrightarrow{(e_3^{B2}, t3)} \quad B3$$

$$B3 \quad \xrightarrow{(e_2^{B3}, t4)} \quad B4$$

$$B4 \xrightarrow{(e_3^{B4}, t5)} B5$$

$$B5 \xrightarrow{(e_2^{B5}, t6)} B6$$

$$B6 \xrightarrow{(e_3^{B6}, t7)} B7$$

donde B_i representa la Base del i-ésimo sólido, t_i la coordenadas generalizadas y e_j^{Bi} que el cambio se realiza desde la base i alrededor del eje j.

Bases solidarias para cada sólido

$$\begin{aligned} xyz &\equiv S0 \\ B1 &\equiv S1 \\ B2 &\equiv S2 \\ B3 &\equiv S3 \\ B4 &\equiv S4 \\ B5 &\equiv S5 \\ B6 &\equiv S6 \\ B7 &\equiv S7 \end{aligned}$$

donde S_i representa el sólido i-ésimo y B_i la base de referencia del sólido i-ésimo.

En la figura 2.3, se muestran el robot en posición vertical identificando los puntos, parámetros geométricos y bases creadas del modelo.

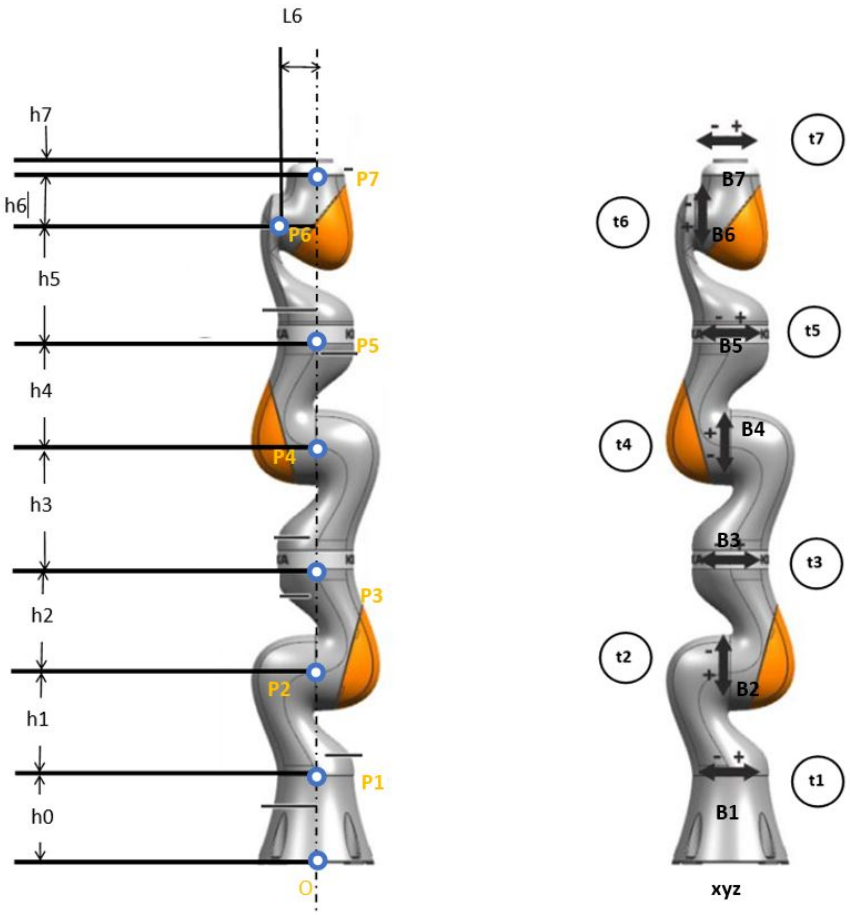


Figura 2.3: Entorno del robot [8]

Hay que hacer énfasis en que el modelo construido, presenta una diferencia a tomar en cuenta con respecto al modelo real ver figura 2.4 ver referencia [8], en que el modelo real la orientación del eje 4 es modelado de forma inversa al resto de los ejes paralelos, y en el modelo construido se ha supuesto la misma orientación para los giros de los ejes 2, 4 y 6, error que se atribuye el autor de

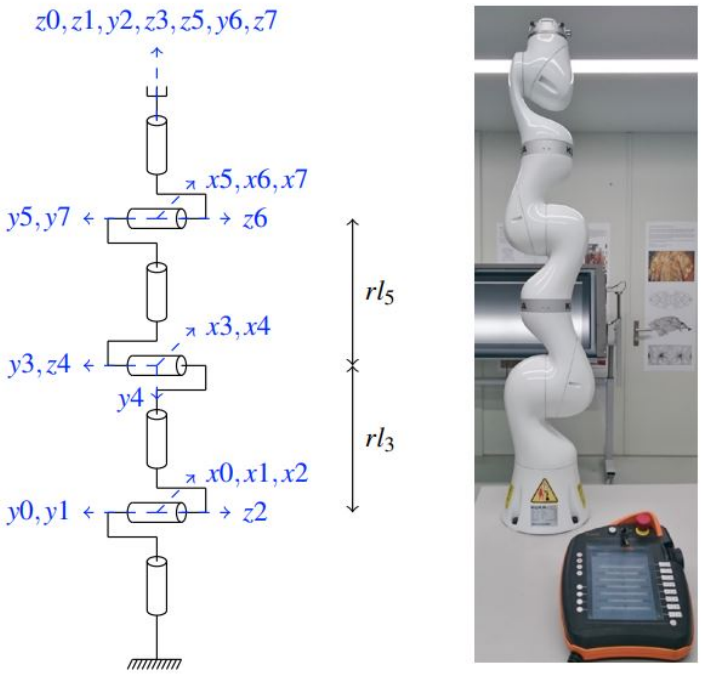


Figura 2.4: Configuración del robot con el eje 4 orientado correctamente.

este trabajo. En conversación con el tutor y para resolver el error cometido, se comenta que este cambio en la orientación del eje será tomado en cuenta al momento de comandar la trayectoria, en pocas palabras que hay que invertir el signo de los valores de posiciones y velocidades de ese eje.

2.4.5. Definición de los Sólidos

Los sólidos se han exportado desde un archivo CAD [9] en Solidworks a SCAD y STL para poder realizar las simulaciones cinemáticas empleando la librería 3D_MEC_MATLAB [5]. Aunque por ahora solo es necesario la geometría de los sólidos para su visualización, cabe mencionar que las propiedades de los sólidos utilizados fueron ingresadas por el autor. En el apartado del modelado dinámico se detallará lo relacionado a las propiedades inerciales de cada sólido y las consideraciones tomadas.

En la figura 2.5, se muestra un ejemplo del sólido base empleado, y sus respectivas propiedades físicas que serán necesarias para el análisis dinámico.

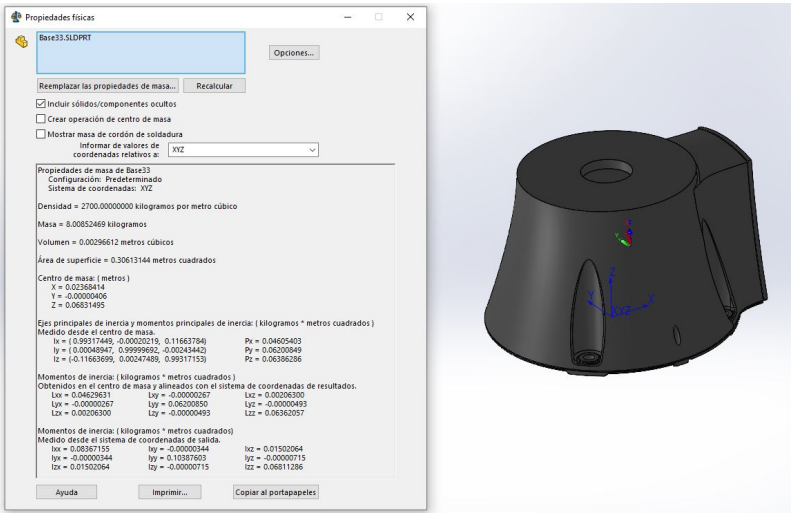


Figura 2.5: Propiedades de inercia de la Base del modelo

Inicialmente se había utilizado las propias funciones de las librería para realizar los cálculos de las propiedades físicas con una geometría simplificada, pero durante la realización del trabajo se hizo posible obtener un modelo del ensamblaje del cual se obtuvieron los sólidos y sus propiedades inerciales. Al realizar la exportación de Solidworks a SCAD para visualizarlos, no se exportaron como solidos por lo que no es posible implementar las funciones del cálculo de las propiedades de dicha librería.

En la figura 2.6 se muestra el modelo ensamblado utilizando el visualizador de la propia la librería.

Figura 2.6: Simulación Cinemática

Una vez construido el modelo cinemático se procede a realizar una simulación para verificar que los movimientos entre los sólidos son los esperados, en pocas palabras que se cumplan los grados libertad del robot.

2.4.6. Modelado Dinámico en el Robot

Una vez construido el modelo cinemático, se procede a la caracterización de los torsores implementando la librería 3D_MEC_MATLAB [5]. Utilizando las ecuaciones desde la 2.3 hasta la 2.12, y siguiendo el planteamiento vectorial, ingresando las fuerzas y momentos como vectores 3D y los torsores como vectores 6D.

Cabe destacar que la librería 3D_MEC_MATLAB permite de forma muy simple la definición de los torsores y rotores y por ende la aplicación del principio de las potencias virtuales de la ecuación 2.14 para obtener las ecuaciones dinámicas, resolver el problema de la dinámica directa y de forma posterior llevar a cabo la simulación del robot.

Para los torsores externos y de fricción, se menciona que no existen fuerzas involucradas, sólo momentos, ya que los sólidos no se desplazan de la junta, solamente rotan respecto al eje de la misma. Para el caso de la gravedad e inercia si se consideran las fuerzas aplicadas, estos torsores se crean casi automáticamente debido a las funciones de la librería identificando solamente el sólido.

Con el modelo dinámico del robot construido se han realizado un conjunto de simulaciones en distintas configuraciones para validar de forma inicial el modelo y evaluar su comportamiento antes los efectos dinámicos debido a la gravedad, fricción e inercia de los sólidos, como se muestra en la figura 2.7.

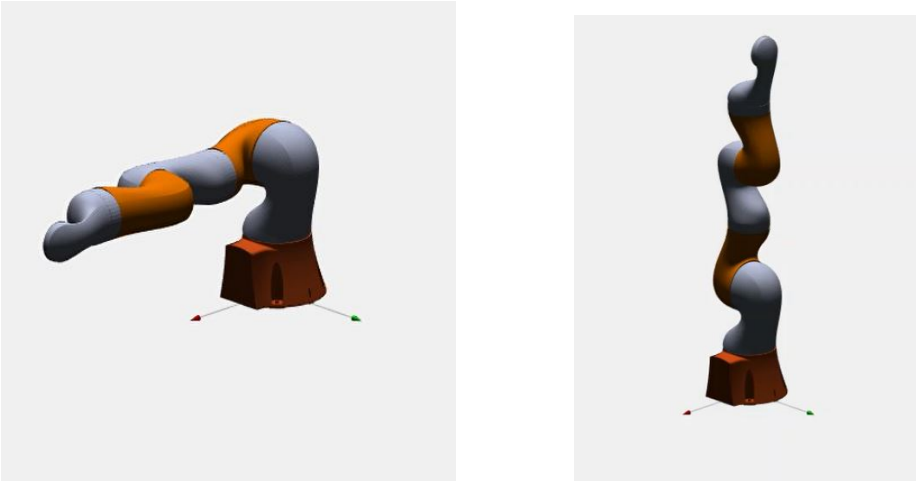
Es decir que el modelo del robot se comporte como se espera. Para las siguientes circunstancias, por ejemplo:

- El comportamiento del robot actuando con 3 sólidos

(a) Posición de la simulación 1 (b) Posición de la simulación 2 (vertical)

Figura 2.7: Simulaciones Dinámicas simplificadas con 3 sólidos

- El Comportamiento del robot actuando con 5 sólidos



(a) Posición de la simulación 1 (b) Posición de la simulación 2 (vertical)

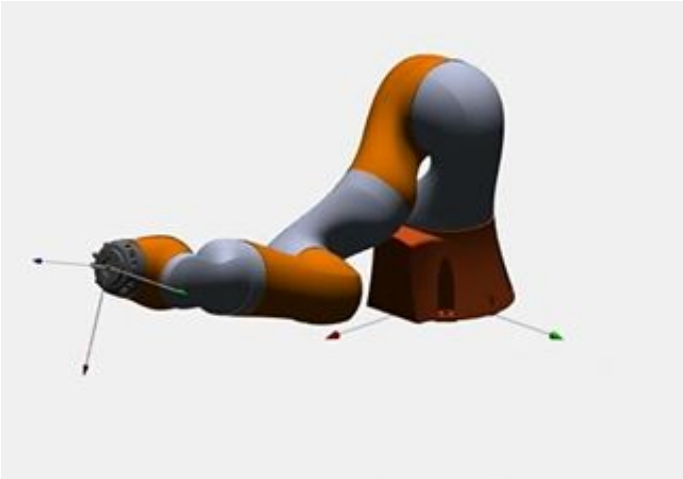
Figura 2.8: Simulaciones Dinámicas simplificadas con 5 sólidos

Las figuras 2.7 y 2.8 muestran las posiciones iniciales empleadas para llevar a cabo el conjunto de simulaciones dinámicas simplificadas empleando 3 sólidos inicialmente y luego 5 sólidos, de esta manera es más fácil identificar el comportamiento dinámico cuando interactúan menos sólidos.

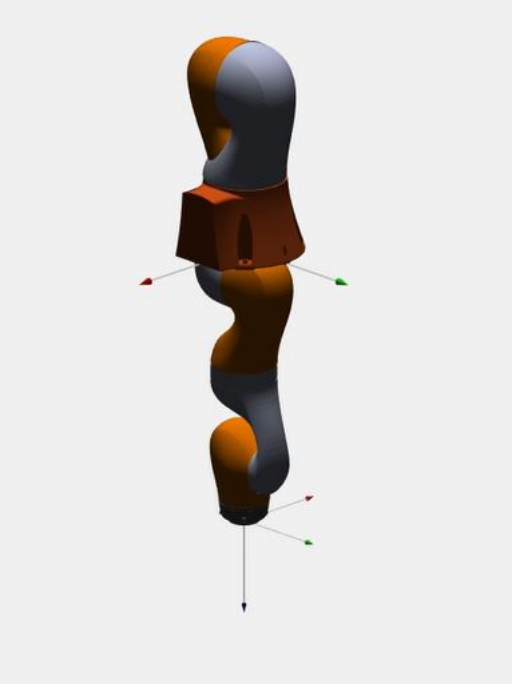
Una vez que estamos satisfechos y de acuerdo con el comportamiento dinámico del modelo de 3 y 5 sólidos se procede al modelado del robot con todos sus sólidos. La figura 2.9 muestra las posiciones iniciales de las simulaciones de todo el robot, en donde se espera que para la primera simulación el robot empiece a oscilar como un péndulo de muchos sólidos hasta llegar una posición de reposo como en la figura b. Y la segunda simulación se ha iniciado el robot desde esa posición para validar que no se mueve y haga cosas raras.

Se han realizado más simulaciones, pero con las indicadas acá es suficiente y se da por válido el modelo construido. Las simulaciones mostradas pueden ser proporcionadas como vídeos a solicitud del lector.

- El Comportamiento del robot actuando completo



(a) Posición de inicio 1



(b) Posición de inicio 2

Figura 2.9: Simulación Dinámica del Robot

3. REDUCCIÓN DEL MODELO DINÁMICO

De la sección anterior, del conjunto de las ecuaciones dinámicas se ha obtenido un modelo lineal ver ecuación 2.19, respecto a lo parámetros inerciales y de fricción (incógnitas) ver ecuación 2.2. Para la estimación de estos parámetros se requiere la evaluación de la matriz de observación W y el vector de las fuerzas y momentos obtenidos experimentalmente τ. Para garantizar la identificabilidad de los parámetros inerciales y de fricción es necesario realizar siempre una reducción (re-parametrización) del modelo, lo que resulta en un número menor de parámetros.

Esta reducción se puede interpretar matemáticamente, en que, al construir la matriz de observación W, presenta propiedades que hacen que la resolución sea no trivial y físicamente se refiere a que algunos de los parámetros no tienen influencia independiente en la dinámica del sistema, sino como combinación lineal de otros parámetros [4].

Para la construcción de la matriz de observación W, partimos de las ecuaciones de la dinámica inversa del sistema mecánico para n instantes, agrupando todas ellas, se obtiene W:

$$W\phi = \begin{bmatrix} K(q_1, \dot{q}_1, \ddot{q}_1) \\ \cdots \\ K(q_n, \dot{q}_n, \ddot{q}_n) \end{bmatrix} \phi = \begin{Bmatrix} \tau_1 \\ \cdots \\ \tau_n \end{Bmatrix} = Y \tag{3.1}$$

donde W representa la matriz de observación del sistema, ϕ los parámetros dinámicos, K y τ_i la matriz de observación y el vector de parees de los actuadores en un instante dado respectivamente e Y el vector de pares de los actuadores en todos los instantes.

De la reducción anterior, resulta que la matriz de observación W tiende a ser deficiente en rango (no es de rango máximo) por lo que existen combinaciones lineales entre sus columnas, lo que se traduce en que no es posible estimar todos los parámetros (o combinación de estos). Es decir, que es necesario una vez construida la matriz de observación W, emplear un método para obtener las combinaciones de parámetros que son estimables. Así como la nueva matriz de observación reducida del modelo W_b.

Cabe mencionar que esta matriz de observaciones W_b, será de rango máximo para el nuevo conjunto de parámetros (combinación de los parámetros inerciales y de fricción). Desde este punto se hace hincapié que cuando se habla de la matriz de observación y de los parámetros a estimar nos referimos a los del modelo reducido, es decir la matriz de observaciones de lo parámetros base W_b y los parámetros base ϕ_{Base}.

3.1. Reducción del modelo a parámetros base: Método QR

Uno de los mejores métodos numéricos para abordar este problema es el que utiliza la descomposición QR de la matriz W. Sea W la matriz de observación del sistema $W\phi = Y$ de tamaño $n \times p$ y sean Q,R y E tres matrices tal que:

$$\mathbf{W}_{n\times p}\mathbf{E}_{p\times p} = \mathbf{Q}_{n\times n}\mathbf{R}_{n\times p} \tag{3.2}$$

donde Q es una matriz cuadrada y ortonormal, R es una matriz triangular superior y E es una matriz de permutación (también ortonormal) de forma que los valores absolutos de los elementos de la diagonal de R están ordenados de forma decreciente [6, pág. 76].

La descomposición QR, por tanto reescribe la matriz W en términos de una matriz ortonormal, Q y una triangular superior.

El rango de W será igual al rango de R, $rank$(R) = $rank$(W). Si el rango de W es menor que el número de sus columnas, $rank(W) = r < p$, entonces los últimos $p - r$ elementos de la diagonal de R serán nulos y la matriz WE puede escribirse como:

$$\begin{aligned} WE &= [W_1, W_2] = [Q_1, Q_2] \\ &= [Q_1 R_1, Q_1 R_2] \end{aligned}$$

donde R_1 tiene rango máximo y es de tamaño r x r (es una matriz regular). Las matrices WE y Q las dividiremos en dos partes atendiendo al rango de R, de forma que W_1 y Q_1 serán las primeras r columnas de WE y Q, respectivamente. Como consecuencia:

$$W_1 = Q_1 R_1 \quad y \quad W_2 = Q_1 R_2$$

despejando Q_1 de la primera y sustituyendo en la segunda se deduce la relación:

$$W_2 = W1\beta$$

donde $\beta = R_1^{-1} R_2$. Esta operación es posible porque R_1 es cuadrada y de rango completo(y por tanto inversible) [6, pág. 76].

Reordenando el vector de parámetros ϕ mediante la matriz E.

$$\begin{aligned} W\phi = (WE)(E'\phi) &= [W_1, W_2] \begin{Bmatrix} \phi_1 \\ \phi_2 \end{Bmatrix} = [W_1, W_1\beta] \begin{Bmatrix} \phi_1 \\ \phi_2 \end{Bmatrix} \\ &= W_1[I_b, \beta] \begin{Bmatrix} \phi_1 \\ \phi_2 \end{Bmatrix} = W_1(\phi_1 + \beta\phi_2) = W_b\phi_b = Y \end{aligned} \tag{3.3}$$

siendo $W_1 = W_b$ representa los parámetros base del modelo reducido. Por tanto, un conjunto de columnas independientes de W constituyen la *Matriz de Observación del Modelo Reducido*, W_b, mientras que el vector de *Parámetros Base*, ϕ_b, puede escribirse como una combinación lineal de los parámetros originales [6, pág. 77]:

$$\phi_b = \phi_1 + \beta\phi_2 \tag{3.4}$$

donde ϕ_b representa los parámetros a estimar del modelo, ϕ_1 y ϕ_2 pertenecen a los parámetros inerciales del sistema.

El algoritmo para realizar la reducción de modelos mediante la descomposición QR sería la siguiente [6, pág. 77]:

- Calcular **Q,R** y **E** de la descomposición QR de **W**.
- Calcular el rango de **W**: r
- Determinar el número de columnas de W: p
- Extraer las matrices $\mathbf{R}_1$ y $\mathbf{R}_2$ de **R**.
- Calcular la matriz β
- Reordenar las columnas de **W** con **WE** y Extraer las primeras r columnas de **WE**
- Crear un vector simbólico de parámetros $\phi = \{p_1; p_2; ...; p_n\}$.
- Reordenar las filas de ϕ con $E'\phi$.
- Extraer ϕ_1 y ϕ_2 a partir de $E'\phi$.
- Determinar las expresiones simbólicas de los parámetros base ϕ_b

3.2. Implementación QR al robot

Una vez construida la matriz de observaciones de manera simbólica, a través de la función de matlabFunction de K y cómo se mencionó, se requiere identificar las columnas independientes de la matriz y la combinación lineal de los parámetros originales. Así como el vector de permutaciones de las columnas que se empleará para obtener la matriz de observaciones de los parámetros base de rango máximo.

Se ha utilizado la función QR de MATLAB para la implementación del método y con la llamada a la matlabFunction de K creada, evaluamos la función simbólica de forma numérica para obtener W de los parámetros base.

De la implementación del método QR en la matriz de observación original se han obtenido el conjunto de combinaciones lineales de los parámetros independientes. En el cuadro 3.1 se observa el conjunto de combinaciones lineales obtenidas.

El rango máximo de la matriz de observación de los parámetros base obtenido es de 57 variables. Que es consistente con los resultados obtenidos en el trabajo consultado ver referencia [10].

Lo mencionado en esta sección se ha llevado a cabo en MATLAB con la rutina **Main_Reduccion**. ver Anexo C.

Ixy7
Ixz7
Iyz7
Izz7
Ixx5 + Ixx6 + Ixx7 - Iyy5 - (65.45*m7)/10000 -(809*mz6)/5000
Iyz6
Ixz4
mx7
Ixy6
Ixz6
my7
Ixz5
Ixx5 + Ixx6 - Iyy5 + Iyy7 - (65.45*m7)/10000 - (809*mz6)/5000
Ixy4
Iyz5 - (431*my6)/2000
Iyy5 - Ixx5 + Izz5
Iyz4
Ixy5
Iyy5 - Ixx6 - Ixx5 + Iyy6
Iyz3 - (431*my4)/2000
Ixz3
mx6
Iyy4 + Iyy5 + (340*m5)/10000 + (4*m6)/25 + (4*m7)/25 + (369*mz5)/1000
Ixy3
mx4
(809*m7)/10000 + mz6 + mz7
Ixx3+Ixx4+Iyy2+Iyy5+0.0418*m3+0.176*m4+0.210*m5+841*m6/2500+0.336*m7+0.409*mz3+0.369*mz5
my5 + my6
mx5
Ixx5 - Iyy5 + Izz6
Ixx3+Iyy2+Izz4+441*m3+441*m4/2500+441*m5/2500 +441*m6/2500+441*m7/2500+409*mz3/1000
(369*m5)/2000 + (2*m6)/5 + (2*m7)/5 + mz4 + mz5
Iyy2+Iyy3+418*m3/10000+441*m4/2500+441*m5/2500+441*m6/2500+441*m7/2500+409*mz3/1000
Iyz2
Ixy2Izz3-Iyy2-Ixx3-418*m3/10000-441*m4/2500-441*m5/2500-441*m6/2500-441*m7/2500-409*mz3/1000
c7
c3
c4
c2
Ixz2
c6
Iyy2 - Ixx2 + Izz2
mx2
my3 + my4
mx3
(409*m3)/2000 + (21*m4)/50 + (21*m5)/50 + (21*m6)/50 + (4200*m7)/10000 + mz2 + mz3
c5
cc1
cc7
Ixx2 - Iyy2 + Izz1
cc6
cc4
cc3
cc5
cc2
c1

Cuadro 3.1: Parámetros Base

4. DISEÑO DE EXPERIMENTOS

En las secciones 2 y 3, se ha abordado la forma en que se puede construir un modelo dinámico inverso de un sistema mecánico. Este modelo tal como se describe es lineal respecto a los parámetros del sistema. Estos modelos, son útiles para diversas áreas como la reducción de modelos visto en la sección 3, la optimización del diseño (optimización de las trayectorias) detallado en esta sección, pero son especialmente útiles para la estimación de los parámetros de sistema a través de experimentos que se aborda en la siguiente sección.

Los experimentos se realizan imponiendo un movimiento dado y midiendo, los pares de los actuadores y el propio movimiento realizado para conseguirlo.

Como se ha repetido a lo largo del trabajo, se ha procurado que el sistema sea lineal en los parámetros dinámicos por lo que aplicar la ecuación de la dinámica inversa del sistema 3.1 ($W\phi$=Y) y con los datos medidos en el experimento es posible evaluar la matriz de observaciones W , obtener el vector de los pares de los actuadores (Y) y así resolver el sistema de ecuaciones para estimar los parámetros (ϕ).

De lo anterior podemos concluir que la precisión con la que se puede estimar los parámetros depende de como se realice el experimento, por lo que es necesario planificar y analizar previamente como se llevarán a cabo dichos experimentos que nos permitan obtener resultados de la mejor calidad posible.

Del diseño de experimentos nos centraremos en la optimización de la trayectoria, aspecto de gran relevancia en la robótica. Esta optimización consiste en buscar una trayectoria óptima para el movimiento del sistema durante el experimento y aprovechando la linealidad del sistema vista en la ecuación de la dinámica inversa de la ecuación 3.1, el movimiento del sistema aparece únicamente en la matriz de observación W.

Esta optimización consiste en mejorar las propiedades matemáticas de la matriz de observaciones, aplicando criterios de optimización de mínimos locales. Que nos brindan soluciones localmente óptimas.

4.1. Criterio de optimización de trayectorias

Ya se mencionó que, el objetivo es optimizar la trayectoria que seguirá el robot y para ello es necesario saber que solución es mejor que otra, es decir definir el criterio a emplear. A continuación se describen el criterio de optimización en base a la transmisión de errores y el criterio de optimización en base a la información aplicados en el trabajo.

Criterio en base a la transmisión de errores: número de condición

Este criterio supone que al tomar las medidas en el experimento no se comete error en las mediciones de las posiciones, velocidad y aceleración por lo que el error es asociado solamente a los pares de los actuadores medidos.

De la ecuación de la dinámica inversa planteada, se pueden estimar los parámetros del sistema, como aquellos que minimicen la discrepancia entre mediciones y predicciones, para resolver el problema de optimización:

$$\hat{\phi} = arg\, min_{\phi} \parallel Y_m - W\phi \parallel \tag{4.1}$$

donde $\hat{\phi}$ es el vector de parámetros estimados, Y_m contiene los pares de los actuadores medidos en el experimento. Que se resuelve como un problema de mínimos cuadrados, dando solución mediante la matriz pseudoinversa ver referencia [6, pág. 75].

$$\hat{\phi} = (W'W)^{-1}W'Y_m \tag{4.2}$$

donde $(W'W)^{-1}W'$ es la matriz pseudoinversa del sistema.

El criterio consiste en determinar una cota máxima de la desviación en la estimación de los parámetros dinámicos, es decir que la matriz de observación asociada al experimento posea un número de condición lo más bajo posible. Por lo que al aplicar el criterio se busca minimizar el número de condición de la matriz de observación que a su vez minimicen la transmisión de errores, desde la medición de los torques hasta la propia estimación.

$$\mathcal{F}_{\kappa}(W) = \kappa(W) \tag{4.3}$$

donde $\kappa(W)$ representa el número de condición de la matriz W.

Criterio en base a la Información: D-optimality

Este criterio plantea que a la hora de realizar el experimento conocemos los sensores utilizados para medir las posiciones y pares de los actuadores por lo que sabemos el ruido de las mediciones y el problema de estimación se aborda desde una perspectiva estadística.

De la estimación de parámetros es posible plantear la matriz de varianzas y covarianzas que podemos utilizar para buscar trayectorias que proporcionen las menores varianzas posibles en la estimación.

$$VAR[\hat{\phi}] = (W'\Sigma^{-1}W)^{-1} \tag{4.4}$$

donde Σ representa la matriz de las varianzas y covarianzas de los pares de los actuadores del sistema.

Ya que el criterio de optimización requiere un valor escalar que minimizar, se realiza una conversión de esa matriz empleando el menos logaritmo del determinante del inverso de dicha matriz ver referencia [6, pág. 77].

$$\mathcal{F}_{D}(W) = -\log(det(W'\Sigma^{-1}W)) \tag{4.5}$$

la aplicación del $(-log)$, permite obtener un valor numérico y ayuda a converger al algoritmo.

4.2. Parametrización de Trayectorias

La optimización de trayectorias se encarga de buscar una trayectoria que ofrezca las mejores características posibles para la realización de un experimento con el que estimar los parámetros de sistemas mecánicos. Así, la solución que nos proporciona es, precisamente, una trayectoria que, de forma general, consta de infinitas configuraciones intermedias entre la inicial y final. Para la optimización de trayectorias se hace necesaria la definición de trayectorias en función de un conjunto finito y si es posible reducido, de *parámetros de trayectoria*. En definitiva, se hace necesaria una *parametrización de la trayectoria* mediante alguna función g que proporcione los valores de las coordenadas q en función del tiempo y en función también de un conjunto de parámetros de trayectoria ϑ ver referencia [6, pág. 77]:

$$q(t) = g(\vartheta, t) \longrightarrow \mathcal{T} = \mathcal{T}(\vartheta) \tag{4.6}$$

donde τ representa la trayectoria que contiene el conjunto de las trayectorias individuales de cada grado de libertad. De esta manera, se obtiene una trayectoria continua en función de un número discreto de parámetros. Una de las mejores formas de parametrizar una trayectoria es la utilización de *Series Finitas de Fourier*. Así, la función g de la ecuación 4.6 y sus derivadas temporales, para la parametrización de la i-ésima coordenada independiente, quedarían como sigue:

$$q_i(t) = q_{i0} + \sum_{j=1}^{NH}\left[\frac{a_{ij}}{2\pi fj}\sin(2\pi fjt) - \frac{b_{ij}}{2\pi fj}\cos(2\pi fjt)\right] \tag{4.7}$$

$$\dot{q}_i(t) = \sum_{j=1}^{NH}\left[a_{ij}\cos(2\pi fjt) + b_{ij}\sin(2\pi fjt)\right] \tag{4.8}$$

$$\ddot{q}_i(t) = \sum_{j=1}^{NH}\left[-a_{ij}2\pi fj\sin(2\pi fjt) + b_{ij}2\pi fj\cos(2\pi fjt)\right] \tag{4.9}$$

donde N_H es el número de armónicos utilizados en la serie finita de Fourier, f es la frecuencia fundamental y el periodo fundamental viene dado por la expresión $T_f = 1/f$. En este caso, los parámetros de los que depende esta función son q_{i0} y los coeficientes a_{ij} y b_{ij} . Así el vector de parámetros de trayectoria quedaría de la siguiente manera:

$$\vartheta = (\vartheta_1, \cdots, \vartheta_{N_{dof}}) \tag{4.10}$$

donde para cada grado de libertad del mecanismo se deberán determinar 2NH +1 parámetros:

$$\vartheta_i = (q_{i0}, a_{i1}, b_{i1} \cdots, a_{iN_H}, b_{iN_H}) \tag{4.11}$$

esta parametrización tiene muy buenas propiedades a la hora de optimizar trayectorias y estimar parámetros [10].

4.3. Restricciones en los actuadores

Estas restricciones se relacionan con la movilidad del robot, debido a las limitaciones que presentan la carreras de los actuadores, los límites de velocidad y la necesidad de permanecer dentro del espacio de trabajo del robot, evitando pasar por posiciones singulares.

Estas restricciones se pueden representar como una inecuación a la que esta sujeta la solución a optimizar.

$$h_R(\vartheta) \leq 0 \tag{4.12}$$

en la sub-sección anterior se describió las expresiones a través de series finitas de Fourier con las que se han parametrizado las trayectorias que son contenidas en la matriz $R(\vartheta)$, y las restricciones antes mencionados debido a la limitaciones de los actuadores en posición, velocidad y aceleración se convierten en los limites inferiores (**lb**,lowbound)y superiores (**ub**,upbound) para cada uno de los actuadores y se representa como sigue:

$$lb \leq R(\vartheta) \leq ub \tag{4.13}$$

el vector ϑ, por su parte, contiene los parámetros de trayectoria en la forma descrita mediante las ecuaciones 4.10 y 4.11. De esta manera, las restricciones pueden introducirse en los algoritmos de optimización como restricciones lineales, lo cuál facilita enormemente la labor del método numérico.

En la referencia [6] nos indican que, mediante la parametrización con series finitas de Fourier y la elección de las coordenadas de los actuadores como coordenadas independientes con las que se guía el mecanismo, se ha podido conseguir que estas ecuaciones de restricción aparezcan como expresiones lineales en los parámetros de trayectoria, de forma que su inclusión en el problema de optimización no conlleve demasiado coste computacional.

4.4. Optimización de trayectorias

Como se avisto en la sub-sección 4.1, los criterios de optimización de trayectorias plantean buscar una matriz de observación W que proporcione buenas propiedades matemáticas a la hora de realizar la estimación de parámetros. Dado que la matriz W depende de la trayectoria realizada por el sistema dinámico, se plantea la búsqueda de trayectorias óptimas para que W tenga las mejores propiedades matemáticas posibles.

En la sub-sección 4.2 se han descrito la parametrización de la trayectorias en un número finito de parámetros de trayectoria y en la 4.3 las restricciones debido a los limites en posición, velocidad y aceleración de cada uno de los actuadores. Por lo que ahora es posible plantear el problema de la optimización de la trayectoria.

Se dice que $\mathcal{T}(\vartheta^*)$ es trayectoria óptima según el criterio $\mathcal{F}$ y las restricciones h_R si:

$$\vartheta^* = arg\, min_\vartheta\{\mathcal{F}(W(\vartheta))\}; \quad sujeto\, a : h_R(\vartheta) \leq 0 \tag{4.14}$$

siendo ϑ^* los parámetros óptimos de la trayectoria optimizada.

4.5. Aplicación a un robot KUKA LBR iiwa

Para llevar a cabo el experimento, de las lecturas realizadas en la bibliografía [10] y con indicaciones del tutor del trabajo se han seleccionado las siguientes condiciones.

- La duración del experimento será de 10 segundos(t), por lo que la frecuencia de muestreo es 0.10 (f) y el número de instantes o puntos a evaluar sera 100 (n).
- Para la parametrización de las trayectorias a optimizar se han empleado series finitas de Fourier con 5 armónico.
- Para comandar el robot es necesario emplear trayectorias que comiencen y finalicen en las misma posición, por lo que será necesario acoplar un trozo de trayectoria adicional al inicio y final de las trayectoria optimizadas,como se detallará adelante.

Parametrización de la excitación de trayectoria

Para la parametrización de las trayectorias se ha empleado las series finitas de Fourier con 5 armónicos para cada coordenada generalizada, planteadas en la ecuación 4.7.

$$q_i = 1 + a_{i1}cos(wt_j) + b_{i1}sin(wt_j) + \cdots + a_{i5}cos(5wt_j) + b_{i5}sin(5wt_j) \tag{4.15}$$

donde w es la frecuencia (constante) y t el vector de los instantes de tiempo y q_i las coordenada generalizada del i-ésimo grado de libertad. Sabiendo que las coordenadas generalizadas son independientes podemos decir que, para cada grado de libertad del robot se deberán determinar $2N_H + 1$ parámetros, es decir 11 parámetros por coordenada, 77 en total.:

$$\vartheta_i = (a_{i0}, a_{i1}, b_{i1}, a_{i2}, b_{i2}, a_{i3}, b_{i3}, a_{i4}, b_{i4}, a_{i5}, b_{i5}) \tag{4.16}$$

siendo a_{ij} y b_{ij} los parámetros correspondiente al armónico j del solido i y a_{i0} el parámetro constante presente solo en las coordenadas generalizadas.

Derivando de forma analítica como se indica en la sección 4.2 en la ecuación 4.7 construida con las series de Fourier, obtenemos las funciones para las velocidades y aceleraciones generalizadas definidas en las ecuaciones 4.8 y 4.9.

Lo antes mencionado se ha llevado a cabo en MATLAB en la segunda sección de la rutina **main_simbolic**. ver Anexo B. Creando las variables de forma simbólica y construyendo las series finitas de Fourier para ser evaluadas cunado se realiza la excitación perimétrica y los experimentos.

Restricciones debido a los actuadores

Del manual de especificaciones del robot KUKA [2], se visualiza que las restricciones del robot son establecidas como las limitaciones de movimiento y el máximo torque suministrado por los actuadores como se muestra en la siguiente figura.

Axis data / Range of motion		LBR iiwa 7 kg Maximum torque	LBR iiwa 7 kg Maximum velocity	LBR iiwa 14 kg Maximum torque	LBR iiwa 14 kg Maximum velocity
Axis 1 (A1)	± 170°	176 Nm	98°/s	320 Nm	85°/s
Axis 2 (A2)	± 120°	176 Nm	98°/s	320 Nm	85°/s
Axis 3 (A3)	± 170°	110 Nm	100°/s	176 Nm	100°/s
Axis 4 (A4)	± 120°	110 Nm	130°/s	176 Nm	75°/s
Axis 5 (A5)	± 170°	110 Nm	140°/s	110 Nm	130°/s
Axis 6 (A6)	± 120°	40 Nm	180°/s	40 Nm	135°/s
Axis 7 (A7)	± 175°	40 Nm	180°/s	40 Nm	135°/s

Figura 4.1: Tabla de especificaciones del fabricante [2]

De nuestro interés y de cara al experimento se han seleccionado los limites de carrera de posición y velocidad correspondiente al LBR iiwa 14 Kg. Para los limites de posición se han tomado directamente de la primera columna de la tabla anterior y para las velocidades, él tutor se ha comunicado con la empresa ALDAKIN, para que suministrarán la información de las restricciones propias de los motores, dado que existe un factor de conversión asociado debido a la multiplicadora en cada eje, el cual no es proporcionado en el manual antes mencionado.

Los pares de los actuadores máximos tabulados para cada eje, los tomaremos en cuenta al momento de realizar el experimento y validar el modelo para comprobar que los resultados se encuentren en rangos de aceptables.

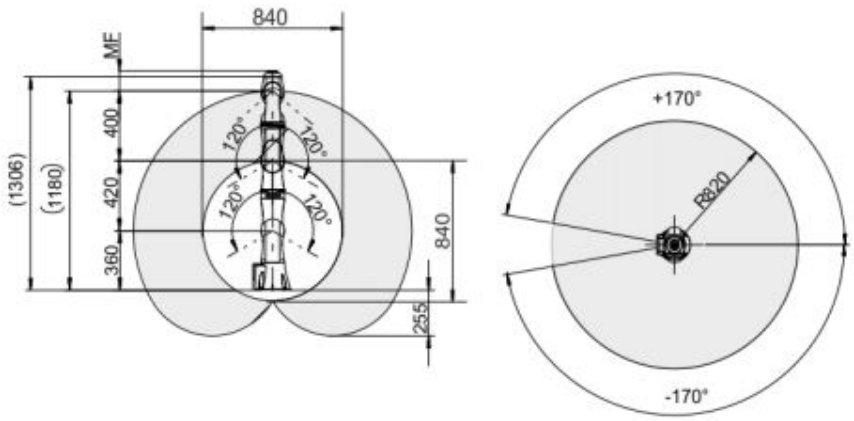


Figura 4.2: Restricciones del entorno[2]

En la figura 4.2 se puede visualizar las restricciones en los actuadores y el espacio de trabajo del robot.

De lo mencionado en la sección 4.3 resulta importante señalar que cada una de las restricciones expresadas mediante las inecuaciones de restricción, han de aplicarse a cada actuador y en cada instante de tiempo en el que evaluemos la matriz de observación.

Por tanto, la matriz $R(\vartheta)$ de la inecuación 4.13 tendrá, en general, $2N_{gdl}N_{pts}$, (4 por las restricciones en (2)posición y (2)velocidad) es decir, 4*7(100) filas y tantas columnas como parámetros de trayectoria, es decir, $N_{gdl}(1 + 2N_H)$, 7*(1+2*5)=77 incógnitas.

Se ha empleado de forma adicional un factor de seguridad del 95 % para las posiciones y velocidades para asegurar que la soluciones permanezcan dentro del rango definido. Ya que si algún valor se sale de rango al momento de comandar el robot en el experimento este ocasionaría daños a los actuadores del robot ya que estaría solicitando movimientos imposibles sea por posición o velocidad.

En la tabla 4.1 se presenta un condensado de las restricciones utilizadas tanto en posición como en velocidad agrupando los limites en inferiores y superiores como se indicó en la ecuación 4.13 de la sección 4.3. A todas los limites se le aplicará el factor de seguridad antes mencionado.

Se aclara que en el caso de los limites inferiores (Lowbound), el signo negativo representa que la orientación de los movimiento son en sentido contrario a los Upbound.

Restricciones de movilidad y velocidad de los actuadores									
	Eje	1	2	3	4	5	6	7	Unidad
Upbound	q_{max}	170	120	170	120	170	120	175	[°]
	dq_{max}	2267	2267	2667	2000	2167	3600	3600	[rpm]
Lowbound	q_{min}	-170	-120	-170	-120	-170	-120	-175	(°)
	dq_{min}	-2267	-2267	-2667	-2000	-2167	-3600	-3600	rpm

Cuadro 4.1: Restricciones de los actuadores

Lo antes mencionado se ha llevado a cabo en MATLAB en la la función **get_A_b** como se muestra en el Anexo D que es llamada en *Main_Trayectorias*. Dicha función es utilizada para alimentar la función de optimización de MATLAB (fmincon) [7] empleada para optimizar las función objetivo como se detalla más adelante.

Función Objetivo

La función objetivo es aquella que se busca minimizar su valor, por lo que contiene en su interior la definición de los criterios mencionados en la sección 4.1 en las ecuaciones 4.2 y 4.5.

Es necesario, para alimentar la función objetivo que deseamos optimizar las condiciones del experimento como ser la frecuencia de muestreo, el tiempo total del experimento y el criterio de optimización a utilizar para dicha trayectoria.

El procedimiento llevado a cabo en la función objetivo es el siguiente, se asignan valores aleatorios a los parámetros de trayectoria para construir la matriz de observación W, y empleando el vector de permutaciones de columnas independientes V obtenido del análisis simbólico, se construye la matriz de observación de los parámetros base del modelo reducido necesario para aplicar el criterio de optimización seleccionado y da como salida el valor de interés que sirve como métrica de lo bueno que es la trayectoria y que nos ayudará de forma posterior para seleccionar las mejor.

Lo antes mencionado se ha llevado a cabo en MATLAB en la la función **f_objetivo**. mostrada en el Anexo D, que es llamada en *Main_Trayectorias*. La cual es utilizada para alimentar la función de optimización de MATLAB (fmincon) [7].

Optimización de las trayectorias

Para llevar a cabo la optimización de la trayectoria para el robot KUKA iiwa LBR, se ha empleado el criterio en base al número de condición y el criterio en base a la información *D-optimility* descritos en la sección 4.1.

En las secciones anteriores se han descrito la función objetivo y la función que devuelve las restricciones del sistema, ambas en conjunto con el vector de parámetros de trayectoria son las necesarias para alimentar la función de optimización de MATLAB.

Para la optimización de la Función objetivo, se ha utilizado la función fmincon de MATLAB [7] que busca mínimos locales con las restricciones definidas. Debido a la naturaleza de la función y la selección aleatoria de los parámetros de trayectoria iniciales la solución no será la más óptima. Pero si buscará los parámetros más óptimos que respeten las restricciones impuestas en todos los instantes de tiempo.

Dado que la optimización es un proceso de evaluación iterativa y algo tardado ya que depende de donde se inicie y el mínimo local que se encuentre. Por sugerencia del tutor durante la optimización se ha visualizado el comportamiento del valor de la función objetivo para cada una de las iteraciones, la cual brinda una visión, si durante la optimización se va minimizando el valor de la función.

En la figura 4.3 se muestra el comportamiento de una trayectoria optimizada y empleada para la validación del modelo en el siguiente capitulo.

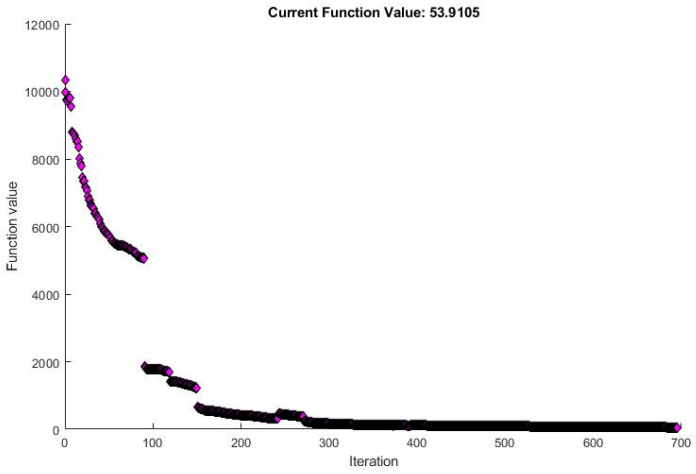


Figura 4.3: Valor de la función objetivo vs Número de iteraciones

Como se puede apreciar en la figura 4.3 durante la optimización, el valor de la función objetivo va decreciendo hasta llegar un mínimo que la función (fmicon) de MATLAB considera aceptable

en base al número de iteraciones y evaluaciones impuestas, así como otros valores internos y modificables.

Una vez realizada la optimización de diferentes trayectorias se han obtenido los siguientes resultados. Se han optimizado 4 trayectorias para cada criterio.

Criterio de Optimización Utilizado		
Trayectorias creadas	NC	MV
Trayectoria 1	57.89	240.76
Trayectoria 2	55.15	247.78
Trayectoria 3	49.25	241.69
Trayectoria 4	53.91	251.3

Cuadro 4.2: Trayectorias optimizadas según los Criterios de Optimización

Para llevar a cabo el proceso de validación explicado en la siguiente sección, cogemos de la tabla 4.2 y del criterio del número de la condición dos trayectorias, la trayectoria para llevar a cabo la estimación de los parámetros base del modelo reducido(Trayectoria 3) y la trayectoria para realizar la validación del modelo construido (Trayectoria 4).

Lo antes mencionado se ha llevado a cabo en MATLAB en la la función **Main_Trayectorias**. ver Anexo D.

Verificación de las trayectorias seleccionadas

Para la selección de las mejores trayectorias, no sólo nos vale el valor de la función objetivo obtenido, sino que es necesario visualizar el comportamiento de las posiciones y velocidades de cada uno de los 7 ejes de los motores del robot, para garantizar que cumple las restricciones impuestas.

En las siguientes figuras se muestra el comportamiento de la trayectoria seleccionada para llevar a cabo la estimación de los parámetros base del modelo reducido (Trayectoria 3, tabla 2).

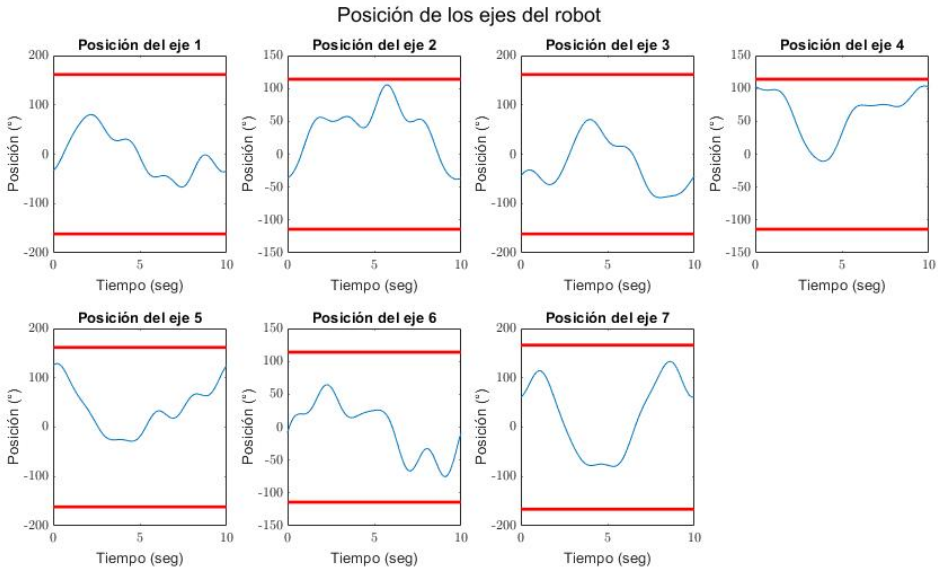


Figura 4.4: Restricciones de posición

En la figura 4.4 se puede observar que las posiciones de los ejes cumplen las restricciones impuestas (líneas rojas). En todas las trayectorias optimizadas se ha verificado lo anterior, encontrando dos de ellas que en algún momento superaban los límites por poco, pero lo suficiente para considerarlas no validas para el experimento.

En la figura 4.5 podemos visualizar las velocidades de los ejes en revoluciones por minuto y sus restricciones (líneas rojas).

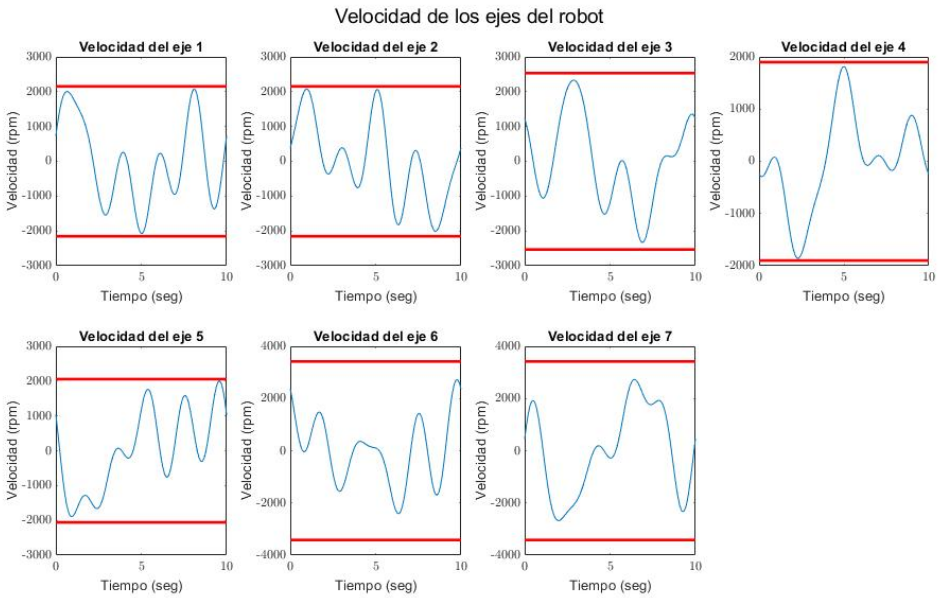


Figura 4.5: Restricciones de Velocidad

La verificación de las trayectorias se realizó en MATLAB a través de la rutina *Trayectorias_Graficas* ver Anexo D y el modelo de Simscape a través de la rutina *Main_Simscape* ver Apendice F. .

Una vez seleccionada las trayectorias que cumplan las restricciones en todo momento y presenten valores aceptables según el criterio empleado. Se procede a la comprobación de las mismas con un modelo CAD (3D_MEC) en donde es posible apreciar el movimiento de los sólidos del robot con la trayectoria comandada, (experimento virtual) como lo debería realizar en el experimento físico con el robot real.

Consideraciones sobre la tipología del robot

De cara a realizar los experimentos con las trayectorias obtenidas, es necesario mencionar dos cosas, primero que el robot con el que se trabaja KUKA LBR iiwa permite el ingreso de la trayectoria tal cual la hemos creado (punto a punto), y segundo que el robot tiene un principio básico que siempre debe conocer dónde se encuentra antes de iniciar cualquier movimiento, por lo que siempre inicia de una posición conocida (HOME), dicha posición es mostrada en la figura 4.6.

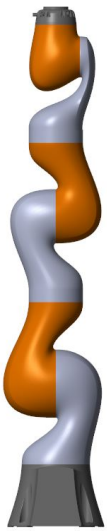


Figura 4.6: Robot en posición Home

Esta posición no es más que el robot completo extendido de forma vertical(todos los sólidos del robot alineados). Y es deseado que finalice en esa posición de ser posible, aunque de no ser así el robot retorna a ella por sí solo antes de realizar otra asignación.

De lo mencionado anteriormente y según lo observado en la figura 4.4,nuestras trayectorias optimizadas no siempre parten de esa posición. Por lo que es necesario acoplar trozos de trayectoria que cumplan dichas condiciones.

El tutor de este Trabajo de Fin de Máster Xabier Iriarte, ha realizado un código que permite acoplar al inicio y al final de las trayectorias optimizadas los trozos de curvas necesarios para que se cumpla que, en el experimento la trayectoria comandada comience y termine en la misma posición (HOME) y se acople perfectamente tanto al inicio como al final de nuestras trayectorias. Para ello a utilizado curvas de Bézier de orden 5.

Es en este momento y con el mismo código se realiza la corrección mencionada sobre la orientación del eje 4 (invertido en el modelo construido) ver figura 2.4 del trabajo de la referencia [10]. Cabe mencionar que esta trayectoria total es la que será utilizada para comandar al robot en el experimento directamente.

Dicho código también nos permite realizar la simulación de la trayectoria utilizando un modelo creado en $3D\_MEC\_MATLAB$ y almacenar los datos de la simulación en un archivo "csv" u otro formato deseado, que será empelado para alimentar el robot en el experimento así como para cualquier simulación de la trayectoria en un modelo virtual.

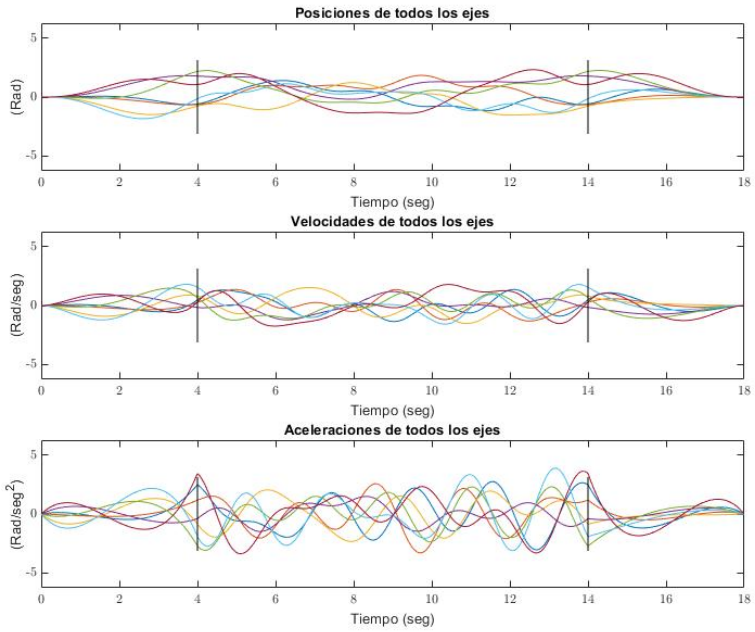


Figura 4.7: Trayectorias para los experimentos

De las curvas mostradas en la figura 4.7 las lineas verticales en 4 y 14 segundos marcan el acoplamiento de las curvas de Bézier con las curvas de las trayectorias optimizadas.

De las figuras 4.8 4.9 se destaca que todos los ejes cumplen la segunda condición, que el robot inicia y finaliza en HOME. Estas curvas son comparables con las de las figuras 4.4 y 4.5, con la única diferencia de las curvas acopladas a sus extremos, esto comentario se realiza debido a que por el escalado de la imagen puede dar la impresión que son diferentes.

Aunque la verificación de las trayectorias está realizada perfectamente con el código mencionado y creado por el tutor del trabajo. De forma adicional el autor de este Trabajo de Fin de Máster se tomó la tarea de llevar a cabo la simulación para visualizar las trayectorias empleando otras herramientas como el complemento (SIMSCAPE,$Multi-body$) de MATLAB, que en conjunto con Simulink permite modelar el comportamiento de un sistema exportado de un ensamblaje desde un modelo CAD(Solidworks).

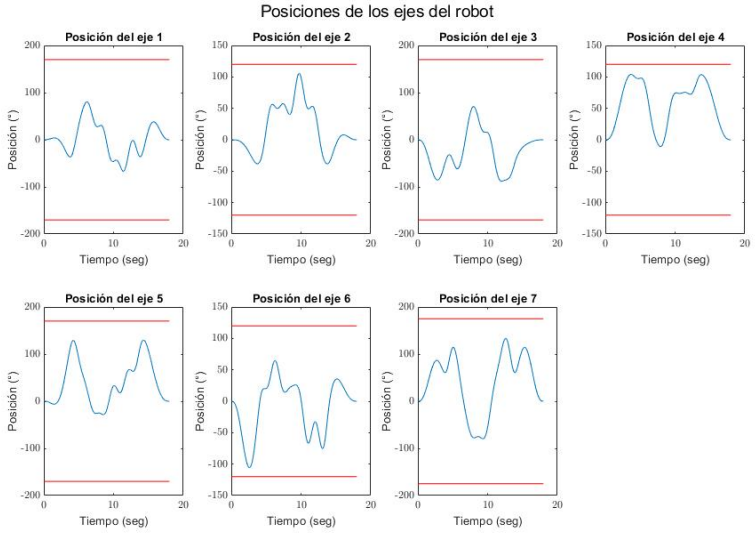


Figura 4.8: Trayectorias para las posiciones de los experimentos

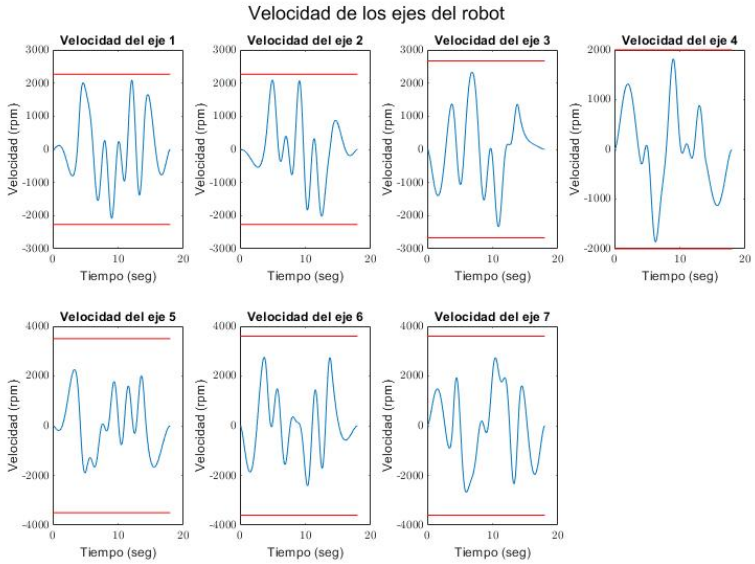


Figura 4.9: Trayectorias de las velocidades para los experimentos

Simulación del Experimento

Como se ha mencionado en los objetivos y a lo largo de este trabajo, para validar el modelo matemático construido y estimar los parámetros inerciales y de fricción es necesario llevar a cabo un conjunto de experimentos con el robot tal como se detalla en las secciones siguientes.

Debido a unos inconvenientes que se han presentado en la empresa ALDAKIN que se salen de las manos del tutor Xabier Iriarte y el autor de este trabajo, no fue posible concluir el conjunto de experimentos necesarios para cumplir la validación. Solamente se realizo la prueba inicial y luego surgió el inconveniente, por lo que fue necesario buscar una alternativa.

De forma muy general se describe el procedimiento que se debe seguir para realizar el experimento

1. Comandar el robot con una trayectoria optimizada previamente y que respete todas las restricciones establecidas.

2. Observar el comportamiento del robot, es decir, que realice lo que se espera que haga.

3. Obtener mediante la medición de los sensores propios del robot ,las posiciones pares de los actuadores o torques de los motores en las juntas.

4. Realizar un conjunto de repeticiones para obtener muchos datos y de forma posterior tratarlos con estadística.

5. Repetir los pasos anteriores para otra trayectoria optimizada.

Es necesario realizar el experimento con cierta repetibilidad y con trayectorias diferentes, ya que se requieren mínimo dos conjunto de resultados. El primer conjunto de datos (trayectoria 1) para la estimación de los parámetros y el segundo conjunto de datos (trayectoria 2) para la validación del modelo ambas seleccionadas de la tabla 4.2.

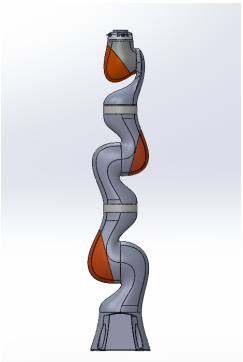
En conversación con el tutor y en vista del tiempo disponible para realizar el trabajo se optó por realizar el experimento de forma virtual mediante un conjunto de simulaciones perturbadas con ruido.

En siguiente apartado se detalla como se llevaron a cabo las simulaciones para sustituir los experimentos pertinentes. De cara a validar el modelo creado, se ha simulado el movimiento que se debe comandar al robot y añadiéndole un ruido gaussiano que represente los errores debido a los actuadores, amortiguación, holgura, error del sensor, error en la medición y demás perturbaciones existentes en el robot.

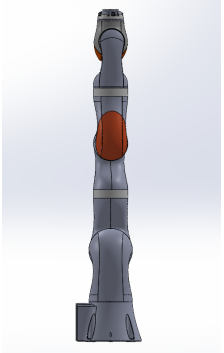
Modelo Virtual creado para el experimento

El objetivo de este modelo, es que sea igual al robot real. El cual se ha obtenido de un archivo CAD disponible en internet en GRABCAD [9], sitio de internet en donde se encuentran diversos modelos CAD, se ha seleccionado aquel que el autor considera que mejor representa al robot real ver figura 4.10. Luego de evaluar el modelo seleccionado se concluye que cumple solamente desde el punto de vista geométrico. Esté fue medido y comparado con las especificaciones definidas en el manual por KUKA[2].

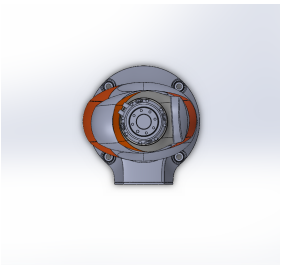
De ahora en adelante cuando se menciona el modelo experimental nos referimos al Modelo virtual para los experimentos.



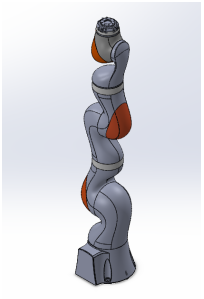
(a) Vista Frontal



(b) Vista Lateral



(c) Vista Superior



(d) isométrico

Figura 4.10: Modelo del robot exportado para la simulación

Para la construcción y modificaciones del modelo experimental se ha empleado el programa Solidworks, del cual se aprovecha su versatilidad para realizar el ensamblaje de los componentes y su compatibilidad con Simulink, que hasta la realización del trabajo el autor desconocía. Esta última ventaja se detalla más adelante.

El modelo experimental inicial no cumple las propiedades inerciales de cada sólido del robot, ya que fueron creados como cuerpos sólidos sin considerar la distribución de masas internas y los espacios necesarios para los actuadores, reductoras cableado y demás accesorios. Por lo tanto para realizar los experimentos se modificaron todos los sólidos. Realizando cortes internos (vaciados) en cada uno, retirando masa y aproximando de una mejor manera a lo que se espera sea la distribución de masa de cada sólido. **Esto incurre claramente en cierto error, pero es la mejor aproximación lograda para simular los experimentos**, ver figura 4.11.

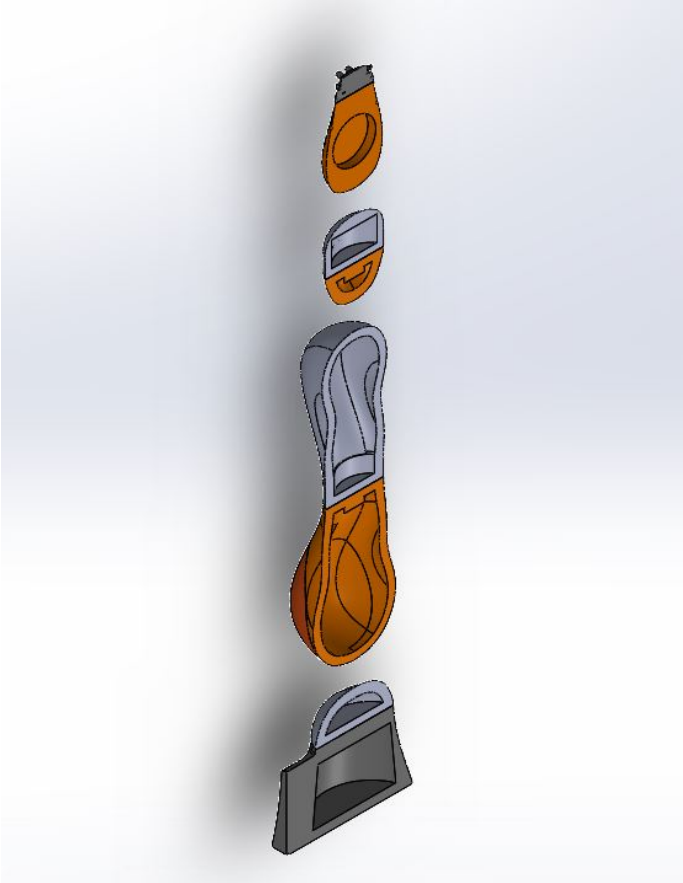


Figura 4.11: Corte de sección del modelo experimental

Una vez realizada las modificaciones en la distribución de las masas de los sólidos. Se procede a modificar las bases de referencia inercial de cada uno como se muestra en la 4.12, este paso es fundamental para que el modelo experimental coincidan con nuestro modelo. Luego se realiza el ensamblaje de los sólidos por medio de juntas de revolución entre par de sólidos (Juntas cinemáticas) de la manera más simplificada posible.

Lo anterior es llevado a cabo para obtener de forma más aproximada los parámetros inerciales del modelo experimental. Para todos los sólidos se ha considerado que están construidos de aluminio con un densidad de $2700{,}00 Kg/m^3$, **se hace énfasis en que el robot real y sus componentes están construidos de materiales diversos.**

Al igual que en la librería 3d_MEC_MATLAB creada por Javier Ros [5], nos permite obtener las propiedades inerciales de los sólidos. Solidworks nos proporciona las masas, posición hacia el centro de gravedad, tensor de inercia a partir del sistema de coordenada seleccionado (de allí la importancia de re-referenciar las bases de todos los sólidos), en la figura 4.13 se muestra la estructura que nos proporciona el programa con los datos mencionados.

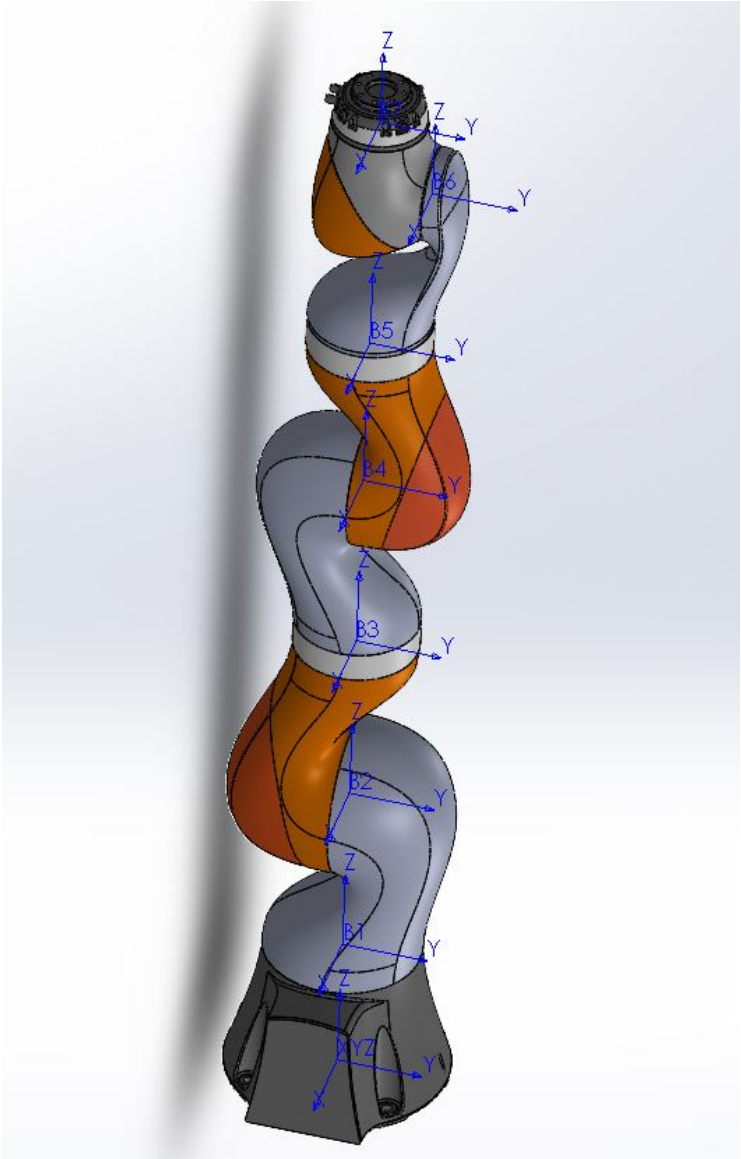


Figura 4.12: Bases del modelo

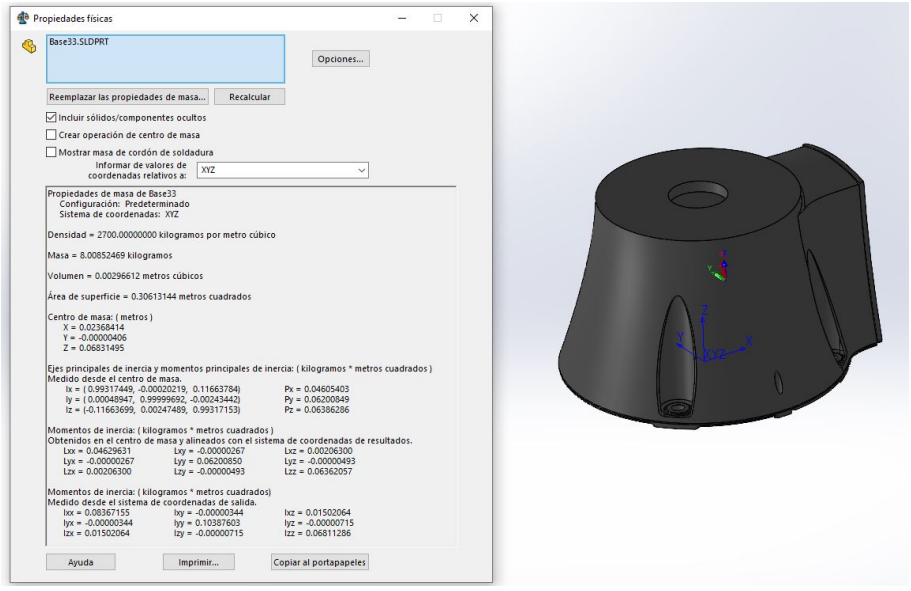


Figura 4.13: Propeidades de la Base del modelo

SIMULINK Y SIMSCAPE

Con el modelo listo y con las modificaciones necesarias se ha exportado el modelo a Simulink y MATLAB, con el cual podemos manipular las variables de interés, sólidos y pares cinemáticos. El programa también nos permite visualizar el modelo a través del complemento de SIMSCAPE.

De la exportación se obtienen dos archivos de interés, un primer archivo (Data_File.m) con la información de los sólidos, uniones y transformaciones de las bases(smiDATA.mat), se corre una vez y se guarda la estructura y un modelo (sxl) que posee el diagrama de bloques con la misma información. Ambos se pueden manipular, el archivo .slx es llamado por MATLAB y presenta como salida la simulación con SIMSCAPE.

En la figura 4.14 se puede observar que nuestro modelo esta construido principalmente por los bloques de los sólidos del sistema , unidos entre sí con los bloques del par de revolución aplicado.

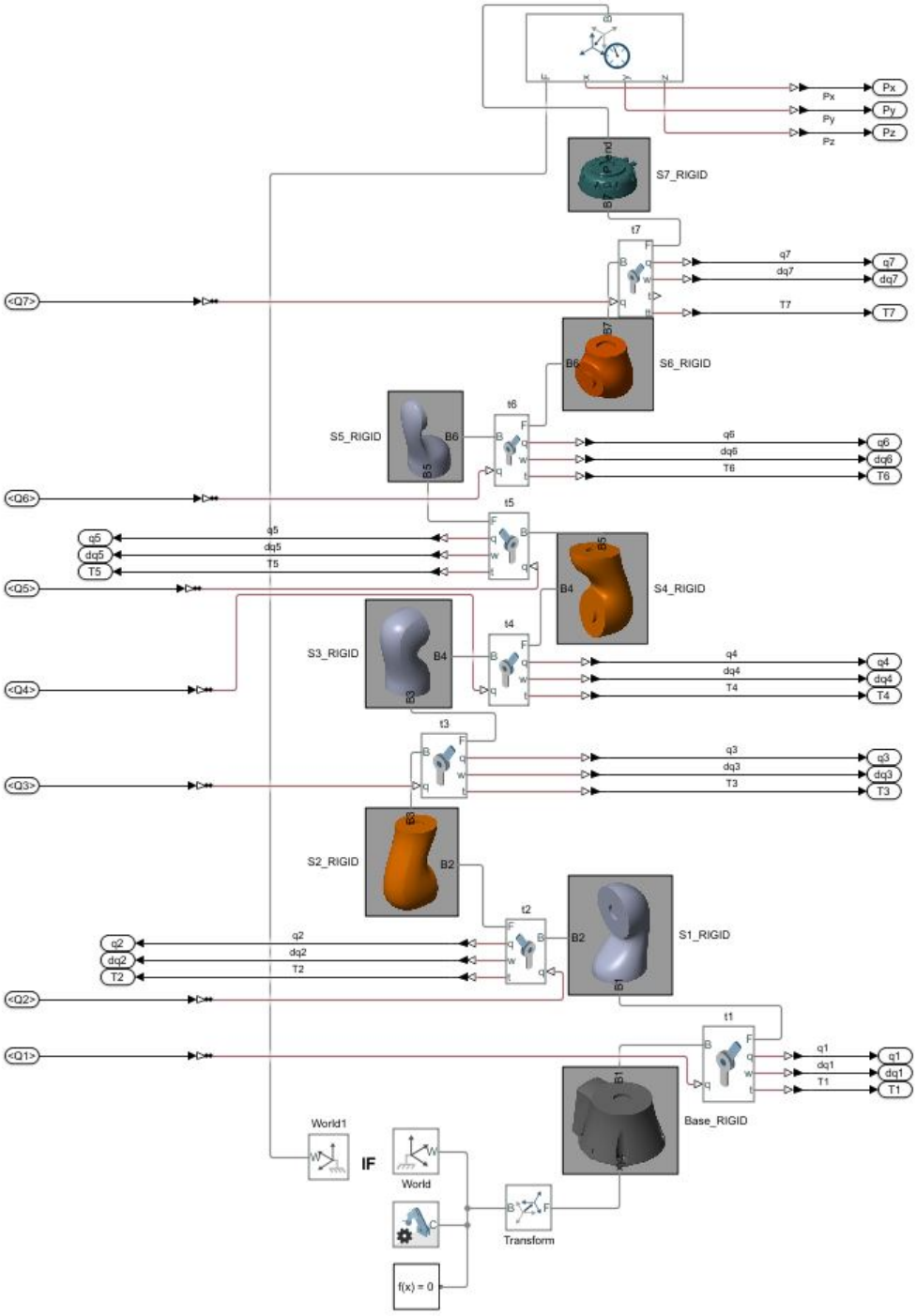


Figura 4.14: Ensamblaje de Simulink

Los bloques de los sólidos contiene en su interior, la referencia al archivo del sólido del modelo CAD utilizado, las bases creadas en dicho archivo y las transformaciones que genera SIMSCAPE. Los bloques de las revoluciones reciben como entrada los movimientos comandados por la trayectoria optimizada y acoplada como se indicó en la sección anterior. Se presentan como salida (simulando sensores) tres variables, las posiciones para validar que a la salida del modelo se sigue la trayectoria comandada, las velocidades y los torques de los motores para cada eje.

Sí que es verdad, que el modelo experimental es alimentado con las misma trayectoria que el modelo usado para estimar los parámetros base y poseen las mismas propiedades dinámicas consideradas como reales. No obstante, la salida de los torques del modelo experimental la podemos emplear para comparar nuestros torques estimados con el modelo. Ya que estos pares de los actuadores los calcula en programa en base a la dinámica del sistema.

Con el modelo de la figura 4.15 somos capaces de simular las trayectoria para visualizarla, graficar las curvas de posición, velocidad y confirmar que hace lo que quiero. De forma adicional puedo obtener un conjunto de valores de pares de los actuadores que el programa calcula automáticamente para mover las masas de cada sólido y cumplir la trayectoria esperada, como se mostrará al final de esta sección.

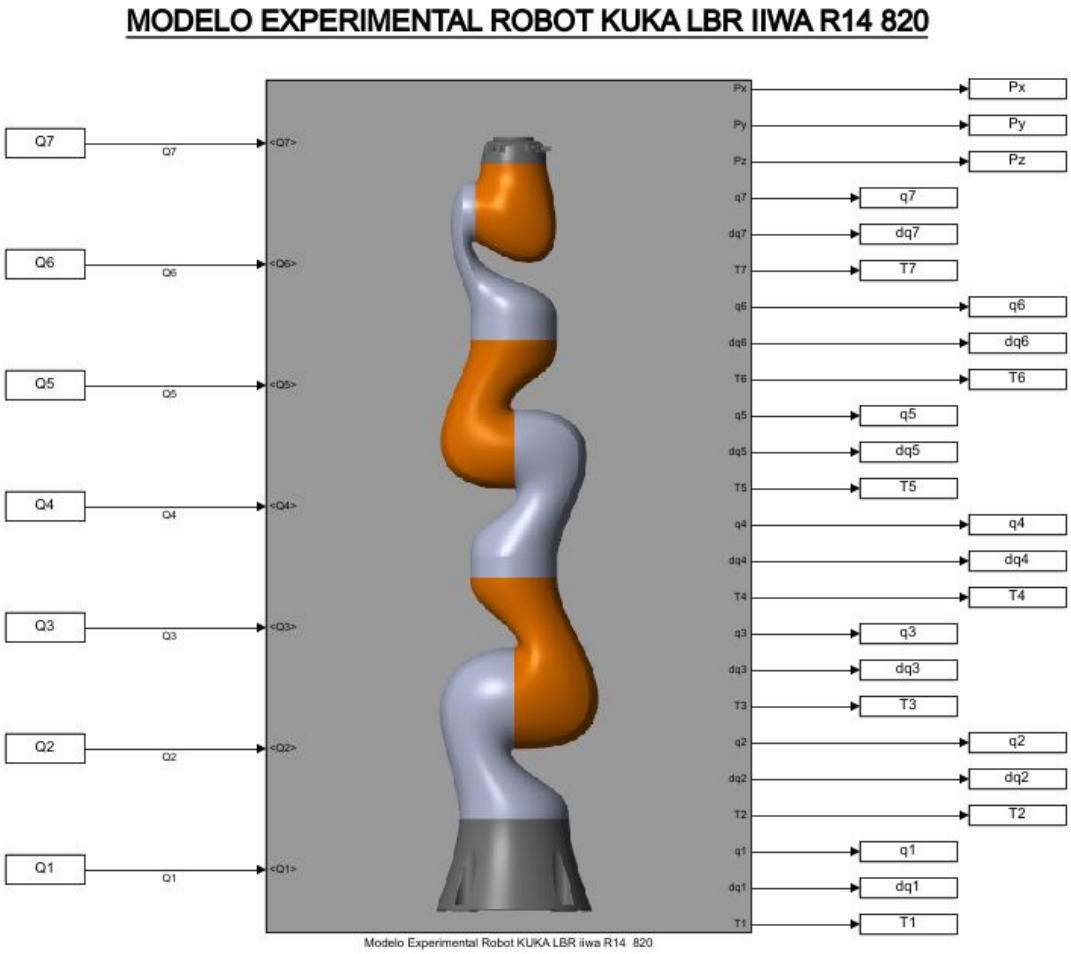


Figura 4.15: Modelo Experimental, entradas y salidas (sensores)

SIMSCAPE

Se ha realizado el modelo con el complemento de MATLAB, SIMSCAPE Multibody, para visualizar las trayectoria optimizada, validar que no existen colisiones entre los sólidos y que se respeten las restricciones angulares.

Como se ha mencionado, este modelo creado me permite comandar directamente de MATLAB las posiciones deseadas por cada eje.

A continuación se muestra en la figura 4.16 la visualización que nos brinda SIMCAPE Multibody, del modelo.

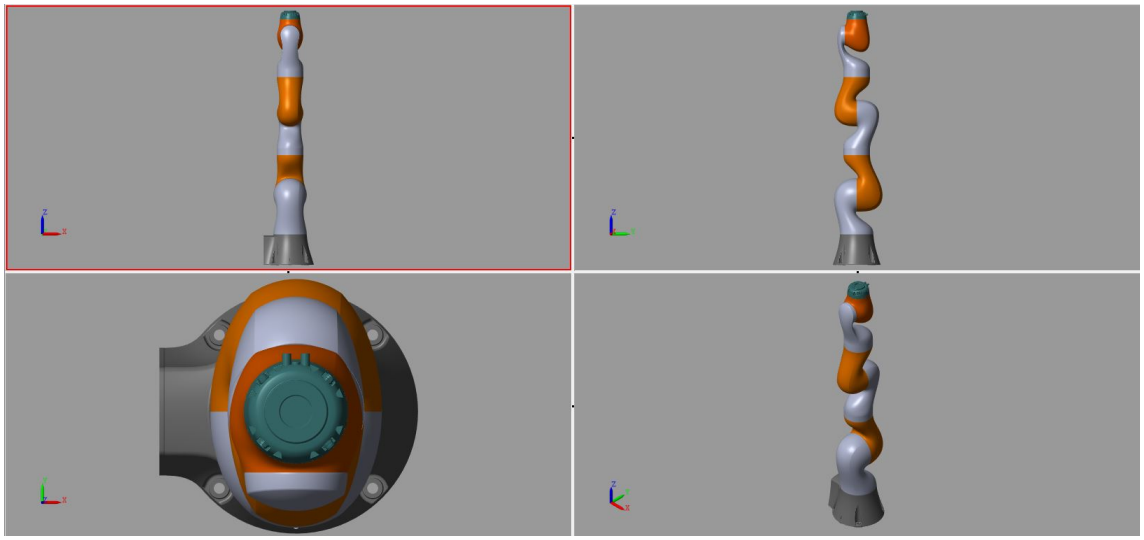


Figura 4.16: Modelo construido en SIMSCAPE

En la figura 4.17 se muestra como se definen los sólidos en la herramienta, en la cual hace referencia al archivo CAD de la pieza y las propiedades inerciales se toman de la rutina KUKA_iiwa_Ensamblaje_DataFile, creada al exportar el modelo, el cual es modificable una vez exportado el ensamblaje.

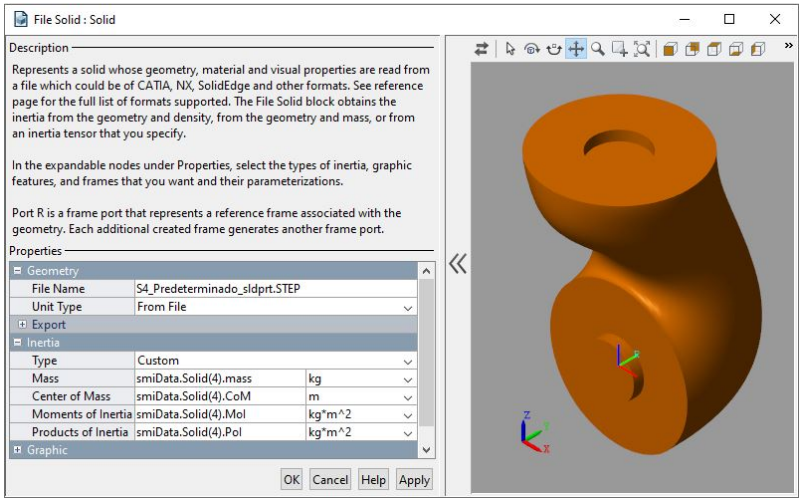


Figura 4.17: Propiedades de sólidos en Simscape

Se menciona que la herramienta, realiza la transformación automáticamente de las bases, se realiza empleando la regla de la mano derecha con el eje z positivo siempre en dirección del giro como comúnmente se maneja en la robótica.

Para las juntas cinemáticas se ha despreciado los efectos de la rigidez y amortiguación de la misma.

Se representa el modelo en la figura 4.18 una vista isométrica, del visualizador de SIMSCAPE y un vídeo del robot con la trayectoria comandada.

Figura 4.18: Modelo experimental en Simscape

La verificación de las trayectorias se realizó en MATLAB a través de la rutina *Trayectorias_Graficas* ver Anexo D y el modelo de Simscape a través de la rutina *Main_Simscape* ver Apendice F. .

Para finalizar con el modelo experimental se han generado las curvas a partir de las salidas del modelo "medidas" por los sensores colocados. Obteniendo los resultados de las figuras 4.19 y 4.20, en donde se aprecia que el robot realiza el movimiento de todos los ejes de la forma esperada, iguales que en la figura 4.8 y 4.9.

En la figura 4.21 se muestran las salidas de los torque que serán compararás con las generadas en mi modelo "real" al final de la siguiente sección. Estas salidas se han tomado directamente del modelo sin realizar ningún tipo de filtrado, se ha empleado la configuración estándar de Simulink. Se menciona, ya que en las curvas se puede observar un elevado ruido. También hay que tomar en cuenta que al modelo no se le ha definido valores de rigidez ni amortiguación de la junta, las cuales el programas las ha tomado como cero.

Considero que dedicando más tiempo y estudiando con más profundidad los valores adecuados para las juntas, los resultados mejorarían y dado que este modelado se ha realizado de forma adicional, el análisis antes mencionado esta fuera del alcance de este trabajo. Pero abre una puerta en el ámbito personal, para trabajos o investigaciones futuras en el modelado mecánico con este tipo de herramientas.

En la figura 4.22, es posible comparar el torque (azul) calculado por Simulink y las curvas verdes las estimadas por el modelo. En 5 ejes se aprecia que ambas siguen la misma forma, pero con mayor ruido en el modelo de Simulink.

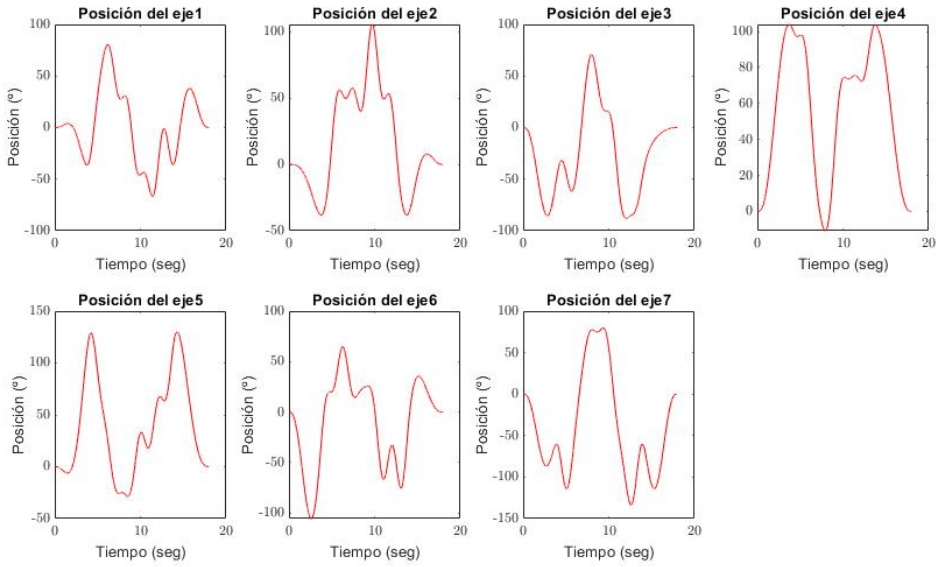


Figura 4.19: Trayectorias para las posiciones del modelo

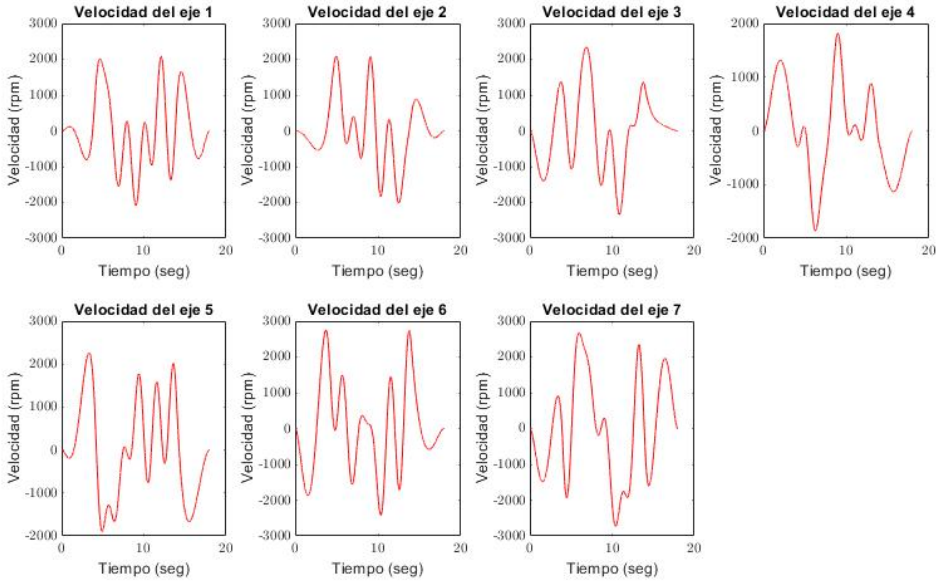


Figura 4.20: Trayectorias de las Velocidades de los sensores

5. VALIDACIÓN

5.1. Simulaciones para sustituir a los experimentos

Obtención del Torque real de los motores

En este apartado se detalla como se llevaron a cabo las simulaciones para sustituir los experimentos pertinentes que no se han podido realizar. De cara a validar el modelo creado el tutor a sugerido realizar el experimento de forma virtual, simulando el movimiento que se le comandaría al robot y añadirle un ruido gaussiano que represente los errores debido a los actuadores, amortiguación, holgura y demás perturbaciones existentes en el robot.

Ya que se va a simular el experimento, procedemos a calcular los torques en los diferentes ejes de cada actuador (motor), empleando la ecuación de la dinámica inversa que de forma lineal relaciona la matriz de observaciones W y los parámetros inerciales y de fricción definidos como reales. ϕ_{real}.

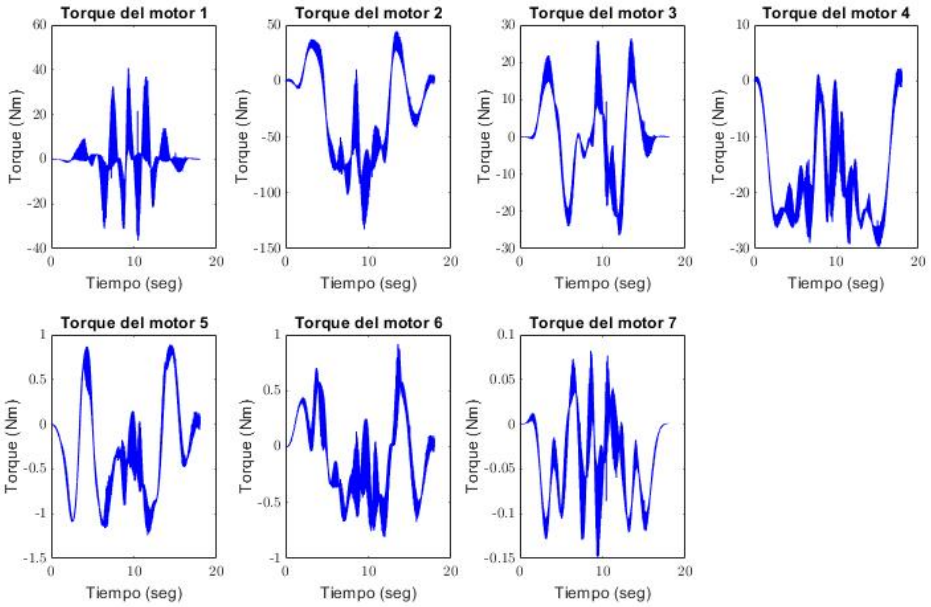


Figura 4.21: Torques del modelo experimental

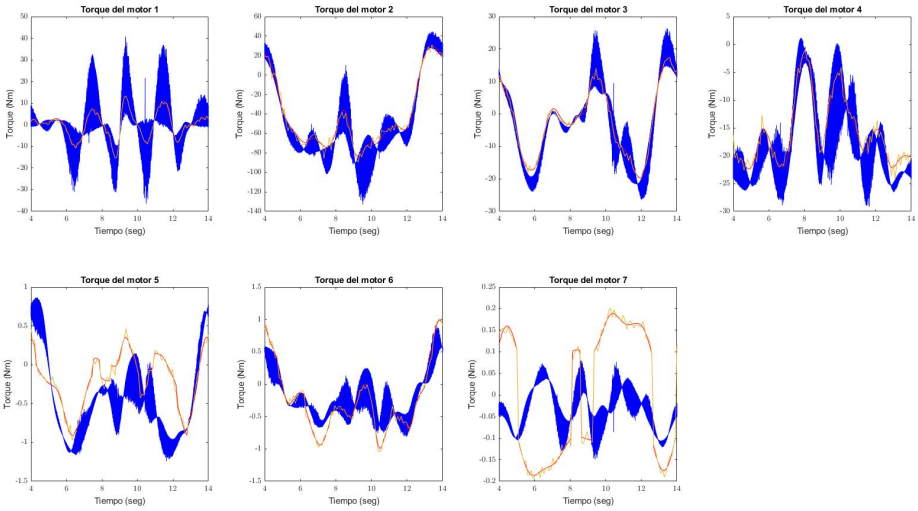


Figura 4.22: Torques del modelo experimental vrs Estimados, comparación preliminar

$$T_{real} = W\phi_{real} \tag{5.1}$$

este vector resultante, contiene los torques "reales" generados por los motores del robot con la trayectoria comandada.

Aplicar error de medición a los sensores de posición y Torques (RUIDO)

Para la simulación del experimento se procede a aplicar un ruido gaussiano a los dos variables medidas (posición y torque), dicho ruido se puede asociar a errores en la técnica de medición, error debido al propio sensor (precisión) y demás factores que pueden surgirr en un experimento, en las ecuaciones 5.2 y 5.3 se muestra el ruido asumido y la creación del torque medido con el ruido..

$$\sigma_{Torque} = 0{,}05 * Tmax_i \tag{5.2}$$

$$T_m = T_{real} + \sigma_{Torque} * randn(size(T_{real})) \tag{5.3}$$

Aproximación a una serie finita de Fourier a los datos medidos

Como es de esperar, debido a la dinámica del sistema de control con la que comandemos el experimento y por las medidas contaminadas con ruido, el robot no seguirá de forma perfecta la trayectoria comandada.

La influencia de lo antes mencionado se puede reducir creando una trayectoria parametrizada mediante series finitas de Fourier del mismo número de armónicos que la usará para la excitación. De esta manera se puede estimar la trayectoria realizada en el experimento por los actuadores, como aquella que con la misma parametrización se ajuste de forma más óptima a los desplazamientos medidos. Y dado que la función empelada para la trayectoria es continua y derivable, es posible obtener a través de la derivada analítica de la función estimada las velocidades y aceleraciones para poder evaluar el modelo [6].

5.2. Estimación Máximo Verosímil de las Variables phi y Tau (Modelos lineales)

En esta sección se describirá la estimación Máximo Verosímil de las variables ϕ y Σ, empleando un modelo lineal en los parámetros inerciales y de fricción, así como considerando un error con distribución normal de media 0 y varianza (Σ).

Sea $Y_m = Y + \epsilon$ y supongamos que $\mathbf{x}$ se mide sin error. Para el modelo $y = K(x) * \phi$ calcular el estimador $\hat{\phi}_{MV}$ tomado de la referencia [6].

$$\epsilon \sim N(0, \Sigma), \quad Y_m \sim N(Y, \Sigma), \quad dim(Y) = n. \tag{5.4}$$

Supongamos que las ecuaciones de nuestro sistema son:

$$K_{(d\times p)}\phi_{(p\times 1)} = \mathbf{y}_{(d\times 1)} \tag{5.5}$$

donde p es el número de parámetros y d es el número de ecuaciones del sistema $K\phi = y$ (la dimensión del sistema).

Si se realizan mediciones en n instantes de tiempo, podemos montar el sistema de ecuaciones:

$$W_{(d\cdot n)\times p}\phi_{p\times 1} = \mathbf{y}_{(d\cdot n)\times 1} \tag{5.6}$$

donde la matriz W y el vector $\mathbf{Y}$ se definen de la siguiente manera

$$W = \begin{bmatrix} K_1 \\ K_2 \\ \vdots \\ K_n \end{bmatrix}; Y = \begin{Bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{Bmatrix}. \tag{5.7}$$

Suponiendo que el error de medición es independiente del instante de tiempo en el que se haga la medición, la matriz de error de medición del sistema ($\Upsilon_{d\cdot n\times d\cdot n}$) se creará mediante composición de n matrices de error de medición instantáneo ($\Sigma_{d\times d}$) de la siguiente manera[6]:

$$\Upsilon = \begin{bmatrix} \Sigma & 0 & 0 & 0 & 0 \\ 0 & \Sigma & 0 & 0 & 0 \\ 0 & 0 & \Sigma & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \Sigma \end{bmatrix} \tag{5.8}$$

El algoritmo empleado para llevar a cabo la estimación del vector de los parámetros de inercia es ϕ y la matriz de varianzas Σ, tomado de la referencia [6] es el siguiente:

$\hat{\Sigma} = I_d$

$\hat{\Upsilon} = I_n \otimes \hat{\Sigma}$

$\hat{\phi}^{(0)} = 0$

$\hat{\phi}^{(1)} = (W'\hat{\Upsilon}^{-1}W)^{-1}W'\hat{\Upsilon}^{-1}Y_m$

$While \parallel \hat{\phi}^{(k+1)-\hat{\phi}^{(k)}} \parallel > TOL$

$$\hat{\phi}^{(k)} = \hat{\phi}^{(k+1)}$$

$$\hat{\Sigma}^{(k)} = \frac{1}{n}\sum_{i=1}^{n}(y_{mi} - K(x_{mi})\hat{\phi}^{(k)})(y_{mi} - K(x_{mi})\hat{\phi}^{(k)})'$$

$$\hat{\Upsilon}^{k} = I_n \otimes \hat{\Sigma}^{k}$$

$$\hat{\phi}^{(k+1)} = (W'\hat{\Upsilon}^{-1}W)^{-1}W'\hat{\Upsilon}^{-1}Y_m$$

End

$VAR[\hat{\phi}] = (W'\Upsilon^{-1}W)^{-1}$

$VAR[y(r)_t] = K(r,:)_t[VAR[\hat{\phi}]]K(r,:)'_t$

donde representa la varianza de la estimación de la r-ésima componente del vector y en el instante t y la expresión $K(r,:)_t$ representa la r-ésima fila de la matriz K en el instante t. Por su parte, i recorre las muestras de todos los instantes de tiempo y k es el número de iteración del cálculo de ϕ y Σ.

5.3. Residuo de los tau

Una vez que se ha obtenido la estimación para los parámetros, $\hat{\phi}$, para una x cualquiera (incluso una que no estuviera en la muestra con la que se ha hecho la estimación de ϕ es posible estimar la variable de salida y, tomado de la referencia [6].

$$\hat{y} = K(q)\hat{\phi} \tag{5.9}$$

Siendo q, las mediciones correspondientes a la posición de todos los ejes para instante dado. Utilizando los datos medidos de x para cada instante, se obtendría la estimación del vector $\mathbf{Y}$:

$$\hat{\mathbf{Y}} = W(\mathbf{q})\hat{\phi} \tag{5.10}$$

Residuo
El residuo o error de estimación (ε) se estima de la siguiente manera:

$$\hat{\varepsilon} = \mathbf{Y_m} - \hat{Y} = Y_m - W(x)\hat{\phi} \tag{5.11}$$

Los valores de la variable $\hat{\varepsilon}$ nos darán una idea de la discrepancia que hay entre los valores medidos y la estimación que se hace de éstos a través del modelo con los parámetros estimados. Dibujando un histograma de los residuos se puede ver si éstos siguen una distribución normal. También se puede estimar su media y desviación típica y observarla en el histograma tomado de la referencia [6].

5.4. Intervalos de confianza, para parámetros y taus

Cuando calculamos la varianza de un estimador, ésta nos da una idea de lo precisa que va a ser nuestra estimación del parámetro. Un Intervalo de Confianza (I.C) nos dice que con una confianza de $1-\alpha$, el intervalo contendrá el valor verdadero de θ. Se utiliza la palabra confianza en vez de probabilidad porque para crear el intervalo se necesita la muestra y si la muestra se ha obtenido, la probabilidad de obtener una muestra cualquiera es 1 (si coincide con la obtenida) ó 0 (si no coincide).

Cuando el tamaño muestral es lo suficientemente grande, se puede hacer uso de las propiedades asintóticas del estimador M.V. ver referencia [6], empleando una distribución normal tal como la se muestra en la figura 5.1.

$$\hat{\theta} \sim N(\theta, \sigma_\theta^2)$$

Es decir, el intervalo de Confianza para el para el parámetro θ puede calcularse como:

$$I.C_{1-\alpha}(\theta) = [\hat{\theta} - Z_{1-\alpha}\sigma_\theta, \hat{\theta} + Z_{1-\alpha}\sigma_\theta] \tag{5.12}$$

Cuando la varianza del parámetro θ sea desconocida, se sustituye por su estimador $\hat{\sigma}_\theta^2 = s^2$. En el caso de multiparámetrico se sustituiría por el correspondiente elemento de la diagonal de $VAR[\theta]$.

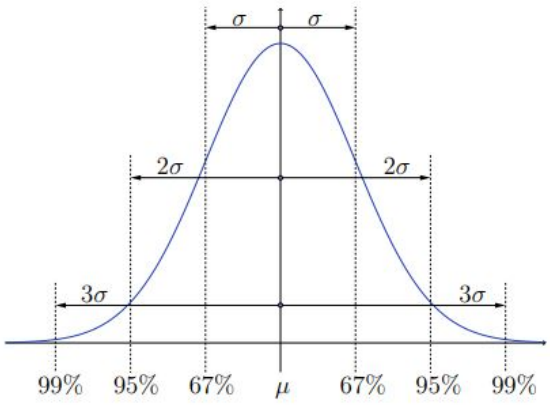


Figura 5.1: Función Densidad de Probabilidad de una distribución Normal [6]

5.5. Validación Cruzada

Para no caer en el sobre-ajuste pero poder obtener modelos que predigan con precisión, es necesaria la selección de modelos en base a algún criterio. Uno de ellos es la validación cruzada, que utiliza una serie de datos para estimar los parámetros dinámicos y otra serie de datos (necesariamente diferente, trayectoria) para validar (dar por bueno) el modelo obtenido mediante estimación, visitar referencia [6].

La selección de modelos mediante validación cruzada de datos puede realizarse de la siguiente manera: visitar referencia [6, pág. 83].

- Dividir todos los datos disponibles en dos grupos.Ambos grupos deben excitar el sistema de forma similar. De un experimento pueden tomarse, por ejemplo, los instantes pares para datos de estimación y los impares para datos de validación.
  1. Datos de estimación: $x_E, \quad y_E, \quad W_E = W(x_E)$
  2. Datos de validación: $x_v, \quad y_v, \quad W_v = W(x_v)$
- Estimar los parámetros de cada modelo candidato utilizando los datos de estimación:

$$\hat{\phi}_{\mathbf{E}} = (\mathbf{W}'_{\mathbf{E}}\mathbf{W}_{\mathbf{E}})^{-1}\mathbf{W}'_{\mathbf{E}}\mathbf{y}_{\mathbf{E}} \tag{5.13}$$

- Calcular el error de validación utilizando la estimación de los parámetros obtenidos utilizando los datos de la validación:

$$\hat{\varepsilon}_{\mathbf{v}} = \mathbf{y}_{\mathbf{v}} - \mathbf{W}_{\mathbf{v}}\hat{\phi}_{\mathbf{E}} \tag{5.14}$$

- Haciendo calculado la norma de $\hat{\epsilon_v}$ para todos los modelos candidatos, puede elegirse el modelo que menor norma dé.
- Realizar las comprobaciones adicionales para dar por bueno el modelo utilizando siempre los errores de validación ($\hat{\epsilon_v}$).

5.6. Aplicación de la validación en el robot KUKA iiwa LBR

Obtención del Torque real de los motores

Se indicó que el robot posee 7 actuadores (conjunto motor-reductora) para proporcionar el movimiento deseado. Cada conjunto sólo posee un encoder que puede estar ubicado directamente a la salida del motor es decir en la entrada de la multiplicadora o a la salida de la misma por lo que existirá un error ya que no se puede medir la rigidez ni la amortiguación de la junta.

Respecto al error de los sensores, se considera que ningún sensor es perfecto por lo que el error debido a dicha precisión es considerado.

Del experimento se toman medidas de las posiciones de las juntas en grados [°] y torques aplicados por cada motor en Newton-metros [N-m].

Para crear la simulación se ha utilizado los valores de inercia ´´reales" obtenidos a partir de las modificaciones en el modelo CAD. Es decir, que para la solución le definimos los parámetros inerciales y de fricción aproximados o creados. **Cabe resaltar que este paso no es necesario si se realizarán los experimentos.**

Para los parámetros de fricción se han considerado valores bajos ya que, no se desea que se estos tengan mayor influencia de la que deberían sobre los resultados.

De lo explicado en la sección anterior es claro, que el modelo no representa al 100 % la distribución de las masas de los sólidos. es decir existe un pequeño error asociado, que ha sido inevitable dada la situación.

Ya que se va a simular el experimento, procedemos a calcular los torques en los diferentes ejes de cada actuador (motor), empleando la ecuación de la dinámica inversa que de forma lineal relaciona la matriz de observaciones base W y los parámetros inerciales y de fricción definidos como reales, ϕ_{real} como se muestra en la ecuación 5.15.

$$T_{real} = W\phi_{real} \tag{5.15}$$

Este vector resultante, contiene los torques "reales" generados por los motores del robot con la trayectoria comandada.

A continuación en la figura siguiente se muestra las trayectorias de las posiciones simuladas como reales", las medidas, que son no más que los valores reales con ruido gaussiano aleatorio agregado para cada eje.

Aplicar error de medición a los sensores de posición y Torques (RUIDO)

Para ello se han considero inicialmente unos valores de desviaciones algo realista. Se ha seleccionado el 5 % del valor máximo para cada eje. Es decir se han tratado los ejes cada uno por separado debido a las diferencias en el orden de magnitud entre ellos. Luego se empelará la Estimación Máximo Verosímil para sistemas lineales para estimar la desviación típica del error de medición y comparar con las asumidas, en las ecuaciones 5.2 y 5.3 en donde se define el ruido asumido y la creación del torque medido con el ruido.

En la figura 5.2 se muestran los torques reales y los medidos.

$$\sigma_Q = 0{,}05 * Qmax_i \tag{5.16}$$

$$Q_m = Q_{real} + \sigma_Q * randn(size(Q_{real})) \tag{5.17}$$

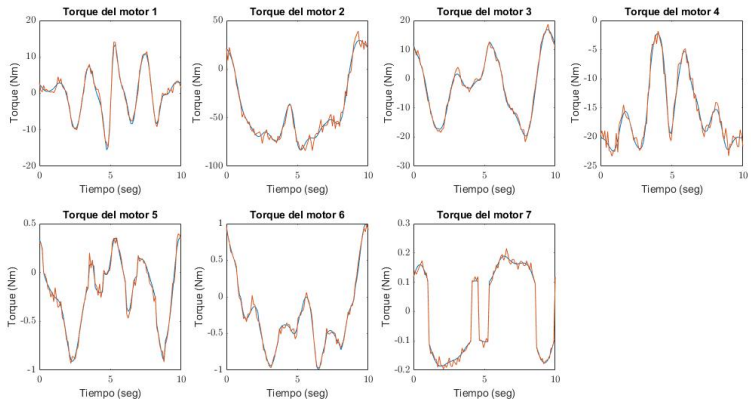


Figura 5.2: Medición de Torques con Ruido

En la figura 5.3 se muestran las posiciones medidas [º] y con ruido.

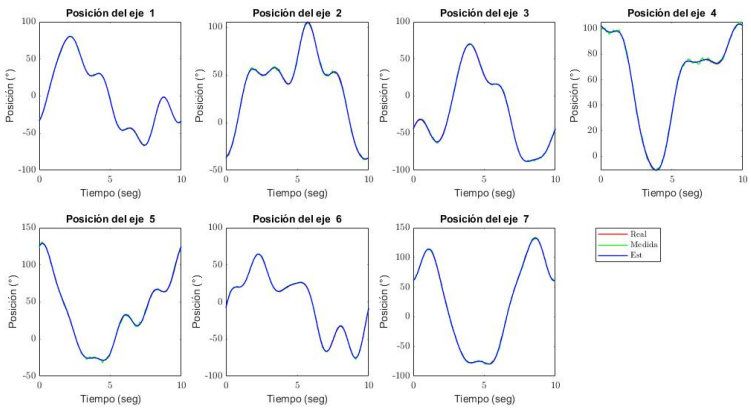


Figura 5.3: Medición de Posiciones con Ruido.

Aproximación a una serie finita de Fourier los datos medidos

Una vez realizada la simulación que reemplaza el experimento, se procede a buscar una serie finita de Fourier que se aproxime a la trayectoria medida (con ruido). Es decir, una curva que sea lo más parecida posible a la medida pero que contenga propiedades matemáticas que le permiten ser comparable de una mejor manera con la trayectoria comandada (5 armónicos) para estimar las posiciones de cada uno de los ejes del robot.

En la figura 5.4 se observan la comparación entre las curvas de la posición real, medidas y estimadas.

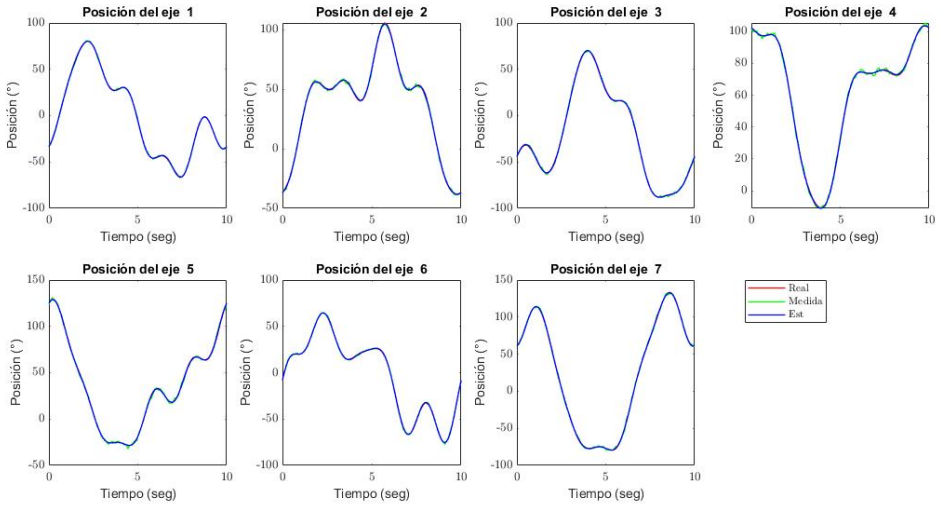


Figura 5.4: Posición de los ejes estimados.

De la misma manera que se ha hecho con las posiciones, se realiza la estimación de las velocidades y aceleraciones de los ejes, ver figura 5.5.

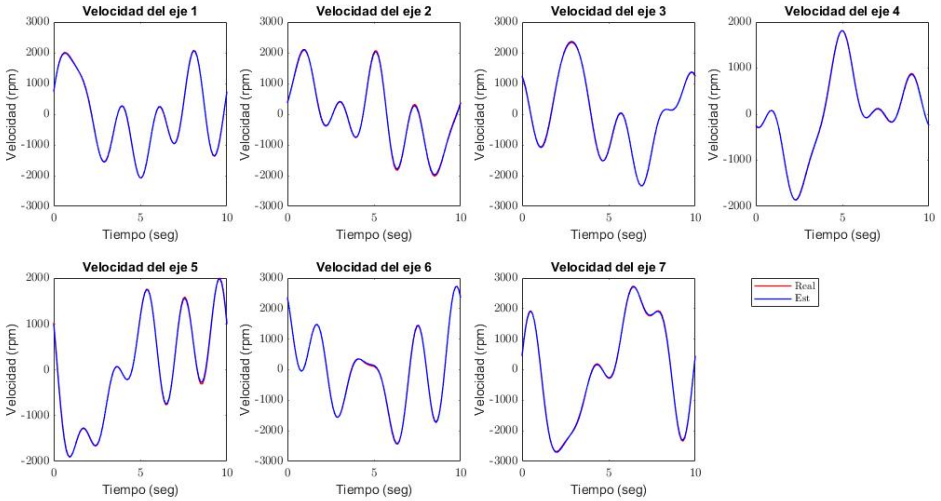


Figura 5.5: Velocidades de los ejes estimadas.

Estimación de los parámetros

Una vez que realizado los experimentos y obtenido los valores del vector de los pares de los actuadores y construida la matriz de observación del modelo reducida con los datos medidos, Se procede a la estimación de los parámetros base.

Se han representado los parámetros base estimados vs parámetros base reales, en donde se puede apreciar que la figura 5.7, en donde se estimo la matriz de varianzas presenta menos error que la varianza impuesta. ver figura 5.6.

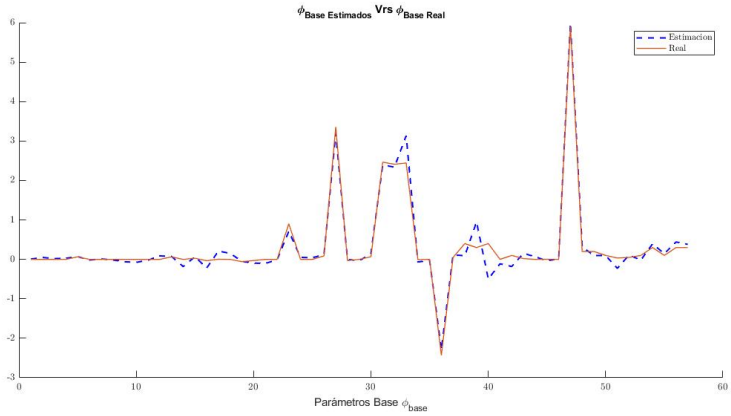


Figura 5.6: Parámetros base estimados vrs reales con desviación impuesta

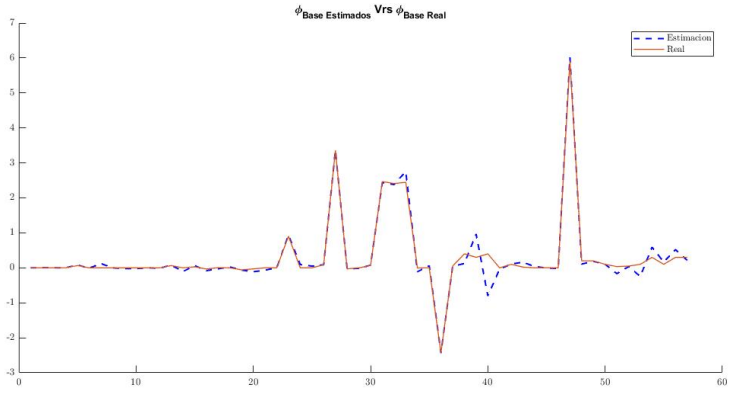


Figura 5.7: Parámetros base estimados vrs reales con desviación estimada con MV

De la figura 5.7 se puede observar que los parámetros estimados se encuentran todos en el rango esperado del 95 %, 3σ.

En la figura 5.8 donde compara los parámetros estimados y los parámetros dinámicos considerados como los reales "simulados", se puede apreciar que para la mayoría de los parámetros base (57), los valores coinciden o están muy cercanos [8].

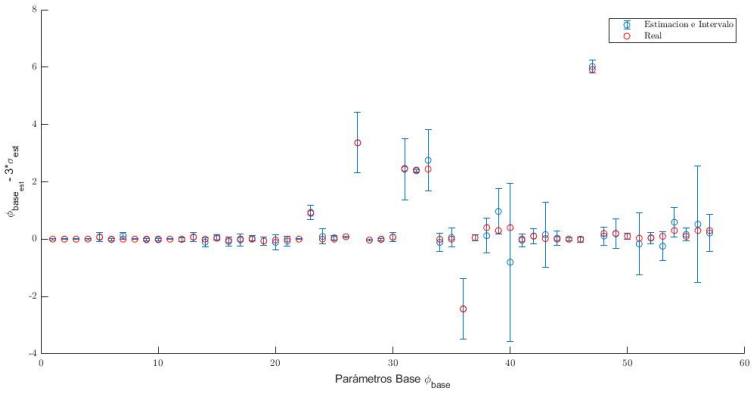


Figura 5.8: Intervalo de error de parámetros de trayectoria 1

Histogramas de los torques, Tau medidos

Para la evaluación del modelo construido se ha estimado del residuo, de los torques, entre los valores estimados y los medidos en el experimento. Cabe mencionar que para el manejo estadístico correcto y debido a las diferencias en el orden de magnitud de los rangos de movimientos de cada uno de los ejes se ha realizado el residuo de cada eje de forma individual.

De los resultados se aprecia que todos los ejes presentan una distribución normal acorde con lo esperado y que nos brindan una métrica de lo bueno o malo que puede ser los parámetros estimados.

Se han representado los residuos, en donde se puede apreciar que la figura 5.10, en donde se estimo la matriz de varianzas presenta menos un poco menos de error para algunos ejes que la varianza impuesta, ver figura 5.9.

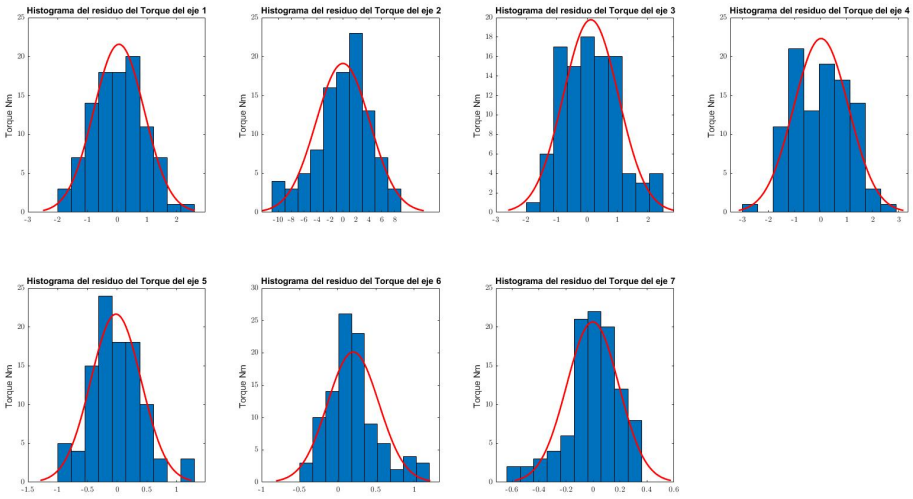


Figura 5.9: Histograma del Residuo de los Torques trayectoria 1.

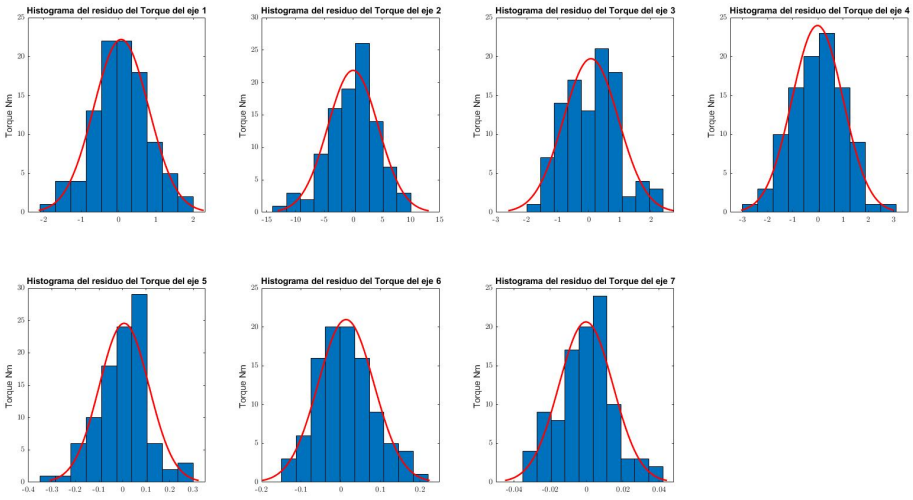


Figura 5.10: Histograma del residuo de los Torques trayectoria 1 empelando MV

De lo anterior, se define que se trabajará con los resultados proporcionados con la estimación de las varianza a través del M.V.

Intervalo de confianza empleado 3σ.

En la figura 5.11 se muestra el intervalo de confianza de los torques estimados, así como los torques medidos.

Cabe mencionar que el nivel de ruido introducido en los torques ha sido de un 1 % por ciento de error de su valor máximo, y la matriz de varianza se ha obtenido de forma iterativa mediante MV.

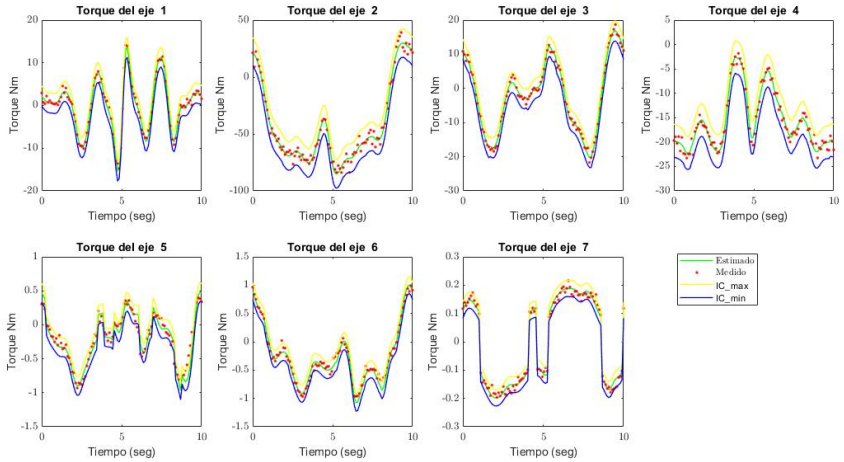


Figura 5.11: Torques de los ejes,estimados, medidos e Intervalo de Confianza de la trayectoria 1.

Para la Trayectoria de Validación

Para la segunda trayectoria se empelará la Estimación Máximo Verosímil para sistemas lineales y estimar la desviación típica del error de medición.

En la figura 5.12 se muestran los torques medidos y con ruido

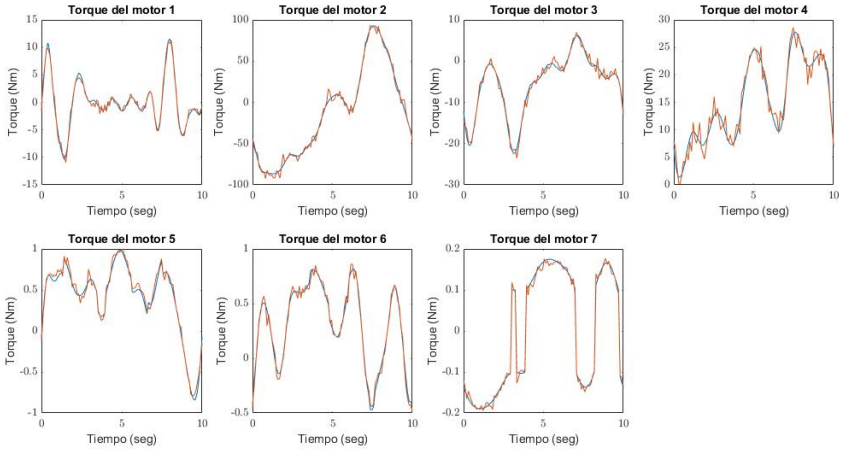


Figura 5.12: Medición de Torques con Ruido validación

En las figura 5.13 se muestran las posiciones medidas [º] y con ruido de la trayectoria de validación.

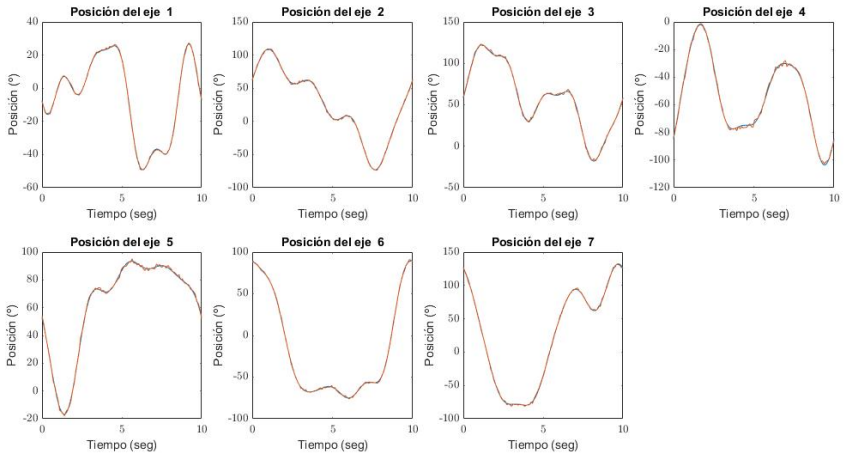


Figura 5.13: Medición de Posiciones con Ruido validación

Aproximación a una serie finita de Fourier los datos medidos

Ahora se aproxima con las series finitas de Fourier los datos medidos como se muestra en la figura 5.14.

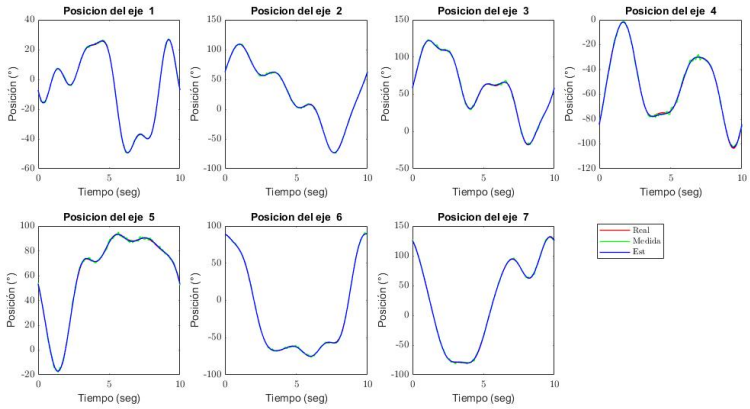


Figura 5.14: Posición de los ejes de la trayectoria 2

De la misma manera que se ha hecho con las posiciones, se realiza con las velocidades y aceleraciones de los ejes, ver figura 5.15

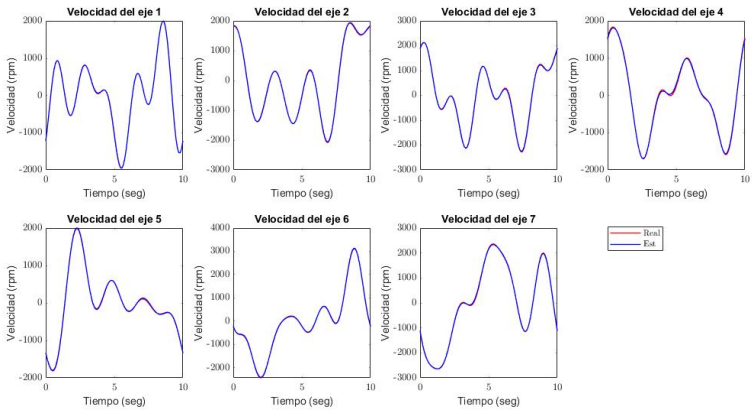


Figura 5.15: velocidades de los ejes de la trayectoria 2

Estimación de los parámetros

Para los parámetros se han comparado con los valores reales ver figura 5.16, así como la gráfica del intervalo de error que se muestra en la figura 5.17. Una última comparación entre ambas trayectorias solamente para comparar resultados, se observa en la figura 5.18.

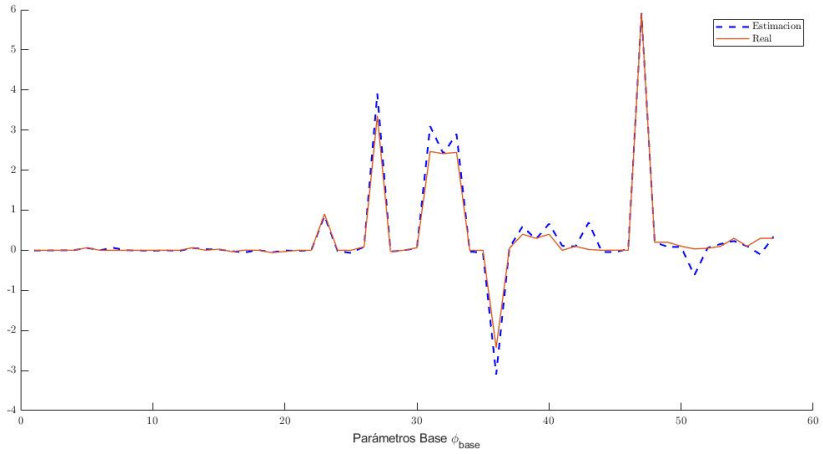


Figura 5.16: Parámetros base estimados vrs reales trayectoria 2 con MV

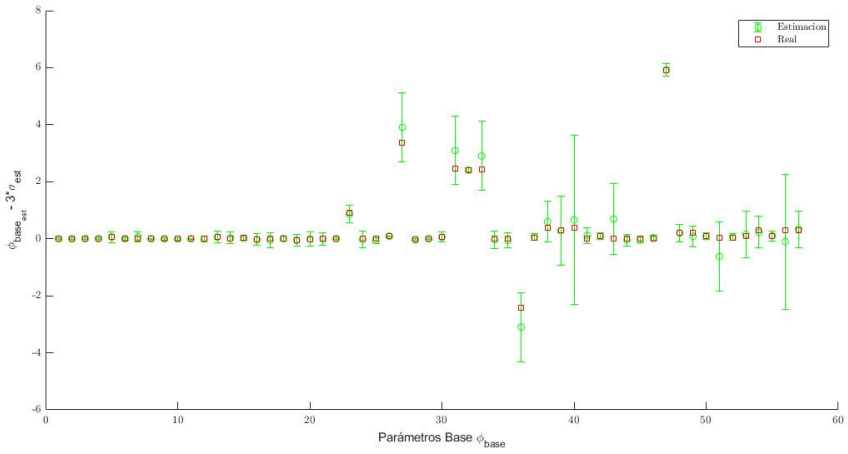


Figura 5.17: Intervalo de error de los parámetros de la trayectoria 2 con MV

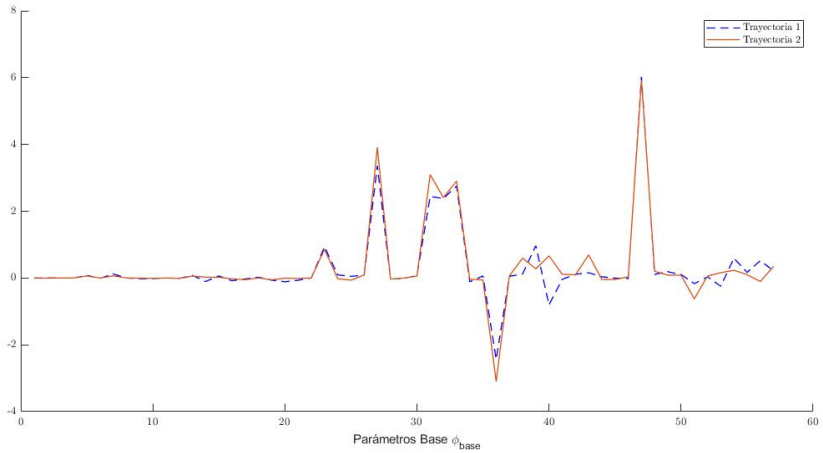


Figura 5.18: Comparación entre los parámetros estimados de las dos trayectorias

Histogramas de los torques, Tau medidos

Ahora se observa que la trayectoria de validación también presenta una distribución normal en el residuo como se observa en la figura 5.19.

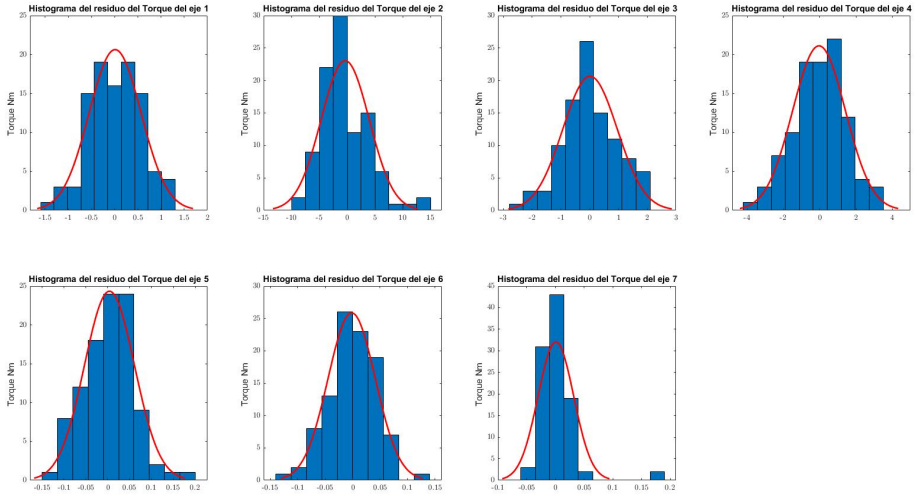


Figura 5.19: Histograma del residuo de los Torques trayectoria 2 MV

Intervalo de confianza empleado. 3σ .

En la figura 5.20 se muestra el intervalo de confianza de los torques estimados, así como los torques medidos de la trayectoria de validación. Se observa que presenta muy buenos resultados ya que se encuentra dentro del intervalo.

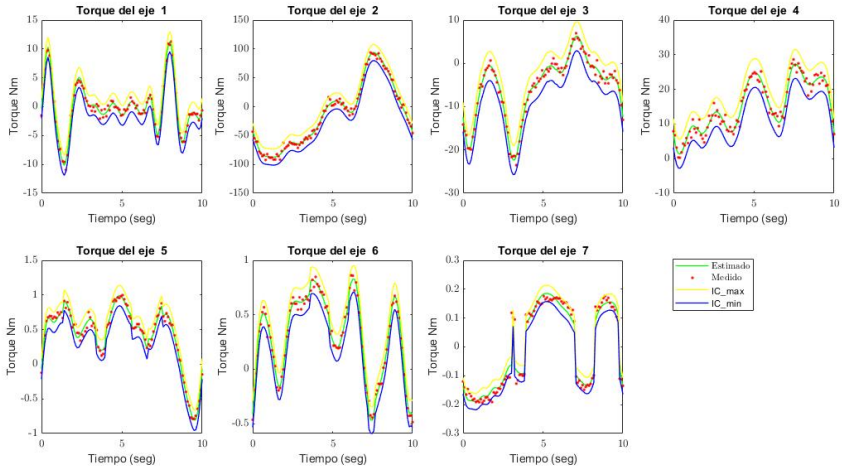


Figura 5.20: Torques de los ejes,validación, medidos e Intervalo de Confianza de la trayectoria 2

Validación cruzada utilizando dos trayectorias optimizadas

Una vez que se han evaluado el comportamiento de los parámetros estimados, intervalos de confianza y residuos de cada trayectoria por separado.

Es momento de evaluar la capacidad de generalidad que posee el modelo, para ello se a llevado a cabo una validación cruzada entre la información propuesta por las dos trayectorias. Como ya se explico en el apartado 5.5 aplicando las ecuaciones 5.13 y 5.14, es posible evaluar el residuo resultante de la trayectoria de validación empleando los parámetros estimados con la trayectoria primera.

En la figura 5.21, se puede observar que a diferencia de los residuos de las trayectorias por separado, este sigue teniendo una distribución normal, pero algo distorsionada para algunos ejes más que otros, siempre con mayor predominancia la media cero, pero si existen valores que son distintos a meda cero y se elevan más que la misma.

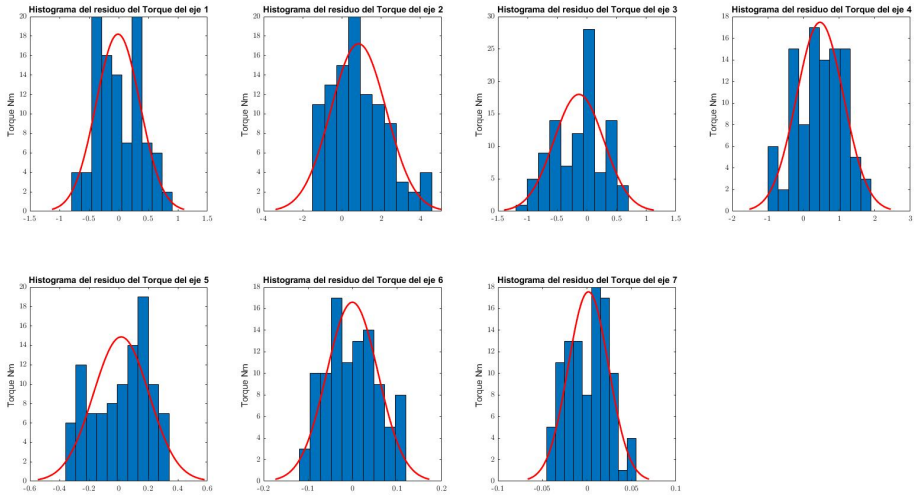


Figura 5.21: Histograma del residuo validación cruzada

La validación se llevo a cabo con la rutina *Main_Experimentos.m* ver Anexo E.

6. CONCLUSIONES

En la presente y última sección de este trabajo se enumerarán el conjunto de conclusiones obtenidas de las diferentes etapas del trabajo de fin de máster, se ha tratado de seguir la misma estructura que en el trabajo para plantear las conclusiones.

6.1. Modelado de Sistemas Mecánicos

- Respecto al modelado del robot, se han seleccionado como puntos de interés la unión entre dos sólidos (juntas cinemáticas), en donde se han definido las bases, los movimientos permitidos y los marcos de referencia de cada sólido.
- De la parametrización, se considera haber realizado una buena selección de las coordenadas generalizadas y los parámetros del robot.
- Con la aplicación del torsor de inercia en un punto perteneciente al sólido pero distinto del centro de gravedad y la selección de los modelos de fricción Viscosa y de Coulomb, se logró construir el modelo del robot y obtener las ecuaciones dinámicas de forma lineal respecto de los parámetros dinámicos.
- Después de haber realizado y analizado el conjunto de simulaciones dinámicas para el robot con 3 sólidos como un péndulo (modelo simplificado) o el robot completo, se concluye que el modelo construido sí representa los efectos dinámicos significativos esperados.

6.2. Reducción del Modelo Dinámico

- Para llevar a cabo la estimación de los parámetros dinámicos del robot, fue necesaria una reducción de los 84 parámetros inerciales y de fricción del modelo a 57 parámetros denominados parámetros base, estos últimos se generan como combinaciones lineales de los originales y son los que realmente se han estimado he identificado.
- La reducción realizada, se puede traducir diciendo que matemáticamente existen columnas de la matriz de observación que son linealmente dependientes de otras, y físicamente, que hay parámetros dinámicos que no tienen influencia sobre el robot y si la tienen, es como una combinación lineal de varios parámetros.
- En este trabajo, se ha empleado un método basado en sustituciones para crear la matriz de observaciones K a partir de las ecuaciones dinámicas, logrando un ahorro computacional considerable, que se traduce en un menor tiempo del cálculo en la creación de la matriz de observaciones. La aplicación de este método fue posible, porque las ecuaciones dinámicas del robot se plantearon lineal a los parámetros dinámicos (inerciales y de fricción).

6.3. Experimentos

- Respecto al diseño de los experimentos este trabajo se a dedicado en mayor parte a la labor previa a los experimentos, que consiste en generar una trayectoria lo suficientemente excitante y optimizada para obtener la mejor información de los experimentos.
- El utilizar una función de trayectoria parametrizada para la excitación del robot, permite optimizar la matriz de observaciones a partir de dichos parámetros (de trayectoria).
- Cabe mencionar, que tiene sentido que la optimización se centre en mejorar las propiedades de la matriz de observación, ya qué, es la que contiene todo lo relevante al movimiento del robot (incluyendo parámetros de trayectoria) y dicha optimización se refleja en una mejora en la estimación de los parámetros.
- Se hace énfasis en que el modelo construido con la librería 3D_MEC_MATLAB, es el necesario para llevar a cabo la estimación e identificación de los parámetros. Ya que, es de donde se obtienen las ecuaciones dinámicas del robot y en el que se basa el trabajo. El segundo modelo, fue construido en Simulink partiendo de un archivo CAD y fue utilizado para la visualización de la trayectoria y cómo otra opción para simular el experimento.

- A modo de comparación, entre el modelo construido con la librería de 3D_MEC_MATLAB y con la herramienta de Simulink se expresa que, son muy similares respecto a su comportamiento dinámico. El modelo de Simulink permite cambiar los atributos de las articulaciones como rigidez y amortiguación no considerados en el primer modelo. Ambos nos permiten simular el experimento obteniendo pequeñas discrepancias en el vector de pares de los actuadores para el eje 5 y 7, ya sean como errores en el orden de magnitud o ligeras ganancias.

6.4. Validación del Modelo

- Comentar, que por el origen de los datos que se han tomado como "reales" se arrastra un error, debido a que la distribución de masas de los sólidos creados no representan con exactitud al robot real. Aunque el error asociado a la geómetra fue reducido al emplear un modelo CAD muy preciso.
- De la estimación, se han obtenido resultados muy satisfactorios considerando que el modelo empleado fue virtual, me hace pensar que esto favoreciera al modelo, aunque considero que al realizar los experimentos y coger los datos experimentales reales de forma correcta también se obtendrán buenos resultados.
- Los valores de los pares de los actuadores obtenidos en el experimento virtual se encuentran en un orden de magnitud razonable y dentro de los limites de las especificaciones del robot.
- Los residuos, o errores en la estimación de los pares, siguen una distribución normal para ambas trayectorias y la discrepancia que hay entre los valores "medidos" y los estimados es muy baja, lo que se traduce en que el modelo se ajusta muy bien a los datos.
- Para el intervalos de confianza de ambas trayectorias se obtuvieron muy buenos resultados, en donde los valores medidos y estimados se ubicaban dentro de dicho intervalo a lo largo de toda la trayectoria en todos los ejes.
- Aunque la comparación directa de los parámetros estimados de dos trayectorias distintas no es una métrica que refleje directamente lo bueno o malo del modelo, se han obtenido resultados muy parecidos entre las dos.
- Aplicando Validación Cruzada al modelo entre dos trayectoria completamente distintas, se obtienen residuos con una ligera distorsión de la campana en la distribución normal, esto puede ocurrir por pequeños errores en nuestro modelo o al ruido existente en los experimentos.
- Es de resaltar que los parámetros estimados son los parámetros base del modelo reducido, que contienen a los parámetros dinámicos del robot.

7. BIBLIOGRAFÍA Y REFERENCIAS

Referencias

[1] KUKA. KUKA Sensitive Robotics LBR iiwa, 2017

[2] KUKA. Material de las especificaciones del Robot LBR iiwa 7R 800/ 14R 820, 2015.

[3] Iriarte Goñi , Xabier. "Identificación de robots manipuladores: reducción de modelos y diseño de experimentos (Tesis Doctoral), 2010"

[4] Ros, Javier. "Procedimientos en Dinámica de Sistemas Multicuerpo", 2019.

[5] Ros, Javier. Librería 3D_MEC implementada en MATLAB, 2019. https://manualzilla.com/doc/6278676/manual-de-usuario-de-3d-mec-1.99.19beta--programa-para-el

[6] Iriarte Goñi , Xabier. Apuntes de asignatura, Identificación de Sistemas dinámicos, Curso 2020-2021

[7] MATLAB, sitio web, https://www.mathworks.com/help/matlab

[8] Yvonne R. Sturz, Lukas M. Affolter, Roy S. Smirth,"Parameter identificationof the KUKU LBR iiwa robot including constraints on physical feasibility" IFAC PapersOnLine 50-1(2017)6863-6868.

[9] GrabCAD, sitio web, https://grabcad.com/library.

[10] Tian Xu, Jizhuang, Yiwen Chenz, Xyanyao NG, Marcelo H. Ang, Jr. Qianqian Fang, Yanhe Zhu and Jie Zhao. "Dynamic Identification of the KUKA LBR iiwa Robot With Retrieval of Physical Parameters Using Global Optimization", Digital Object Identifier 10.1109/ACCESS.2020.3000997

ANEXOS

A. Código del Modelado

```
addpath(genpath('..\Lib_3D_MEC_MATLAB'))
%addpath(genpath('Modelo_Robot'))
Do_you_want_to_run_the_script_again

%%Clean environment
 close all
 clear all
 clc

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% Library lib_3D_MEC_MATLAB initialization
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%%Init lib_3D_MEC_MATLAB environment
init_lib_3D_MEC_MATLAB;
disp('Remember to purgeDraws if drawing functions change');

%----Begin Edit----
g=newParam('g',9.80665);
GravityVector=Vector3D([0,0,-g]','xyz');

%Integration step
Delta_t=0.005;

%Graphical Output refresh freq (n_frames/n_steps)
% Delta_t_refresh=1/10;
Delta_t_refresh=1/24;
% Assembly Init Problem solver parameters
Geom_Eq_init_tol=1.0e-10;
Geom_Eq_init_relax=0.1;

% Assembly Problem solver parameters
Geom_Eq_tol=Delta_t^2*10^-3;
Geom_Eq_relax=0.1;

% Equilibrium Problem solver parameters
Dyn_Eq_tol=1.0e-10;
Dyn_Eq_relax=0.1;

% Perturbed Dynamic State solver parameters
Per_Dyn_State_tol=1.0e-12;
Per_Dyn_State_relax=0.1;

% pop_up_dialogs=false;
has_my_contact_forzes=false;
optimize_matlabFunction=false;
updateDraws_automatic=true;
equilibrium_problem_solved=false;
pop_up_dialogs=false;
Set_characteristic_size(1);
n_traj=0;
%----End Edit----

purgeMSDFunctions; purgeDraws;

fig_dir='fig';
mkdir(fig_dir);

dyn_equations_type='All';

Set_value(t,0);

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% Symbolic Procedures in Multibody Dynamics
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
```

```
% Kinematics
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% Parametrization
%-------------------------------------------------------------------------

%%Define generalized coordinates, velocities and accelerations
%----Begin Edit----

newCoord('t1',0,0); % Giro del S1 respecto la base en el eje vertical (Z)
newCoord('t2',0.7*pi,0); % Giro del S2 respecto al S1, (Y)
newCoord('t3',0,0); % Giro del S3 respecto al S2, (z)
newCoord('t4',0,0); % Giro del S4 respecto al S3, (Y)
newCoord('t5',0,0); % Giro del S5 respecto al S4, (z)
newCoord('t6',0,0); % Giro del S6 respecto al S5, (Y)
newCoord('t7',0,0); % Giro del S7 respecto al S6, (z)

%----End Edit----
q ,dq ,ddq % Vectores de Coordenadas, velocidades y aceleraciones generalizadas
n_q, n_dq, n_ddq % N mero de elementos de los vectores

%%Define Geometric Parameters
%----Begin Edit----
newParam('h0',0.1575); % Altura del s lido 0
newParam('h1',0.2025); % Altura del s lido 1
newParam('h2',0.2045); % Altura del s lido 2
newParam('h3',0.2155); % Altura del s lido 3
newParam('h4',0.1845); % Altura del s lido 4
newParam('h5',0.2155); % Altura del s lido 5
newParam('h6',0.0809); % Altura del s lido 6
newParam('L6',0.0607); % Altura del s lido 6
newParam('h7',0.0451); % Altura del s lido 7
%----End Edit----

param,n_param %Vector de par metros geometricos y gravedad, n mero de elementos
    del vector


%%Define Bases, Points and Frames

%----Begin Edit----
%Defining Bases (B)
%Utilizando el prefijo B, para denominar las Bases
newBase('xyz','B1',3,t1); % (Base previa,nueva Base,Orientaci n, Coordenada )
newBase('B1','B2',2,t2);
newBase('B2','B3',3,t3);
newBase('B3','B4',2,t4);
newBase('B4','B5',3,t5);
newBase('B5','B6',2,t6);
newBase('B6','B7',3,t7);


GravityVector=Vector3D([0,0,-g]','xyz');%Gravity redefined

%Defining Points (P)
%Se utiliza el prefijo 'P' para determinar los Puntos de interes
newPoint('O','P1',Vector3D([0,0,h0]','xyz'));% newpoint(nombre punto previo,nombre
    nuevo punto,Vector de posici n)
newPoint('P1','P2',Vector3D([0,0,h1]','B1'));% Vector3D(valor, base)
newPoint('P2','P3',Vector3D([0,0,h2]','B2'));
newPoint('P3','P4',Vector3D([0,0,h3]','B3'));
newPoint('P4','P5',Vector3D([0,0,h4]','B4'));
newPoint('P5','P6',Vector3D([0,L6,h5]','B5'));
newPoint('P6','P7',Vector3D([0,-L6,h6]','B6'));
newPoint('P7','P_end',Vector3D([0,0,h7]','B7'));

%Defining Inertial Frame

%newFrame('O','xyz','Gr'); % newFrame(nombre punto,nombre base,nombre marco)
newFrame('O','xyz','IF');

%Defining Solids Frames (F)
%Se utiliza el prefijo 'F' para indicar los marcos de referencia
%newFrame('O','xyz','IF');
newFrame('P1','B1','F1');
newFrame('P2','B2','F2');
```

```
newFrame('P3','B3','F3');
newFrame('P4','B4','F4');
newFrame('P5','B5','F5');
newFrame('P6','B6','F6');
newFrame('P7','B7','F7');
newFrame('P_end','B7','Fend');
%----End Edit----

%%Define Solids
% Propiedades de masa, centro de masa y componentes del tensor de inercia
% de los s lidos
    [mass,center_of_mass,inertia_tensor] = get_Propiedades_Inercia_Sw3

    % Creaci n y asignaci n de las variables simb licas de las propiedades de
    inercia
    for i=0:7
        newParam(['m',num2str(i)],mass{i+1});
        newParam(['mx',num2str(i)],center_of_mass{i+1}(1,1));
        newParam(['my',num2str(i)],center_of_mass{i+1}(2,1));
        newParam(['mz',num2str(i)],center_of_mass{i+1}(3,1));
        newParam(['Ixx',num2str(i)],inertia_tensor{i+1}(1,1));
        newParam(['Ixy',num2str(i)],inertia_tensor{i+1}(1,2));
        newParam(['Ixz',num2str(i)],inertia_tensor{i+1}(1,3));
        newParam(['Iyy',num2str(i)],inertia_tensor{i+1}(2,2));
        newParam(['Iyz',num2str(i)],inertia_tensor{i+1}(2,3));
        newParam(['Izz',num2str(i)],inertia_tensor{i+1}(3,3));
    end

    % Creaci n de los vectores 3D de los primeros momentos
    first_mass_moment_0=Vector3D([mx0;my0;mz0],'xyz');
    first_mass_moment_1=Vector3D([mx1;my1;mz1],'B1');
    first_mass_moment_2=Vector3D([mx2;my2;mz2],'B2');
    first_mass_moment_3=Vector3D([mx3;my3;mz3],'B3');
    first_mass_moment_4=Vector3D([mx4;my4;mz4],'B4');
    first_mass_moment_5=Vector3D([mx5;my5;mz5],'B5');
    first_mass_moment_6=Vector3D([mx6;my6;mz6],'B6');
    first_mass_moment_7=Vector3D([mx7;my7;mz7],'B7');
    % Creaci n de los vectores 3D del tensor de Inercia
    inertia_tensor_0=Vector3D([Ixx0,Ixy0,Ixz0;Ixy0,Iyy0,Iyz0;Ixz0,Iyz0,Izz0],'xyz')
    ;
    inertia_tensor_1=Vector3D([Ixx1,Ixy1,Ixz1;Ixy1,Iyy1,Iyz1;Ixz1,Iyz1,Izz1],'B1');
    inertia_tensor_2=Vector3D([Ixx2,Ixy2,Ixz2;Ixy2,Iyy2,Iyz2;Ixz2,Iyz2,Izz2],'B2');
    inertia_tensor_3=Vector3D([Ixx3,Ixy3,Ixz3;Ixy3,Iyy3,Iyz3;Ixz3,Iyz3,Izz3],'B3');
    inertia_tensor_4=Vector3D([Ixx4,Ixy4,Ixz4;Ixy4,Iyy4,Iyz4;Ixz4,Iyz4,Izz4],'B4');
    inertia_tensor_5=Vector3D([Ixx5,Ixy5,Ixz5;Ixy5,Iyy5,Iyz5;Ixz5,Iyz5,Izz5],'B5');
    inertia_tensor_6=Vector3D([Ixx6,Ixy6,Ixz6;Ixy6,Iyy6,Iyz6;Ixz6,Iyz6,Izz6],'B6');
    inertia_tensor_7=Vector3D([Ixx7,Ixy7,Ixz7;Ixy7,Iyy7,Iyz7;Ixz7,Iyz7,Izz7],'B7');

   % Asignaci n de las propiedades inerciales a cada s lido
S0=newSolid('S0','IF',m0,first_mass_moment_0,inertia_tensor_0,'S0.stl');
S1=newSolid('S1','F1',m1,first_mass_moment_1,inertia_tensor_1,'S1.stl');
S2=newSolid('S2','F2',m2,first_mass_moment_2,inertia_tensor_2,'S2.stl');
S3=newSolid('S3','F3',m3,first_mass_moment_3,inertia_tensor_3,'S3.stl');
S4=newSolid('S4','F4',m4,first_mass_moment_4,inertia_tensor_4,'S4.stl');
S5=newSolid('S5','F5',m5,first_mass_moment_5,inertia_tensor_5,'S5.stl');
S6=newSolid('S6','F6',m6,first_mass_moment_6,inertia_tensor_6,'S6.stl');
S7=newSolid('S7','F7',m7,first_mass_moment_7,inertia_tensor_7,'S7.stl');

% Please note that in the previous calls the center of inertia of the
% solids as well as the solid inertial properties are defined.

%%Draw Points, Frames, Bodies, ...

%Point of view, lighting, material

azimut = 135.2000; elevation =20.2000;
 camzoom(3);
view(azimut,elevation);

% Draw the Solids and Frames

newFrameDraw('abs',0.7);
% newFrameDraw('F0',1);

%square_fv=square_patch(0.1*[-1,1,1,-1],0.1*[-1 -1 1 1],1*[0 0 0 0]);
%newFVDraw('abs',square_fv,[0.5,0.5,0.5]); clear square_fv
```

```
%newSolidDraw('Base2',[0.5,0.3,1]);


newSolidDraw('S0',[0.85,0.325,0.098]);
% newFrameDraw('IF',[0.3,0.3,0.3]);
newSolidDraw('S1',[0.5,0.5,0.5]);
% newFrameDraw('F1',[0.3,0.3,0.3]);
newSolidDraw('S2',[1,0.5,0]);
% newFrameDraw('F2',[0.3,0.3,0.3]);
newSolidDraw('S3',[0.5,0.5,0.5]);
% newFrameDraw('F3',[0.3,0.3,0.3]);
newSolidDraw('S4',[1,0.5,0]);
% newFrameDraw('F4',[0.3,0.3,0.3]);
newSolidDraw('S5',[0.5,0.5,0.5]);
% newFrameDraw('F5',[0.3,0.3,0.3]);
newSolidDraw('S6',[1,0.5,0]);
% newFrameDraw('F6',[0.3,0.3,0.3]);
newSolidDraw('S7',[0.5,0.5,0.5]);
% newFrameDraw('F7',[0.3,0.3,0.3]);
newFrameDraw('Fend',[0.1,0.1,0.1]);
%      %----End Edit----

disp('S lidos del sistema Dibujados')
material('dull');

save('extended_state','t_value','q_value','dq_value','param_value');


%%Draw Sketch of Parametrization
if exist('./draw_dimensions.m','file')
    %%Draw Dimensions
    azimut = 135.2000; elevation =20.2000;
    view(azimut,elevation);
    draw_dimensions;
    load('extended_state');
    updateDraws;
    Help(['Graphical Output shows the parametrization.', newline 'The situation is
    as-given (no Asembly or Initial Assembly problem solved)']);
    delete(dimensions_handle);
else
    Warning('You need to provide dimension drawing instructions on ''./
    draw_dimensions.m'' file') ;
end
%% h
disp('Solidos y bases dibujados')

%-------------------------------------------------------------------------
% Define Constraint Equations
%-------------------------------------------------------------------------
%% Constraint equations at the coordinate level (geometric)
%----Begin Edit----

%% Constraint equations at the velocity level (kinematic)
% a) Geometric equation derivative
% dPhiH=timederivative(Phi); Automaticaly added to dPhi;

% b) non-holonomic

Phi, dPhi, dPhiH, dPhiNH; % Se crean las variables simbolicas de phi, dPhiH y
    dphiNH
n_Phi, n_dPhi, n_dPhiH, n_dPhiNH;%Se crea el contador de las ecuaciones definidas
% Export Assembly Problem

Export_Assembly_Problem;

% Additional Eqs for initial assembly problem

% Coordinate level
%----Begin Edit----
% newPhiInit();
% ----End Edit----
%
% Velocity Level
% ----Begin Edit----
% newdPhiInit();
%----End Edit----
```

```
293
294
295 PhiInit, dPhiInit;
296 n_PhiInit, n_dPhiInit;
297
298 % Export Assembly Problem Init
299 Export_Initial_Assembly_Problem;
300
301 %%Save Screenshot
302 % fig_name=['Unassembled_State_Kinematics_t',num2str(t_value)];
303 % fig_name=[fig_dir,'/',fig_name];
304 % saveas(Graphical_Output,fig_name,'png');
305 % %%Assembly Problem init
306  initial_assembly_problem;
307 %
308 % %%Save Screenshot
309 % fig_name=['Assembled_State_Kinematics_t',num2str(t_value)];
310 % fig_name=[fig_dir,'/',fig_name];
311 % saveas(Graphical_Output,fig_name,'png');
312
313 save('kinematic_extended_state','t_value','q_value','dq_value','param_value')
314
315
316 %Help('Graphical Output shows the situation as given (no Asembly or  Initial
        Assembly problem solved)');
317 %Grafica el sistema con la condiciones iniciales definidas
318 % %% Assembly Problem init
319 % initial_assembly_problem;
320 %
321 % %Help('Graphical Output shows the situation, after solving Initial Assembly
        problem');
322
323 %-------------------------------------------------------------------------
324 % Dynamics
325 %-------------------------------------------------------------------------
326 %
327 % %%Wrench Definition
328 % %----Begin Edit----
329 %
330 % % Constraint Wrenches (Definition using helper function newConstraint_Wrench)
331 % S0->S1
332 % Wrenchc1 = newConstraint_Wrench([1,1,1],[1,1,0],'P1','B1','S1','S0');
333 % S1->S2
334 % Wrenchc2 = newConstraint_Wrench([1,1,1],[1,0,1],'P2','B2','S2','S1');
335 % S2->S3
336 % Wrenchc3 = newConstraint_Wrench([1,1,1],[1,1,0],'P3','B3','S3','S2');
337 % S3->S4
338 % Wrenchc4 = newConstraint_Wrench([1,1,1],[1,0,1],'P4','B4','S4','S3');
339 % S4->S5
340 % Wrenchc5 = newConstraint_Wrench([1,1,1],[1,1,0],'P5','B5','S5','S4');
341 % S5->S6
342 % Wrenchc6 = newConstraint_Wrench([1,1,1],[1,0,1],'P6','B6','S6','S5');
343 % S6->S7
344 % Wrenchc7 = newConstraint_Wrench([1,1,1],[1,1,0],'P7','B7','S7','S6');
345 %
346 % ----End Edit----
347 epsilon, n_epsilon % Incognitas de Fuerzas y momentosa de enlace
348
349 % Constitutive
350 %----Begin Edit----
351 tic
352 % Viscous Gr->S1
353 %OBS: Se han creado dos wrenchs para cada s lido, su acci n y Reaccion
        respectivamente.
354 %Vector6D(magnitud_fuerza,magnitud_momento,'Punto_solido','Acci n_s lido','
        Reaccion_s lido')
355
356 c1=newParam('c1',0.2); % Par metros de fricci n viscosa entre s lidos
357 cc1=newParam('cc1',0.2); % Par metros de fricci n de Coulumb entre s lidos
358
359 FV_S0_S1=Vector3D([0,0,0]','B1'); %Se define las fuerzas de fricc n.
360 MV_S0_S1_P1=Vector3D([0,0,(-c1*dt1-cc1*sign(dt1))]','B1'); %Se define el momento de
        fricc n
361 WrenchV_S0_S1=Vector6D(FV_S0_S1,MV_S0_S1_P1,'P1','S1','S0'); % Se construye el
        wrench con la fuerza y momento...
362 WrenchV_S1_S0=Vector6D(-FV_S0_S1,-MV_S0_S1_P1,'P1','S0','S1'); % Se construye el
```

```
    wrench con la fuerza y momento...
363
364
365 % Viscous S1->S2
366 c2=newParam('c2',0.2);
367 cc2=newParam('cc2',0.1);
368
369 FV_S1_S2=Vector3D([0,0,0]','B2'); %Se define las fuerzas de fricc n.
370 MV_S1_S2_P2=Vector3D([0,(-c2*dt2-cc2*sign(dt2)),0]','B2'); %Se define el momento de
      fricc n
371 WrenchV_S1_S2=Vector6D(FV_S1_S2,MV_S1_S2_P2,'P2','S2','S1'); % Se construye el
    wrench con la fuerza y momento...
372 WrenchV_S2_S1=Vector6D(-FV_S1_S2,-MV_S1_S2_P2,'P2','S1','S2'); % Se construye el
    wrench con la fuerza y momento...
373
374 % Viscous S2->S3
375 c3=newParam('c3',0.1);
376 cc3=newParam('cc3',0.1);
377
378 FV_S2_S3=Vector3D([0,0,0]','B3'); %Se define las fuerzas de fricc n.
379 MV_S2_S3_P3=Vector3D([0,0,(-c3*dt3-cc3*sign(dt3))]','B3'); %Se define el momento de
      fricc n
380 WrenchV_S2_S3=Vector6D(FV_S2_S3,MV_S2_S3_P3,'P3','S3','S2'); % Se construye el
    wrench con la fuerza y momento...
381 WrenchV_S3_S2=Vector6D(-FV_S2_S3,-MV_S2_S3_P3,'P3','S2','S3'); % Se construye el
    wrench con la fuerza y momento...
382
383 % Viscous S3->S4
384 c4=newParam('c4',0.1);
385 cc4=newParam('cc4',0.1);
386
387 FV_S3_S4=Vector3D([0,0,0]','B4'); %Se define las fuerzas de fricc n.
388 MV_S3_S4_P4=Vector3D([0,(-c4*dt4-cc4*sign(dt4)),0]','B4'); %Se define el momento de
      fricc n
389 WrenchV_S3_S4=Vector6D(FV_S3_S4,MV_S3_S4_P4,'P4','S4','S3'); % Se construye el
    wrench con la fuerza y momento...
390 WrenchV_S4_S3=Vector6D(-FV_S3_S4,-MV_S3_S4_P4,'P4','S3','S4'); % Se construye el
    wrench con la fuerza y momento...
391
392 % Viscous S4->S5
393 c5=newParam('c5',0.2);
394 cc5=newParam('cc5',0.1);
395
396 FV_S4_S5=Vector3D([0,0,0]','B5'); %Se define las fuerzas de fricc n.
397 MV_S4_S5_P5=Vector3D([0,0,(-c5*dt5-cc5*sign(dt5))]','B5'); %Se define el momento de
      fricc n
398 WrenchV_S4_S5=Vector6D(FV_S4_S5,MV_S4_S5_P5,'P5','S5','S4'); % Se construye el
    wrench con la fuerza y momento...
399 WrenchV_S5_S4=Vector6D(-FV_S4_S5,-MV_S4_S5_P5,'P5','S4','S5'); % Se construye el
    wrench con la fuerza y momento...
400
401
402 % Viscous S5->S6
403 c6=newParam('c6',0.1);
404 cc6=newParam('cc6',0.1);
405
406 FV_S5_S6=Vector3D([0,0,0]','B6'); %Se define las fuerzas de fricc n.
407 MV_S5_S6_P6=Vector3D([0,(-c6*dt6-cc6*sign(dt6)),0]','B6'); %Se define el momento de
      fricc n
408 WrenchV_S5_S6=Vector6D(FV_S5_S6,MV_S5_S6_P6,'P6','S6','S5'); % Se construye el
    wrench con la fuerza y momento...
409 WrenchV_S6_S5=Vector6D(-FV_S5_S6,-MV_S5_S6_P6,'P6','S5','S6'); % Se construye el
    wrench con la fuerza y momento...
410
411 % Viscous S6->S7
412 c7=newParam('c7',0.2);
413 cc7=newParam('cc7',0.1);
414
415 FV_S6_S7=Vector3D([0,0,0]','B7'); %Se define las fuerzas de fricc n.
416 MV_S6_S7_P7=Vector3D([0,0,(-c7*dt7-cc7*sign(dt7))]','B7'); %Se define el momento de
      fricc n
417 WrenchV_S6_S7=Vector6D(FV_S6_S7,MV_S6_S7_P7,'P7','S7','S6'); % Se construye el
    wrench con la fuerza y momento...
418 WrenchV_S7_S6=Vector6D(-FV_S6_S7,-MV_S6_S7_P7,'P7','S6','S7'); % Se construye el
    wrench con la fuerza y momento...
419
```

```
%----End Edit----

% Gravity + Inertia

%----Begin Edit----
GravityVector=Vector3D([0,0,-g]','xyz');

%%S1
WrenchIG_S1=newInertia_and_Gravity_Wrench('S1');
%%S2
WrenchIG_S2=newInertia_and_Gravity_Wrench('S2');
%%S3
WrenchIG_S3=newInertia_and_Gravity_Wrench('S3');
%%S4
WrenchIG_S4=newInertia_and_Gravity_Wrench('S4');
%%S5
WrenchIG_S5=newInertia_and_Gravity_Wrench('S5');
%%S6
WrenchIG_S6=newInertia_and_Gravity_Wrench('S6');
%%S7
WrenchIG_S7=newInertia_and_Gravity_Wrench('S7');

%----End Edit----

% Externa

%----Begin Edit----

% Motor S0->S1
M1=newParam('M1',0);

FM_S0_S1=Vector3D([0,0,0]','B1'); %Se define las fuerzas asociadas al motor.
MM_S0_S1_P1=Vector3D([0,0,M1]','B1'); %Se define el momento externo aplicado por el
    motor.
WrenchM_S0_S1=Vector6D(FM_S0_S1,MM_S0_S1_P1,'P1','S1','S0'); % Se construye el
    wrench con la fuerza y momento...
WrenchM_S1_S0=Vector6D(-FM_S0_S1,-MM_S0_S1_P1,'P1','S0','S1'); % Se construye el
    wrench con la fuerza y momento...

% Motor S1->S2
M2=newParam('M2',0);

FM_S1_S2=Vector3D([0,0,0]','B2'); %Se define las fuerzas asociadas al motor.
MM_S1_S2_P2=Vector3D([0,M2,0]','B2'); %Se define el momento externo aplicado por el
    motor.
WrenchM_S1_S2=Vector6D(FM_S1_S2,MM_S1_S2_P2,'P2','S2','S1'); % Se construye el
    wrench con la fuerza y momento...
WrenchM_S2_S1=Vector6D(-FM_S1_S2,-MM_S1_S2_P2,'P2','S1','S2'); % Se construye el
    wrench con la fuerza y momento...

% Motor S2->S3
M3=newParam('M3',0);

FM_S2_S3=Vector3D([0,0,0]','B3'); %Se define las fuerzas asociadas al motor.
MM_S2_S3_P3=Vector3D([0,0,M3]','B3'); %Se define el momento externo aplicado por el
    motor.
WrenchM_S2_S3=Vector6D(FM_S2_S3,MM_S2_S3_P3,'P3','S3','S2'); % Se construye el
    wrench con la fuerza y momento...
WrenchM_S3_S2=Vector6D(-FM_S2_S3,-MM_S2_S3_P3,'P3','S2','S3'); % Se construye el
    wrench con la fuerza y momento...

% Motor S3->S4
M4=newParam('M4',0);

FM_S3_S4=Vector3D([0,0,0]','B4'); %Se define las fuerzas asociadas al motor.
MM_S3_S4_P4=Vector3D([0,M4,0]','B4'); %Se define el momento externo aplicado por el
    motor.
WrenchM_S3_S4=Vector6D(FM_S3_S4,MM_S3_S4_P4,'P4','S4','S3'); % Se construye el
    wrench con la fuerza y momento...
WrenchM_S4_S3=Vector6D(-FM_S3_S4,-MM_S3_S4_P4,'P4','S3','S4'); % Se construye el
    wrench con la fuerza y momento...

% Motor S4->S5
M5=newParam('M5',0);

FM_S4_S5=Vector3D([0,0,0]','B5'); %Se define las fuerzas asociadas al motor.
```

```
484 MM_S4_S5_P5=Vector3D([0,0,M5]','B5'); %Se define el momento externo aplicado por el
      motor.
485 WrenchM_S4_S5=Vector6D(FM_S4_S5,MM_S4_S5_P5,'P5','S5','S4'); % Se construye el
    wrench con la fuerza y momento...
486 WrenchM_S5_S4=Vector6D(-FM_S4_S5,-MM_S4_S5_P5,'P5','S4','S5'); % Se construye el
    wrench con la fuerza y momento...
487
488 % Motor S5->S6
489 M6=newParam('M6',0);
490
491 FM_S5_S6=Vector3D([0,0,0]','B6');%Se define las fuerzas asociadas al motor.
492 MM_S5_S6_P6=Vector3D([0,M6,0]','B6'); %Se define el momento externo aplicado por el
      motor.
493 WrenchM_S5_S6=Vector6D(FM_S5_S6,MM_S5_S6_P6,'P6','S6','S5'); % Se construye el
    wrench con la fuerza y momento...
494 WrenchM_S6_S5=Vector6D(-FM_S5_S6,-MM_S5_S6_P6,'P6','S5','S6'); % Se construye el
    wrench con la fuerza y momento...
495
496 % Motor S6->S7
497 M7=newParam('M7',0)
498
499 FM_S6_S7=Vector3D([0,0,0]','B7'); %Se define las fuerzas asociadas al motor.
500 MM_S6_S7_P7=Vector3D([0,0,M7]','B7'); %Se define el momento externo aplicado por el
      motor.
501 WrenchM_S6_S7=Vector6D(FM_S6_S7,MM_S6_S7_P7,'P7','S7','S6'); % Se construye el
    wrench con la fuerza y momento...
502 WrenchM_S7_S6=Vector6D(-FM_S6_S7,-MM_S6_S7_P7,'P7','S6','S7'); % Se construye el
    wrench con la fuerza y momento...
503
504
505
506 %%Draw Forces and Moments
507 %----Begin Edit----
508
509  MV_S1_S2=WrenchV_S1_S2.MV;
510  newVectorDraw('P2','MV_S1_S2',[2,2,2],[0.5,1,0.2]);
511  MV_S3_S4=WrenchV_S3_S4.MV;
512  newVectorDraw('P4','MV_S3_S4',[2,2,2],[0,1,0.3]);
513
514 %----End Edit----
515
516
517 %%Define Newton-Euler Dynamic Equations
518 % No se incluyen los Wrences debido a las fuerzas de enlase en el sistema
519 % using Screws (Vector6D)
520 %----Begin Edit----
521
522 % S0
523 %Sum_Wrenches_S0= WrenchIG_S0 + WrenchV_S1_S0 + WrenchM_S1_S0;
524
525 % S1
526 Sum_Wrenches_S1= WrenchIG_S1 + WrenchV_S0_S1 + WrenchV_S2_S1 + WrenchM_S0_S1 +
    WrenchM_S2_S1;
527 % Se realiza la sumatoria de todos los wrenches del S1 (gravedad, inercia,
    viscocidad,Motor)
528
529 % S2
530 Sum_Wrenches_S2= WrenchIG_S2 + WrenchV_S1_S2 + WrenchV_S3_S2 + WrenchM_S1_S2 +
    WrenchM_S3_S2;
531 % Se realiza la sumatoria de todos los wrenches del S2 (gravedad, inercia,
    viscocidad,Motor)
532
533 % S3
534 Sum_Wrenches_S3= WrenchIG_S3 + WrenchV_S2_S3+WrenchV_S4_S3 + WrenchM_S2_S3 +
    WrenchM_S4_S3;
535 % Se realiza la sumatoria de todos los wrenches del S3 (gravedad, inercia,
    viscocidad,Motor)
536
537 % S4
538 Sum_Wrenches_S4= WrenchIG_S4 + WrenchV_S3_S4 + WrenchV_S5_S4 + WrenchM_S3_S4 +
    WrenchM_S5_S4;
539 % Se realiza la sumatoria de todos los wrenches del S4 (gravedad, inercia,
    viscocidad,Motor)
540
541 % S5
542 Sum_Wrenches_S5= WrenchIG_S5 + WrenchV_S4_S5 + WrenchV_S6_S5 + WrenchM_S4_S5 +
```

```
    WrenchM_S6_S5;
% Se realiza la sumatoria de todos los wrenches del S5 (gravedad, inercia,
    viscocidad,Motor)

% S6
Sum_Wrenches_S6= WrenchIG_S6 + WrenchV_S5_S6 + WrenchV_S7_S6 + WrenchM_S5_S6 +
    WrenchM_S7_S6;
% Se realiza la sumatoria de todos los wrenches del S6 (gravedad, inercia,
    viscocidad,Motor)

% S7
Sum_Wrenches_S7= WrenchIG_S7  + WrenchV_S6_S7 + WrenchM_S6_S7;
% Se realiza la sumatoria de todos los wrenches del S7 (gravedad, inercia,
    viscocidad,  Motor)


%----End Edit----

%----Begin Edit----
% newDyn_eq_NE([Sum_Wrenches_S0;
%     Sum_Wrenches_S1;
%     Sum_Wrenches_S2;
%     Sum_Wrenches_S3;
%     Sum_Wrenches_S4;
%     Sum_Wrenches_S5;
%     Sum_Wrenches_S6;
%     Sum_Wrenches_S7]);
% %----End Edit----

%Export Direct Dynamics Problem Newton-Euler
%So mass matrix is positive definite
% Dyn_eq_NE=-Dyn_eq_NE; % Dyn_eq_NE=simplify(Dyn_eq_NE);
% Export_Direct_Dynamics_Problem_NE()

%%Virtual Power
%----Begin Edit----

%Twist_S0 = Twist('S0');
Twist_S1 = Twist('S1');
Twist_S2 = Twist('S2');
Twist_S3 = Twist('S3');
Twist_S4 = Twist('S4');
Twist_S5 = Twist('S5');
Twist_S6 = Twist('S6');
Twist_S7 = Twist('S7');


for i=1:n_q
%      newDyn_eq_VP(dot( Sum_Wrenches_S0 , diff( Twist_S0 , dq(i) ) )+...
        newDyn_eq_VP(dot( Sum_Wrenches_S1 , diff( Twist_S1 , dq(i) ) )+...
        dot( Sum_Wrenches_S2 , diff( Twist_S2 , dq(i) ) )+...
        dot( Sum_Wrenches_S3 , diff( Twist_S3 , dq(i) ) )+...
        dot( Sum_Wrenches_S4 , diff( Twist_S4 , dq(i) ) )+...
        dot( Sum_Wrenches_S5 , diff( Twist_S5 , dq(i) ) )+...
        dot( Sum_Wrenches_S6 , diff( Twist_S6 , dq(i) ) )+...
        dot( Sum_Wrenches_S7 , diff( Twist_S7 , dq(i) ) ));
end
%----End Edit----
toc
disp('Sumatoria de Wrenches y Twist realizada, pendiente Dyn_eq_VP')
tic
%%Export Direct Dynamics Problem "a la" Virtual Power or Lagrange
%So mass matrix is positive definite
%save( 'Ecuaciones_Dinamicas','Dyn_eq_VP')
% cd ..\Simb lico
%save( 'Ecuaciones_Dinamicas','Dyn_eq_VP')
% cd ..\Modelo_Robot

Dyn_eq_VP=-Dyn_eq_VP;

Export_Direct_Dynamics_Problem_VP
toc
disp('Ecuaciones Dinamicas Calculadas')
save('extended_state','t_value','q_value','dq_value','epsilon_value','lambda_value'
    ,'lambda_aux_value','param_value')

```

```
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%% Numerical Procedures in Multibody Dynamics
%*************************************************************************
%                       ---*SIMULACI N CINEM TICA*---                    *
%*************************************************************************
%%Load a saved Initial State (kinematical)
% load('extended_state')
% updateDraws;
% assembly_problem;
%
% %%Kinematic Simulation  (from current state)
% % Setup time step an integration span
% % Each time you run this section you get T_span additional time of
% % simulation
% disp('Iniciando Simulaci n Cinem tica')
% T_span=1
% t_end=Get_value(t)+T_span;
% t_value_ant=Get_value(t);
%
% fid = fopen('kinematics_extended_state_series.txt','w');
% printf_extended_state(fid);
%
% updateDraws_automatic=false;
%
% kinematic_simulation_Euler;
%
% updateDraws_automatic=true;
%
% fclose(fid);
%
% %%Save Screenshot
% fig_name=['Final_State_Simulation_t',num2str(t_value)];
% fig_name=[fig_dir,'/',fig_name];
% saveas(Graphical_Output,fig_name,'png');
%
% % %%Save Extended State
% %
  save('dynamic_extended_state','t_value','q_value','dq_value','epsilon_value','
    lambda_value','lambda_aux_value','param_value')

%% Virtual Power
%*************************************************************************
%          ---*DIN MICA DIRECTA Y SIMULACI N Virtual Power*---      *
%*************************************************************************
%%Direct dynamics Virtual Power
 % Choose Direct dynamics solver equation
dyn_equations_type='Virtual_Power',tic
disp('Resolviendo el problema de la Din mica Directa')
% Load a saved Initial State (dynamical)
 load('dynamic_extended_state');
 updateDraws;
 assembly_problem;
 direct_dynamics_problem;

% Save Extended State
save('dynamic_extended_state','t_value','q_value','dq_value','epsilon_value','
    lambda_value','lambda_aux_value','param_value')
%save ('Dyn_eq_VP2','Dyn_eq_VP')

% Save screenshot
fig_name=['Initial_State_Dynamic_Simulation_t',num2str(t_value)];
fig_name=[fig_dir,'/',fig_name];
saveas(Graphical_Output,fig_name,'png');

% Dynamic Simulation
% Setup time step and integration span
% Each time you run this section you get T_span additional time of
% simulation
disp('Iniciando Simulaci n Din mica')
T_span=4
toc
t_end=Get_value(t)+T_span;
t_value_ant=Get_value(t);

Geom_Eq_tol=Delta_t^2*10^-3;
```

```
Geom_Eq_relax=1;

assembly_problem;
direct_dynamics_problem;

updateDraws_automatic=false;

fid = fopen('dynamics_extended_state_series.txt','w');
printf_extended_state(fid);

dynamic_simulation_Euler;
fclose(fid);
updateDraws_automatic=true;

% save('Q_DQ_DDQ_VALUE_Direct','q_value','dq_value','ddq_value')
% q_value
% dq_value
% ddq_value
%
% %%Save Screenshot
% fig_name=['Final_State_Simulation_t',num2str(t_value)];
% fig_name=[fig_dir,'/',fig_name]
% saveas(Graphical_Output,fig_name,'png');
```

B. Código Simbólico

Main_Symbolic

```
%main_symbolic
close all
clear all
clc
format compact

% Creaci n de las variables simb licas globales

syms  t g f real
syms h0 h1 h2 h3 h4 h5 h6 h7 L6 real
syms M1 M2 M3 M4 M5 M6 M7  real
%fprintf('N mero de grados de libertad y n mero de s lidos de mi modelo f sico\
    n\n')

% N mero de grados de libertad y N mero de s lidos de nuestro modelo din mico a
     representar
[N_gdl,N_sol] =get_N_gdl_N_sol()

% Creaci n del vector simb lico de los Par metros inerciales de cada s lido  y
    de fricci n de los enlaces.
[phi]=get_phi(N_gdl,N_sol);

%         fprintf('Se realizar  el montaje de la matriz K\n\n')
%         % Ecuaciones del modelo creado
         load ('Ecuaciones_Dinamicas.mat')
%         % Vector de ecuaciones din micas
         Eq_Dyn = -Dyn_eq_VP;
%         %Creaci n de la matriz K y el vector Tau simb lico
%         load('K.mat')

K=sym(zeros(7,84));

for i=1:length(phi)
   i,tic
    phi_n = phi;
    phi_n(i)=[];
    Eq_Dyn_evaln=subs(Eq_Dyn,phi_n,sym(zeros(size(phi_n))));
    %Eq_Dyn_evaln = simplify(Eq_Dyn_evaln);
    fprintf('Calculando Kn\n');
    K_n=subs(Eq_Dyn_evaln,phi(i),1);
    K_n=subs(K_n,torques,zeros(size(torques)));
    %K_n=jacobian(Eq_Dyn_evaln,phi(i));
    fprintf('Simplificando Kn\n');
    K_n=simplify(K_n);
    K(:,i)=K_n;
    save ('K','K'),toc
end
```

```
45
46 disp('Resultados gurdados')
47 % K_sym=simplify(K);
48
49  disp('Termino independiente del problema de estimaci n de par metros) es:\n')
50  tau = -subs(Eq_Dyn,phi,zeros(size(phi)));
51  tau=simplify(tau);
52  tau
53
54  save ('tau_Sym','tau')
55
56 syms t   [1 N_gdl]      real          % Coordenadas generalizadas de posici n
57 syms dt  [1 N_gdl]      real          % Velocidades Generalizadas
58 syms ddt [1 N_gdl]      real          % Aceleraciones Generalizadas
59
60 %Creaci n de los vectores de las variables simb licas
61
62 %  fprintf('Vector de coordenadas generalizadas:')
63 q=t';
64 % fprintf('Vector de velocidades generalizadas:')
65 dq=dt';
66 % fprintf('Vector de aceleraciones generalizadas:')
67 ddq=ddt';
68 syms  t g f real
69
70 param = [g h0 h1 h2 h3 h4 h5 h6 h7 L6]';
71 torques = [M1 M2 M3 M4 M5 M6 M7]';
72
73 fprintf('\n\n Se crea la matlabfuction de K y Tau\n\n')
74
75 matlabFunction(K,'File','evalK','vars',{q,dq,ddq,param})
76 matlabFunction(tau,'File','evaltau','vars',{q,torques})
77
78
79 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80 %%%%%%%%%%%%% SECTION 2: SIMPLE PARAMETRIC EXCITATION %%%%%%%%%%%%%%%%
81 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
82
83 % Creaci n de los vectores de coordenadas, velocidades y aceleraciones de
84 % las trayectorias utilizando funciones sinoidales.
85 % fprintf('Parametrizaci n utilizando senos y cosenos \n')
86 [q_tray, dq_tray, ddq_tray]=get_q_dq_ddq_tray(t,f,N_gdl);
87
88 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
89 %%%%%%%%%%%%%% PARAMETRIC EXCITATION SFF%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
90 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
91
92 fprintf('Parametrizaci n utilizando las series finitas de Fourier(5 arm nicos):\n
       \n')
93 N_arm = 5;
94
95 % Creaci n del vector de parametros de trayecoria (organizado) por series finitas
       de
96 % Fourier y las creaci n de las variables simb licas de los mismos
97 [param_tray_SFF ,a0 ,a ,b]=get_param_tray_SFF(N_gdl,N_arm)
98
99 % Creaci n de los vectores de coordenadas, velocidades y aceleraciones de
100 % las trayectorias utilizando las series finitas de Fourier.
101 [q_tray_SFF,dq_tray_SFF,ddq_tray_SFF]=get_q_dq_ddq_tray_SFF(t,f,N_gdl,N_arm);
102
103 % Creaci n de los vectores del Jacobiano de las coordenadas, velocidades y
       aceleraciones de
104 % las trayectorias utilizando las series finitas de Fourier.
105 [Jq_tray_SFF,Jdq_tray_SFF,Jddq_tray_SFF]=get_Jacobian_q_dq_ddq_tray(t,f,N_gdl,N_arm
       );
106
107
108 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
109 %%%%%%%% CORRECCION EXPERIMENTOS %%%%%%%%%%%%%%%%%
110 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
111
112 [Wq_sym ,Wdq_sym ,Wddq_sym]=get_Wq_Wdq_Wddq(t,f,N_gdl,N_arm)
113
114 matlabFunction(phi_base_sym,'File','eval_phi_b_real','vars',{phi})
115
116 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%% SIMULACI N EXPERIMENTOS %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load('v_57_Fa')
load('phi_base_sym_Fa')
matlabFunction(phi_base_sym,'File','eval_phi_b_real','vars',{phi})
```

get_phi

```

function [phi]=get_phi(N_gdl,N_sol)
%Devuelve el vector de par metros din micos simbolicos a identificar tanto
% los par metros inerciales como de fricci n de cada s lido.
% La funci n se encarga de crear y organizar los par metros en base al
% n mero de s lidos(inerciales) y el n mero de grados de libertad(fricci n),
% el vector phi esta dividido en dos partes, la primera
% corresponde a los par emtros inerciales y el resto a los de fricci n.

        % Par metros inerciales, formado por la masa, primeros momentos y
        % el vector de inercia de cada s lido.
        syms m    [1 N_sol]                          real  % Masas de los solidos
        syms mx my mz    [1 N_sol]                   real  % Primer momento con
    respecto
        syms Ixx Ixy Ixz Iyy Iyz Izz [1 N_sol]    real  % Aceleraciones
    Generalizadas

        % Par metros de fricci n, formado por los coeficientes de fricci n,
        %[c] coeficiente de fricci n viscosa y [cc] coeficiente de fricci n
    cin tico.
        syms c  [1 N_gdl]       real       % Coeficeinte de fricci n viscoso
        syms cc [1 N_gdl]       real       % Coeficeinte de fircci n cin tico

        % Organizar el vector de par metros
        % din micos (m,mx,my,mz,Ixx,Ixy,Ixz,Iyy,Iyz,Izz) por cada s lido
        N_param_iner= 10; % 6 del tensor de inercia, 3 de los primeros momentos y 1
     la masa
        phi = sym([]);   % Creaci n del vector de par metros din micos vacio

        for j=1:N_sol
            phi(N_param_iner*j-9:N_param_iner*j,1)=[m(j),mx(j),my(j),mz(j),...
                Ixx(j),Ixy(j),Ixz(j),Iyy(j),Iyz(j),Izz(j)];
        end

        phi=[phi;c';cc'];      % Se agregan los par metros de fricci n al vector
end
```

QR_param_base

```

% Una vez definida W (matriz de observaci n se ordena y divide en columnas
    independites y dependietes)
% Se vuelve de rango m ximo W independiente
% Utilizando el M todo QR y el siguiente algoritmo

function  [phi_base_sym,v,rango_W,BETA_ent,Wb] = QR_param_base(W,phi)


    disp('Rango de W')
    rango_W = rank(W)
    %Rango = 56;

    tic
    [Q,R,E]=qr(W);
    toc
    [nfil_W,ncol_W]=size(W)
    p=ncol_W;
    WE=W*E;
    %Dividiendo la Matriz de observaci n en columans indep. y dependientes
    W1=WE(:,1:rango_W);
    W2=WE(:,rango_W+1:end);
    Q1=Q(:,1:rango_W);
    Q2=Q(:,rango_W+1:end);
    R1=R(1:rango_W,1:rango_W);
    R2=R(1:rango_W,rango_W+1:end);
    BETA = R1\R2;

```

```
    % Simplificando el vector de BETA para considerar los t rminos de mayor
    % influencia para los par metros dependientes
    BETA_ent=BETA;
    for i=1:size(BETA_ent,1)
        for j=1:size(BETA_ent,2)
            if abs(BETA_ent(i,j)) < 1e-4
                BETA_ent(i,j)=0;
            end
        end
    end


    %Reordenando el vector de parametros phi seg n t rminos independientes
    % Ordenados de independientes a dependientes y creaci n de v
    phi_ord=E'*phi;
    EpW=E'*[1:p]';
    phi1=phi_ord(1:rango_W,:); % Vector de par metros independienets
    phi2=phi_ord(rango_W+1:end,:); % Vector de par metros dependientes
    v=EpW(1:rango_W);
    phi_base_sym = phi1+BETA_ent*phi2;
    Wb=W1;
    % disp('b) Los par metros base estimados son: ')
   % phi_base_num = pinv(W'*W)*W'*tau;
%     save Sol_numumeric

end
```

C. Código de la Reducción

```
    %main_Numeric
    addpath(genpath('..\Simb lico'))
    close all
    clear all
    clc

    [N_gdl,N_sol] =get_N_gdl_N_sol();
    f=2*pi;
    n=100;
     t = 0:10/(n-1):10;

    %Definici n de los par metros geometricos y gravedad

    [phi]=get_phi(N_gdl,N_sol);
    [param]=get_parametros()

    % Par metros de trayectoria con un arm nico
    N_arm=1;
    param_tray=2*pi*rand((2*N_arm)*N_gdl,1);

    tic
    disp('creando W, para obtener los par metros base y el vector de permutaciones
    ')
    for j=1:length(t)
        q_tray = eval_qtray(t(j),f,param_tray);
        dq_tray = eval_dqtray(t(j),f,param_tray);
        ddq_tray = eval_ddqtray(t(j),f,param_tray);

        W_n=evalK(q_tray,dq_tray,ddq_tray,param);
        W((j-1)*N_gdl+1:j*N_gdl,:) = W_n;
    end,toc

    disp('comenzar Qr')
%   Una vez definida W (matriz de observaci n se ordena y divide en columnas
    independites y dependietes)
    [phi_base_sym,v,rango_W,BETA_ent,Wb] = QR_param_base(W,phi);
    rango_W
% cd ..\Simb lico
%   save('v_57_Fa','v')
%   save ('phi_base_sym_Fa','phi_base_sym')
% cd ..\Numeric
```

D. Código de las Trayectorias

Main_Trayectorias

```
%Main_Trayectorias
addpath(genpath('..\Simb lico'))
format compact
clear all
close all
clc
% profile on
format long
N_arm = 5;       % N mero de armonicos a utilizar
g     =9.80665;
f     =0.1;      % Frecuencia de medida
n     =100;      % N mero de puntos a utilizar
t     = 0:(1/f)/(n-1):10; % Vector de tiempo, de 0 a 10 segundos

[N_gdl,N_sol]=get_N_gdl_N_sol();

F_s=0.95;  % Factor de seguridad de la trayectoria
q_tray_max=F_s*[170;120;170;120;170;120;175]*pi/180; % radianes
q_tray_min=F_s*[-170;-120;-170;-120;-170;-120;-175]*pi/180; % radianes
dq_tray_max=F_s*[2267;2267;2667;2000;2167;3600;3600]*2*pi/60/160;    % rad/segundos
dq_tray_min=-F_s*[2267;2267;2667;2000;2167;3600;3600]*2*pi/60/160;    % rad/segundos
ubound=[q_tray_max, dq_tray_max];
lbound=[q_tray_min, dq_tray_min];

% Creaci n de las matrices y vectores de las restricciones (R y b)
[A,b] = get_A_b(t,f,N_gdl,N_arm,ubound,lbound);


 criterio='1';  % Criterio del n mero de condici n
% criterio='2';  % Criterio de la informaci n

rng('shuffle');
param_tray_SFF=2*pi*rand((2*N_arm+1)*N_gdl,1);
param_tray_0=param_tray_SFF;
load('v_57_Fa.mat')

if strcmp('1',criterio)
    crit=1;
    fprintf('Obteniendo el valor de la funci n objetivo empleando el Criterio del
    n mero de condici n\n');
        value = f_objetivo(t,v,f,param_tray_SFF,crit)
        options=optimoptions(@fmincon,'Display','iter','MaxIter',10000,'
    MaxFunctionEvaluations',inf,'PlotFcns','optimplotfval')

        [param_tray_opt,value]=fmincon(@(param_tray_SFF)f_objetivo(t,v,f,
    param_tray_SFF,crit),param_tray_0,A,b,[],[],[],[],[],options);
        value_str=num2str(value);
        % cd ..\Simb lico
%save( 'Ecuaciones_Dinamicas','Dyn_eq_VP')
% cd ..\Modelo_Robot
        ss=['param_opt_cond','_',value_str,'.mat'];
        command_str2=['save ',ss,' param_tray_opt;'];
        eval(command_str2);

elseif   strcmp('2',criterio)
    crit=2;
        fprintf('Obteniendo el valor de la funci n objetivo empleando el Criterio
    de de la informaci n')
        value = f_objetivo(t,v,f,param_tray_SFF,crit)
        options=optimoptions(@fmincon,'Display','iter','MaxIter',10000,'
    MaxFunctionEvaluations',inf,'PlotFcns','optimplotfval')

        [param_tray_opt,value]=fmincon(@(param_tray_SFF)f_objetivo(t,v,f,
    param_tray_SFF,crit),param_tray_0,A,b,[],[],[],[],[],options);
        value_str=num2str(value);
        j_str=num2str(1);% N mero del fichero a optimizar
        ss=['param_opt_inf','_',value_str,'.mat'];
        command_str2=['save ',ss,' param_tray_opt;'];
        eval(command_str2);
end

% save('Trayectorias\param_opt_cond_a4.mat','param_tray_opt')
```

```

return
```

get_A_b

```
%Creaci n de la matriz R, para
function [A,b] = get_A_b(t,f,N_gdl,N_arm,ubound,lbound)

% Para nuestra configuraci n tenermos limitaciones en posici n y velocidad,
% de la forma Ax<=b, inecuaciones que se deben cumplir.
% Considerando los limites inferiores y superiores tendriamos:
% lbound<R*param_tray<ubound
%R*param_tray<ubound    ----> A=R  ;  b = ubound
% -R*param_tray<lbound ----> A=-R ;  b=-lbound
% ubond
% q_tray_max=[170;120;170;120;170;120;175]*pi/180; % grados (  )
% dq_tray_max=[85;85;100;75;130;135;135]*60/360;    % grados/segundos
% lbond
% dq_tray_min=[85;85;100;75;130;135;135]*60/360;    % grados/segundos
% q_tray_min=[-170;-120;-170;-120;-170;-120;-175]*pi/180; % grados (  )
% ubound=[q_tray_max, dq_tray_max];
% lbound=[q_tray_min, dq_tray_min];
N_ubound = size(ubound,2);
N_lbound = size(lbound,2);
N_f=N_ubound + N_lbound;

 A = zeros(length(t)*N_gdl*N_f,(2*N_arm+1)*N_gdl);% n mero de columnas (N_gdl*2),
    2 = q y dq
 b = zeros(length(t)*N_gdl*N_f,1);

 for i= 1:length(t)

    Aq_i  = eval_Jq_tray(t(i),f);
    Adq_i = eval_Jdq_tray(t(i),f);

    A(4*N_gdl*(i-1)+1:4*N_gdl*i,:)= [Aq_i;-Aq_i;Adq_i;-Adq_i];
    b(4*N_gdl*(i-1)+1:4*N_gdl*i,1)= [ubound(:,1);-lbound(:,1);ubound(:,2);-lbound
    (:,2)];

 end
end
```

f_objetivo

```
function value=f_objetivo(t,v,f,param_tray_SFF,crit)

[N_gdl,N_sol]=get_N_gdl_N_sol();

W_SFF= zeros(N_gdl*length(t),84);
[param_num]=get_parametros();

for j=1:length(t)
    q_tray = eval_qtray_SFF(t(j),f,param_tray_SFF);
    dq_tray = eval_dqtray_SFF(t(j),f,param_tray_SFF);
    ddq_tray = eval_ddqtray_SFF(t(j),f,param_tray_SFF);

    W_n=evalK(q_tray,dq_tray,ddq_tray,param_num);
    W_SFF((j-1)*N_gdl+1:j*N_gdl,:) = W_n;
end

Wb=W_SFF(:,v);
    if crit==1
        value= cond(Wb);
    elseif crit==2
        value = -log(det(10^-5*Wb'*Wb));
    end
end
```

Main_Trayectorias_graficas

```
% Trayectorias_graficas
format compact
clear all
% close all
% clc
format long
N_arm = 5;       % N mero de armonicos a utilizar
g     =9.80665;
f     =0.1;      % Frecuencia de medida
n     =100;      % N mero de puntos a utilizar
t     = 0:(1/f)/(n-1):10; % Vector de tiempo, de 0 a 10 segundos

[N_gdl,N_sol]=get_N_gdl_N_sol();

F_s=0.95;  % Factor de seguridad de la trayectoria
q_tray_max=F_s*[170;120;170;120;170;120;175]*pi/180; % grados (  )
q_tray_min=F_s*[-170;-120;-170;-120;-170;-120;-175]*pi/180; % grados (  )
dq_tray_max=F_s*[2267;2267;2667;2000;2167;3600;3600]*2*pi/60/160;    % grados/
    segundos
dq_tray_min=-F_s*[2267;2267;2667;2000;2167;3600;3600]*2*pi/60/160;    % grados/
    segundos
ubound=[q_tray_max, dq_tray_max];
lbound=[q_tray_min, dq_tray_min];

% Creaci n de las matrices y vectores de las restricciones (R y b)
[A,b] = get_A_b(t,f,N_gdl,N_arm,ubound,lbound);

load('Trayectorias\param_opt_cond_49.2447.mat')

q_opt=A*param_tray_opt;

for i=1:7
   Q(:,i)=q_opt(i:7*4:end);
   DQ(:,i)=q_opt(i+7*2:7*4:end);
end

figure
for i=1:N_gdl
subplot(2,4,i)
plot(t,Q(:,i).*180/pi),hold on;
plot([0,1/f],[-q_tray_max(i),-q_tray_max(i)]*180/pi,'r','LineWidth',2)
plot([0,1/f],[q_tray_max(i),q_tray_max(i)]*180/pi,'r','LineWidth',2)
xlabel('Tiempo (seg)')
ylabel('Posici n (  )')
title(['Posici n del eje ',num2str(i)])
end
sgtitle('Posici n de los ejes del robot')

figure
for i=1:N_gdl
subplot(2,4,i)
plot(t,DQ(:,i).*60*160/2/pi),hold on;
plot([0,1/f],[-dq_tray_max(i),-dq_tray_max(i)].*60*160/2/pi,'r','LineWidth',2)
plot([0,1/f],[dq_tray_max(i),dq_tray_max(i)].*60*160/2/pi,'r','LineWidth',2)
xlabel('Tiempo (seg)')
ylabel('Velocidad (rpm)')
title(['Velocidad del eje ',num2str(i)])
end
sgtitle('Velocidad de los ejes del robot')
```

get_parametros

```

function [param_num]=get_parametros()

% Par metros Geometrios%*************
param_num(1,1) = 9.80665;
param_num(2,1) = 0.1575 ;
param_num(3,1) = 0.2025 ;
param_num(4,1) = 0.2045 ;
param_num(5,1) = 0.2155;
param_num(6,1) = 0.1845;
param_num(7,1) = 0.2155 ;
param_num(8,1) = 0.0809 ;
```

```
param_num(9,1) = 0.0451;
param_num(10,1)= 0.0607;
end
```

E. Código del Experimento

```
addpath(genpath('..\Simb lico')) % Funciones simb licas
addpath(genpath('..\Trayectorias')) % Trayectorias creadas

%%Clean environment
close all
clear all
clear
clc
format compact

%% Simulaci n de los experimentos

n=100;              % N mero de puntos considerados, a lo largo de la trayectoria
t = 0:10/(n-1):10;  % Intervalo de tiempo en la que se realizar  el experimento
f=0.1;              % Frecuencia de muestreo, 10 muestras por segundo
g=9.81;             % Par metro de la gravedad
N_arm=5;            % N mero de arm nicos utilizados para la parametrizaci n de
    la trayectoria

% El numero de gdl y vector de parametros inerciales simbolico
[N_gdl,N_sol]=get_N_gdl_N_sol();
% Creador del vector simb lico de los par metros inerciales
[phi]=get_phi(N_gdl,N_sol);


%% Utilizo los valores de inercia y friccion de mi modelo en 3_Mec
%[mass,center_of_mass,inertia_tensor] = get_solid_prop

[mass,center_of_mass,inertia_tensor] = get_Propiedades_Inercia_Sw3

friccion_c = [0.3;0.4;0.4;0.3;0.2;0.1;0.05]
friccion_cc = [0.2;0.3;0.3;0.1;0.1;0.05;0.1]
%Orden de los par metros inerciales por solido:
% [mi;mxi;myi;mzi;Ixxi;Ixyi;Ixzi;Iyyi;Iyzi;Izzi]

% Vector de los par metros inerciales y de friccion reales

phi_real=[];
for i=1:N_gdl
    phi_real=[phi_real;mass{i+1};...
        center_of_mass{i+1}(1,1);center_of_mass{i+1}(2,1);center_of_mass{i+1}(3,1)
    ;...
        inertia_tensor{i+1}(1,1);inertia_tensor{i+1}(1,2);inertia_tensor{i+1}(1,3)
    ;...
        inertia_tensor{i+1}(2,2);inertia_tensor{i+1}(2,3);inertia_tensor{i+1}(3,3)
    ];
end

%Vector de par metros inerciales y de friccion de mi modelo 3d_mec
phi_real=[phi_real;friccion_c;friccion_cc];


%% Utilizo los parametros optimizados para una de mis trayectorias

% load('Trayectorias\param_opt_cond_49.2447.mat')
load('Trayectorias\param_opt_cond_57.8951.mat')

%Obtengo el vector de par metros  ptimos  de la Serie Finita de Fourier
param_tray_SFF=param_tray_opt;
[param_num]=get_parametros()

 [W,Q_real,DQ_real,DDQ_real] = get_W(t,f,n,param_tray_SFF,param_num);
% save('W','W')
% save('Q_DQ_DDQ.mat','Q_real','DQ_real','DDQ_real')

%load('W.mat')
%load('Q_DQ_DDQ.mat')
% save('W_real','W')
% load('W_real.mat')
T_real = W*phi_real;
```

```
67
68
69 %% Desviaci n tipica de cada eje de forma individual tanto en posici n como
       torque
70
71 % 1) la sigma que tendr as que utilizar deber as ser algo realista. Por
72 % ejemplo, si los pares que vas a tener son del orden de Tmax [N*m],
73 % entonces deber as poner una sigma entorno al 1%-5% que podr a ser algo
74 % realista. Tras "crear" las se ales medidas de esta forma, podr as a
75 % posteriori pensar que realmente son medidas y estimar la desviaci n
76 % t pica del error de medici n, a ver si te sale lo que t  mismo hab as
77 % puesto para crear el ruido de la funci n.
78 %
79 % 2) podr as ponerle a todos los ejes el mismo porcentaje, pero siempre el
80 % % del valor m ximo de para de cada eje. -> Tmax_i -> sigma_i =
81 % 0.05*Tmax_i (si lo hicieras para un 5%)
82
83 % Encontrar la desviaci n tipica, requerida para agregar ruido a los
84 % valores reales, de manera que se han mas parecido a los medidos realmente
85 % (con ruido)
86 % rng('default');
87 %Obteniendo el valor maximo de cada eje y utilizando un sigma de 5% (realista)
88
89 T_abs= abs(T_real);
90 for i=1:N_gdl
91 Tmax_i(i,1) = max(T_abs(i:N_gdl:end,1));
92 sigma_Ti(i,1)= 0.05*Tmax_i(i);
93 end
94
95 %Sigma de cada eje en n instantes
96 for j=1:n
97 sigma_T((j-1)*N_gdl+1:j*N_gdl,1)=sigma_Ti(:,1);
98 t(j)=t(j);
99 end
100
101 T_m = T_real + sigma_T.*randn(size(T_real));
102
103 % figure(nfig);nfig=nfig+1;
104   figure('name','1.-Medicion de Torques con Ruido(individual)'), hold on
105
106 for i=1:N_gdl
107 subplot(2,4,i)
108 plot(t,T_real(i:N_gdl:end,1)),hold on
109 plot(t,T_m(i:N_gdl:end,1))
110 xlabel('Tiempo (seg)')
111 ylabel('Torque (Nm)')
112 title(['Torque del motor ',num2str(i)])
113 end
114
115 % fig=gcf;
116 % saveas(fig,'imagenes\1.-Medicion de Torques con Ruido(individual).jpg', 'jpg')
117
118 Q_abs= abs(Q_real);
119 for i=1:N_gdl
120 Qmax_i(i,1) = max(Q_abs(i:N_gdl:end,1));
121 sigma_Qi(i,1)= 0.01*Qmax_i(i);
122 end
123 %Sigma de cada eje en n instantes
124 for j=1:n
125 sigma_Q((j-1)*N_gdl+1:j*N_gdl,1)=sigma_Qi(:,1);
126 end
127
128 Q_m = Q_real + sigma_Q.*randn(size(Q_real));
129
130 % figure(nfig);nfig=nfig+1;
131 figure('name','2.-Medici n de la posicione de los ejes con Ruido (Individual)'),
       hold on
132 % % title('Torque generado por cada uno de los motores')
133 for i=1:N_gdl
134 subplot(2,4,i)
135 plot(t,Q_real(i:N_gdl:end,1).*180/pi),hold on
136 plot(t,Q_m(i:N_gdl:end,1).*180/pi)
137 xlabel('Tiempo (seg)')
138 ylabel('Posici n (  )')
139 title(['Posici n del eje  ',num2str(i)])
140 end
```

```
141
142 % fig=gcf;
143 % saveas(fig,'imagenes\2.-Medici n de la posicione de los ejes con Ruido (
        Individual).jpg', 'jpg')
144 %% Buscar una serie finita de Fourier que se aproxime a la trayectoria medida (con
        ruido)
145
146 % x = Wx(t)   : plantear una parametrizaci n de x que sea lineal en   .
147 %  _est  = pinv(Wx)xm : estimar los par ametros de trayectoria.
148 % x_est = Wx _est : estimar los valores de la variable x.
149 % dx_est = Wdx _est : estimar los valores de la variable  x .
150 % ddx_est = Wddx _est : estimar los valores de la variable  x .
151 % y = W(x_est,dx_est ,ddx_est)   : plantear la estimaci on de los par ametros   .
152 %      = pinv(W)ym : estimar los par ametros   .
153
154 %   Crear la matriz Wx,Wdx y Wddx
155     for i=1:n   % For para los n instantes del experiemnto
156
157         for j=1:N_gdl       %for para cada trayectoria de los ejes
158             Wq_i(j,:)   = eval_Wq(t(i),f);
159             Wdq_i(j,:)  = eval_Wdq(t(i),f);
160             Wddq_i(j,:) = eval_Wddq(t(i),f);
161         end
162     % Ensamblaje de la matriz Wx,Wdx,Wddx
163     Wq((i-1)*N_gdl+1:i*N_gdl,:) = Wq_i(1:N_gdl,:);
164     Wdq((i-1)*N_gdl+1:i*N_gdl,:) = Wdq_i(1:N_gdl,:);
165     Wddq((i-1)*N_gdl+1:i*N_gdl,:) = Wddq_i(1:N_gdl,:);
166     end
167
168 %Obtenci n de los par metros estimados de la Serie Finita de Fourier que se
169 %aproxima a las mediciones
170 param_SFF_est=[];
171 for i=1:N_gdl
172 param_SFF_est((i-1)*(N_arm*2+1)+1:i*(N_arm*2+1),:)=pinv(Wq(i:N_gdl:end,:))*Q_m(i:
        N_gdl:end,1);
173 end
174
175 %Obtenci n de las posiciones, velocidades y aceleraciones estimadas con las
176 %respectivas derivadas de la funcion analitica de la SFF aproximada. para cada eje
177 for i=1:N_gdl
178 % param_SFF_est((i-1)*11+1:i*11,:)=pinv(Wx(1:7:end,1))*Q_m_i(1:7:end,1);
179 Q_est(i:N_gdl:N_gdl*n,1) = Wq(i:N_gdl:end,:)*param_SFF_est((i-1)*(N_arm*2+1)+1:i*(
        N_arm*2+1),:);
180 DQ_est(i:N_gdl:N_gdl*n,1) = Wdq(i:N_gdl:end,:)*param_SFF_est((i-1)*(N_arm*2+1)+1:i
        *(N_arm*2+1),:);
181 DDQ_est(i:N_gdl:N_gdl*n,1) = Wddq(i:N_gdl:end,:)*param_SFF_est((i-1)*(N_arm*2+1)+1:
        i*(N_arm*2+1),:);
182 end
183
184 % Graficar las posiciones y velocidades aproximadas de la medicion
185 % figure(nfig);nfig=nfig+1;
186 figure('name','3.-posicion de los ejes estimados'), hold on
187
188 for i=1:N_gdl
189 subplot(2,4,i)
190 plot(t,Q_real(i:N_gdl:end,1).*180/pi,'r','LineWidth',1),hold on
191 plot(t,Q_m(i:N_gdl:end,1).*180/pi,'g','LineWidth',1),hold on
192 plot(t,Q_est(i:N_gdl:end,1).*180/pi,'b','LineWidth',1)
193 xlabel('Tiempo (seg)')
194 ylabel('Posici n (  )')
195 title(['Posici n del eje  ',num2str(i)])
196 legend('Real','Medida','Est')
197 end
198
199 % fig=gcf;
200 % saveas(fig,'imagenes\3.-Posici n de los ejes estimados.jpg', 'jpg')
201
202 % figure(nfig);nfig=nfig+1;
203 figure('name','4.-velocidades de los ejes estimadas'), hold on
204 for i=1:N_gdl
205 subplot(2,4,i)
206 plot(t,DQ_real(i:N_gdl:end,1).*160*60/2/pi,'r','LineWidth',1),hold on
207 plot(t,DQ_est(i:N_gdl:end,1).*160*60/2/pi,'b','LineWidth',1)
208 xlabel('Tiempo (seg)')
209 ylabel('Velocidad (rpm)')
210 title(['Velocidad del eje ',num2str(i)])
```

```
legend('Real','Est')
end

% fig=gcf;
% saveas(fig,'imagenes\4.-velocidades de los ejes estimadas.jpg', 'jpg')


%% Proceso de estimaci n de los par metros

%Crear W con los valores estimados
 % En caso de utilizar las curvas estimadas con SFF de la trayectoria con ruido
[param_num]=get_parametros()
 [W_F,Q_estF,DQ_estF,DDQ_estF] = get_W(t,f,n,param_SFF_est,param_num);
% save('W_F','W_F')
% save('Q_DQ_DDQ_est.mat','Q_estF','DQ_estF','DDQ_estF')

% Una vez definida W (matriz de observaci n se ordena y divide en columnas
    independites y dependietes)
    load('v_57_Fa')           % Vector ordenado de columanas linealmente
    independiente
    load('phi_base_sym_Fa')   % Estructura simbolica de los par metros base
     Wb=W_F(:,v);
%     matlabFunction(phi_base_sym,'File','eval_phi_b_real','vars',{phi})
%     matlabFunction(Wb,'File','eval_Wb','vars',{q_tray_SFF,dq_tray_SFF,
    ddq_tray_SFF,v})

% Construcci n de la matriz de obervaciones de los parametros base
%     Wb=W_F(:,v);
    rango_W=rank(Wb);

% Mi sistema es lineal en los par metros base, por lo que los creo a partir de mi
    W
    phib_real=eval_phi_b_real(phi_real);

 % Estimacion de los par metros base del sistema con los Torques medidos
    phib_est2=pinv(Wb)*T_m;
    n_phib = 1:1:rango_W;
    T_est2= Wb*phib_est2;
%     save('T_est2','T_est2')
%figure('name','5.-Histograma del Residuo de los Torques'), hold on
%histfit(epsilon);

epsilon2 = T_m-T_est2;
epsilon2_i=zeros(100,7);
for i=1:7
 epsilon2_i(:,i)=epsilon2(i:7:700,1);
end

% figure(nfig), nfig=nfig+1;
figure('name','5.-Histograma del Residuo de los Torques'), hold on
for i=1:N_gdl
subplot(2,4,i)
histfit(epsilon2_i(:,i));
% xlabel('Tiempo (seg)')
ylabel('Torque Nm ')
title(['Histograma del residuo del Torque del eje ',num2str(i)])
end

% fig=gcf;
% saveas(fig,'imagenes\5.-Histograma del Residuo de los Torques.jpg', 'jpg')

% figure(nfig), nfig=nfig+1;
figure('name','6.-phi_Base Estimado Vrs phi_Base Real'), hold on
plot(n_phib,phib_est2,'--b','LineWidth',1.5), hold on,
plot(n_phib,phib_real,'LineWidth',1)
xlabel('Par metros Base \phi_{base}')
legend('Estimacion','Real')
title('\phi_{Base Estimados} Vrs \phi_{Base Real}')

% fig=gcf;
% saveas(fig,'imagenes\6.-phi_Base Estimado Vrs phi_Base Real.jpg', 'jpg')

 %%  Desviacion de datos, grafico de resultados y margen de error
% % % %Matriz de varianzas y covarianzas
% % % %sigma=mean(sigma_Ti)%% Debo obtener un escalar para mi sigma???
% % % SIGMA = diag(sigma_Ti.^2);
```

```
% % % SIGMA_K =kron(eye(n),SIGMA);
% % % VAR_phib_est = inv(Wb'*inv(SIGMA_K)*Wb);
% % %
% % % % Desviaci n tipica de las varianzas(diagonal)
% % % desv_phib_est=sqrt(diag(VAR_phib_est));
% % %
% % % % Defino mi intervalo de error para identificar si mi variables estimado
% % % % esta dentro de dicho intervalo
% % % error_pos=zeros(rango_W,1);
% % % error_neg=zeros(rango_W,1);
% % % for i=1:rango_W
% % % error_pos(i)= +3*desv_phib_est(i);
% % % error_neg(i)= -3*desv_phib_est(i);
% % % end
% % % n_phi_b = 1:1:rango_W;
% % %
% % % figure('name','Parametros base estimados e intervalo de error'), hold on
% % % errorbar(n_phib,phib_est,error_pos,error_neg,'o'),hold on
% % % plot(n_phib,phib_real,'or')
% % % xlabel('Par metros Base \phi_{base}')
% % % ylabel('\phi_{base_{est}}+ 3*\sigma_{est}')
% % % title('\phi_{Base Estimados} Vrs \phi_{Base Real}')
% % % legend('Estimaci n','Real')
%%  Desviacion de datos, grafico de resultados y margen de error calculando SIGMA
%Matriz de varianzas y covarianzas
%sigma=mean(sigma_Ti)%% Debo obtener un escalar para mi sigma???

Tol=10^-8;
phib_est0=0;
[VAR_phib_est,desv_phib_est,phib_est]=get_matriz_error(Tol,n,phib_est0,Wb,T_m);
T_est= Wb*phib_est;

% Desviaci n tipica de las varianzas(diagonal)
% desv_phi_b_est=sqrt(diag(VAR_phi_b_est));

% figure('name','7.-Histograma del Residuo de los Torques'), hold on
% histfit(epsilon);
epsilon = T_m-T_est;
epsilon_i=zeros(100,7);
for i=1:7
 epsilon_i(:,i)=epsilon(i:7:700,1);
end

% figure(nfig), nfig=nfig+1;
figure('name','7.-Histograma del Residuo de los Torques'), hold on
for i=1:N_gdl
subplot(2,4,i)
histfit(epsilon_i(:,i));
% xlabel('Tiempo (seg)')
ylabel('Torque Nm ')
title(['Histograma del residuo del Torque del eje ',num2str(i)])
end
% fig=gcf;
% saveas(fig,'imagenes\7.-Histograma del Residuo de los Torques.jpg', 'jpg')


for i=1:rango_W
error_pos(i)= +3*desv_phib_est(i);
error_neg(i)= -3*desv_phib_est(i);
end



figure('name','8.-Parametros base estimados e intervalo de error'), hold on
errorbar(n_phib,phib_est,error_pos,error_neg,'o'),hold on
plot(n_phib,phib_real,'or')
xlabel('Par metros Base \phi_{base}')
ylabel('\phi_{base_{est}}- 3*\sigma_{est}')
legend('Estimacion e Intervalo','Real')

%     figure(nfig), nfig=nfig+1;
    figure('name','9.-phi_Base Estimado Vrs phi_Base Real'), hold on
    plot(n_phib,phib_est,'--b','LineWidth',1.5), hold on
    plot(n_phib,phib_real,'LineWidth',1)
    legend('Estimacion','Real')
    xlabel('Par metros Base \phi_{base}')
```

```
    title('\phi_{Base Estimados} Vrs \phi_{Base Real}')

%% %%%%%%%%%%%%%%%Validacion del modelo%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load('Trayectorias\param_opt_cond_53.9105.mat')
%asigno el vector de parmetros optimos de Serie Finita de Fourier
param_tray_SFF=param_tray_opt;
[param_num]=get_parametros()

[Wv,Qv_real,DQv_real,DDQv_real] = get_W(t,f,n,param_tray_SFF,param_num);
% save('Wv','Wv')
%save('Qv_DQv_DDQv_real.mat','Qv_real','DQv_real','DDQv_real')
% load('Wv.mat')
% load('Qv_DQv_DDQv_real.mat')

Tv_real = Wv*phi_real;

%% Desviaci n tipica de cada eje de forma individual tanto en posici n como
    torque
% Encontrar la desviaci n tipica, requerida para agregar ruido a los
% valores reales, de manera que se han mas parecido a los medidos realmente
% (con ruido)

%Obteniendo el valor maximo de cada eje y utilizando un sigma de 5% (realista)
Tv_abs= abs(Tv_real);
for i=1:N_gdl
Tvmax_i(i,1) = max(Tv_abs(i:N_gdl:end,1));
sigma_Tvi(i,1)= 0.05*Tvmax_i(i);
end
%Sigma de cada eje en n instantes
for j=1:n
sigma_Tv((j-1)*N_gdl+1:j*N_gdl,1)=sigma_Tvi(:,1);
end

Tv_m = Tv_real + sigma_Tv.*randn(size(Tv_real));
figure('name','10.-Medicion de Torques con Ruido(individual)'), hold on
% % title('Torque generado por cada uno de los motores')
for i=1:N_gdl
subplot(2,4,i)
plot(t,Tv_real(i:N_gdl:end,1)),hold on
plot(t,Tv_m(i:N_gdl:end,1))
xlabel('Tiempo (seg)')
ylabel('Torque (Nm)')
title(['Torque del motor ',num2str(i)])
end


Qv_abs= abs(Qv_real);
for i=1:N_gdl
Qvmax_i(i,1) = max(Qv_abs(i:N_gdl:end,1));
sigma_Qvi(i,1)= 0.01*Qvmax_i(i);
end
%Sigma de cada eje en n instantes
for j=1:n
sigma_Qv((j-1)*N_gdl+1:j*N_gdl,1)=sigma_Qvi(:,1);
end

Qv_m = Qv_real + sigma_Qv.*randn(size(Qv_real));

figure('name','11.-Medici n de la posici n de los ejes con Ruido (Individual)'),
    hold on
% % title('Torque generado por cada uno de los motores')
for i=1:N_gdl
subplot(2,4,i)
plot(t,Qv_real(i:N_gdl:end,1).*180/pi),hold on
plot(t,Qv_m(i:N_gdl:end,1).*180/pi)
xlabel('Tiempo (seg)')
ylabel('Posici n (  )')
title(['Posici n del eje  ',num2str(i)])
end

%% Buscar una serie finita de Fourier que se aproxime a la trayectoria medida (con
    ruido)

```

```
%   Crear la matriz Wx,Wdx y Wddx
    for i=1:n   % For para los n instantes del experiemnto

        for j=1:N_gdl       %for para cada trayectoria de los ejes
            Wq_i(j,:)   = eval_Wq(t(i),f);
            Wdq_i(j,:)  = eval_Wdq(t(i),f);
            Wddq_i(j,:) = eval_Wddq(t(i),f);
        end
    % Ensamblaje de la matriz Wx,Wdx,Wddx
    Wvq((i-1)*N_gdl+1:i*N_gdl,:) = Wq_i(1:N_gdl,:);
    Wvdq((i-1)*N_gdl+1:i*N_gdl,:) = Wdq_i(1:N_gdl,:);
    Wvddq((i-1)*N_gdl+1:i*N_gdl,:) = Wddq_i(1:N_gdl,:);
    end

%Obtenci n de los par metros estimados de la Serie Finita de Fourier que se
%aproxima a las mediciones
paramv_SFF_est=[];
for i=1:N_gdl
paramv_SFF_est((i-1)*(N_arm*2+1)+1:i*(N_arm*2+1),:)=pinv(Wvq(i:N_gdl:end,:))*Qv_m(i
    :N_gdl:end,1);
end

%Obtenci n de las posiciones, velocidades y aceleraciones estimadas con las
%respectivas derivadas de la funcion analitica de la SFF aproximada. para cada eje
for i=1:N_gdl
% param_SFF_est((i-1)*11+1:i*11,:)=pinv(Wx(1:7:end,1))*Q_m_i(1:7:end,1);
Qv_est(i:N_gdl:N_gdl*n,1) = Wvq(i:N_gdl:end,:)*paramv_SFF_est((i-1)*(N_arm*2+1)+1:i
    *(N_arm*2+1),:);
DQv_est(i:N_gdl:N_gdl*n,1) = Wvdq(i:N_gdl:end,:)*paramv_SFF_est((i-1)*(N_arm*2+1)
    +1:i*(N_arm*2+1),:);
DDQv_est(i:N_gdl:N_gdl*n,1) = Wvddq(i:N_gdl:end,:)*paramv_SFF_est((i-1)*(N_arm*2+1)
    +1:i*(N_arm*2+1),:);
end

% Graficar las posiciones y velocidades aproximadas de la medicion

figure('name','12.- Posici n de los ejes estimados'), hold on

for i=1:N_gdl
subplot(2,4,i)
plot(t,Qv_real(i:N_gdl:end,1).*180/pi,'r','LineWidth',1),hold on
plot(t,Qv_m(i:N_gdl:end,1).*180/pi,'g','LineWidth',1),hold on
plot(t,Qv_est(i:N_gdl:end,1).*180/pi,'b','LineWidth',1)
xlabel('Tiempo (seg)')
ylabel('Posici n (  )')
title(['Posicion del eje  ',num2str(i)])
legend('Real','Medida','Est')
end


figure('name','13.- Velocidades de los ejes estimadas'), hold on
for i=1:N_gdl
subplot(2,4,i)
plot(t,DQv_real(i:N_gdl:end,1).*160*60/2/pi,'r','LineWidth',1),hold on
plot(t,DQv_est(i:N_gdl:end,1).*160*60/2/pi,'b','LineWidth',1)
xlabel('Tiempo (seg)')
ylabel('Velocidad (rpm)')
title(['Velocidad del eje ',num2str(i)])
legend('Real','Est')
end

%% Proceso de estimaci n de los par metros

%Crear W con los valores estimados
%En caso de utilizar las curvas estimadas con SFF de la trayectoria con ruido
%Wv_F= zeros(N_gdl*length(t),length(phi));
[param_num]=get_parametros()
[Wv_F,Qv_est,DQv_est,DDQv_est] = get_W(t,f,n,paramv_SFF_est,param_num);
%save('Wv_F.mat','Wv_F')
%save('Qv_DQv_DDQv_est.mat','Qv_est','DQv_est','DDQv_est')
% load('Wv_F.mat')
% load('Qv_DQv_DDQv_est.mat')
% save('Wv_F51','Wv_F')
% load('Wv_F51.mat')

load('v_57_Fa')
```

```
load('phi_base_sym_Fa')
rango_W=rank(Wb);
Wvb=Wv_F(:,v);

% Mi sistema es lineal en los par metros base, por lo que los creo a partir
% de mi Wv
    phib_realv=eval_phi_b_real(phi_real);
% Estimacion de los par metros base del sistema con los Torques medidos
    phib_estv2=pinv(Wvb)*Tv_m;
    n_phib = 1:1:rango_W;
    Tv_est2= Wvb*phib_estv2;
    save('Tv_est2.mat','Tv_est2')
% Desviaci n tipica de las varianzas(diagonal)
% desv_phi_b_est=sqrt(diag(VAR_phi_b_est));

% epsilonv = Tv_m-Tv_est;
% figure('name','13.- Histograma del Residuo de los Torques K Validacion'), hold on
% histfit(epsilonv);

epsilonv2 = Tv_m-Tv_est2;
epsilonv2_i=zeros(100,7);
for i=1:7
 epsilonv2_i(:,i)=epsilonv2(i:7:700,1);
end

figure('name','14.- Histograma del Residuo de los Torques,Validacion'), hold on
for i=1:N_gdl
subplot(2,4,i)
histfit(epsilonv2_i(:,i));
% xlabel('Tiempo (seg)')
ylabel('Torque Nm ')
title(['Histograma del residuo del Torque del eje ',num2str(i)])
end

figure('name','15.- phi_Base Estimado Vrs phi_Base Real'), hold on
plot(n_phib,phib_estv2,'--b','LineWidth',1.5), hold on
plot(n_phib,phib_realv,'LineWidth',1)
legend('Estimacion','Real')
xlabel('Par metros Base \phi_{base}')
title('\phi_{Base Estimados} Vrs \phi_{Base Real}')

%%  Desviacion de datos, grafico de resultados y margen de error calculando SIGMA
%Matriz de varianzas y covarianzas
Tol=10^-8;
phib_est0=0;
[VAR_phib_estv,desv_phib_estv,phib_estv]=get_matriz_error(Tol,n,phib_est0,Wvb,Tv_m)
    ;
Tv_est= Wvb*phib_estv;
%save('Tv_est.mat','Tv_est')
% Desviaci n tipica de las varianzas(diagonal)
% desv_phi_b_est=sqrt(diag(VAR_phi_b_est));
% epsilonv = Tv_m-Tv_est;
%
%   figure('name','15.- Histograma del Residuo de los Torques K'), hold on
%   histfit(epsilonv);

epsilonv = Tv_m-Tv_est;
epsilonv_i=zeros(100,7);
for i=1:7
 epsilonv_i(:,i)=epsilonv(i:7:700,1);
end

figure('name','16.- Histograma del Residuo de los Torques,Validacion'), hold on
for i=1:N_gdl
subplot(2,4,i)
histfit(epsilonv_i(:,i));
% xlabel('Tiempo (seg)')
ylabel('Torque Nm ')
title(['Histograma del residuo del Torque del eje ',num2str(i)])
end

%   epsiloncc = phib_est-phib_estv;
%   figure('name','16.- Histograma del Residuo con validaci n'), hold on
%   histfit(epsiloncc);

figure('name','17.- Parametros base estimados Vrs P. Reales validacion'), hold on
```

```
plot(n_phib,phib_estv,'--b','LineWidth',1.5), hold on
plot(n_phib,phib_realv,'LineWidth',1)
xlabel('Par metros Base \phi_{base}')
legend('Estimacion','Real')

for i=1:rango_W
error_posk(i)= +3*desv_phib_estv(i);
error_negk(i)= -3*desv_phib_estv(i);
end

figure('name','18.- Par metros base estimados e intervalo de error'), hold on
errorbar(n_phib,phib_estv,error_posk,error_negk,'og'),hold on
plot(n_phib,phib_realv,'sr')
xlabel('Par metros Base \phi_{base}')
ylabel('\phi_{base_{est}}- 3*\sigma_{est}')
legend('Estimacion','Real')


figure('name','19.- Comparaci n de los par metros estimados en dos trayectorias')
    , hold on
    plot(n_phib,phib_est,'--b','LineWidth',1), hold on
    plot(n_phib,phib_estv,'LineWidth',1)
    legend('Trayectoria 1',' Trayectoria 2')
    xlabel('Par metros Base \phi_{base}')
%% Error der validacion

Epsilon_validacion=Tv_est-Wvb*phib_est;
Epsilon_validacionM=zeros(100,7);
for i=1:7
 Epsilon_validacionM(:,i)=Epsilon_validacion(i:7:700,1);
end

figure('name','20.- Histograma del residuo de la Validaci n Cruzada'), hold on
for i=1:N_gdl
subplot(2,4,i)
histfit(Epsilon_validacionM(:,i));
% xlabel('Tiempo (seg)')
ylabel('Torque Nm ')
title(['Histograma del residuo del Torque del eje ',num2str(i)])
end


  %% Resultados Finales

figure('name','21.-Torque de los ejes estimados'), hold on
for i=1:N_gdl
subplot(2,4,i)
plot(t,T_est(i:N_gdl:end,1),'g','LineWidth',1),hold on
plot(t,T_m(i:N_gdl:end,1),'.r','LineWidth',1),hold on
% plot(t,Q_est(i:N_gdl:end,1).*180/pi,'b','LineWidth',1)
xlabel('Tiempo (seg)')
ylabel('Torque Nm')
title(['Torque del eje  ',num2str(i)])
legend('Estimado','Medido')
end


IC_min_T= T_est-3*sigma_T;
IC_max_T= T_est+3*sigma_T;

figure('name','22.- Torque de los ejes estimados e intervalos de confianza'), hold
    on
for i=1:N_gdl
subplot(2,4,i)
plot(t,T_est(i:N_gdl:end,1),'g','LineWidth',1),hold on
plot(t,T_m(i:N_gdl:end,1),'.r','LineWidth',1),hold on
plot(t,IC_max_T(i:N_gdl:end,1),'y','LineWidth',1),hold on
plot(t,IC_min_T(i:N_gdl:end,1),'b','LineWidth',1),hold on
% plot(t,Q_est(i:N_gdl:end,1).*180/pi,'b','LineWidth',1)
xlabel('Tiempo (seg)')
ylabel('Torque Nm')
title(['Torque del eje  ',num2str(i)])
legend('Estimado','Medido','IC_max','IC_min')
end
```

```
IC_min_Tv= Tv_est-3*sigma_Tv;
IC_max_Tv= Tv_est+3*sigma_Tv;

figure('name','23.-Torque de los ejes estimados e intrevalo de confianza validacion
    '), hold on

for i=1:N_gdl
subplot(2,4,i)
plot(t,Tv_est(i:N_gdl:end,1),'g','LineWidth',1),hold on
plot(t,Tv_m(i:N_gdl:end,1),'.r','LineWidth',1),hold on
plot(t,IC_max_Tv(i:N_gdl:end,1),'y','LineWidth',1),hold on
plot(t,IC_min_Tv(i:N_gdl:end,1),'b','LineWidth',1),hold on
% plot(t,Q_est(i:N_gdl:end,1).*180/pi,'b','LineWidth',1)
xlabel('Tiempo (seg)')
ylabel('Torque Nm')
title(['Torque del eje  ',num2str(i)])
legend('Estimado','Medido','IC_max','IC_min')
end
```

get_Propiedades_Inercia_Sw3

```
function [mass,center_of_mass,inertia_tensor] = get_Propiedades_Inercia_Sw3

% Masa de los s lidos creados
mass={8.00851643;5.57836486;5.28596317;4.54387162;4.07488166...
    ;3.18922203;3.30664153;0.84032115};

% Centro de masa de los s lidos
center_of_mass_0=[0.02368411;0;0.06831492];
center_of_mass_1=[0;0.02942832;0.12113433];%%
center_of_mass_2=[-0.00013633;-0.03689466;0.06589408];%%
center_of_mass_3=[0;-0.02765274;0.12768323];%%
center_of_mass_4=[0;0.03058484;0.07220698];%%
center_of_mass_5=[0;0.03066905;0.09172994];%%
center_of_mass_6=[0;-0.06119947;0.00250842];
center_of_mass_7=[0.00001462;0.00008710;0.01746594];

center_of_mass= {center_of_mass_0;center_of_mass_1;center_of_mass_2;
    center_of_mass_3;...
    center_of_mass_4;center_of_mass_5;center_of_mass_6;center_of_mass_7};

%Componentes de los Tensores de inercia de los s lidos
inertia_tensor_0=[0.08367145,0,0.01502060;0,0.10387593,0;0.01502060,0,0.06811282];
inertia_tensor_1=[0.14099861,0,0;0,0.13683345,0.02737149;0,0.02737149,0.02743275];
inertia_tensor_2
    =[0.08110360,0,0;0,0.07358222,-0.00503521;0,-0.00503521,0.02736241];
inertia_tensor_3
    =[0.11962720,0,0;0,0.11451190,-0.02313401;0,-0.02313401,0.01761697];
inertia_tensor_4=[0.05595238,0,0;0,0.05075992,0.00411955;0,0.00411955,0.01595760];
inertia_tensor_5=[0.05194039,0,0;0,0.04613032,0.01572631;0,0.01572631,0.01190130];
inertia_tensor_6=[0.02353577,0,0;0,0.01116808,0;0,0,0.01996721];
inertia_tensor_7=[0.00085693,-0.00000031,0.00000038;-0.00000031,...
    0.00085352,0.00000188;0.00000038,0.00000188,0.00092356];
% inertia_tensor_7=[0,0,0;0,0,0;0,0,0];
inertia_tensor = {inertia_tensor_0;inertia_tensor_1;inertia_tensor_2;
    inertia_tensor_3;...
    inertia_tensor_4;inertia_tensor_5;inertia_tensor_6;inertia_tensor_7};
end
```

F. Código de Simscape

```
clc
clear all
close all
% Obtengo las posiciones de mi trayectoria con la curva de bezzer acoplada
% del main_numeric_bezier...
%trayectoria utilizada, param_opt_cond_49.2447.mat
load('Q_DQ_DDQ.mat')
%Para alimentar mi modelo, los inputs de las articulaciones, deben incluir
%un vector de los intantes de timepo y posiciones respectivas para cada
%eje.
load('time')
load('smiData.mat') %%Se ha anadido la estructura que contiene los elementos
%de transformaci n y juntas de los s lidos, obtenido de correr el archivo
% Kukaiiwa_DataFile
```

```
t_end=time(end);
%La orientaci n de las bases entre s lidos se rige por la regla de la mano
%derecha, del s lido 1 al s lido 2, en cada enlace.
Q1=[time',Q(1,:)'];
Q2=[time',Q(2,:)'];
Q3=[time',Q(3,:)'];
Q4=[time',-Q(4,:)'];    % En esta articulaci n el modelo tiene invertida su
    orientaci n
Q5=[time',Q(5,:)'];
Q6=[time',Q(6,:)'];
Q7=[time',-Q(7,:)'];    % En esta articulaci n el modelo tiene invertida su
    orientaci n
%Realizo la simulaci n con mi modelo de simulink y simscape
sim('KUKA_iiwa.slx')

T=zeros(7*4400,1);
j=1;
for i=1:length(T1)
T((i-1)*7+1:7*i,1)=[T1(i);T2(i);T3(i);-T4(i);T5(i);T6(i);T7(i)];
end

QQ=zeros(7*4400,1);
j=1;
for i=1:length(q1)
QQ((i-1)*7+1:7*j,1)=[q1(i);q2(i);q3(i);-q4(i);q5(i);q6(i);q7(i)];
j=j+1;
end

DQQ=zeros(7*4400,1);
j=1;
for i=1:length(q1)
DQQ((i-1)*7+1:7*j,1)=[dq1(i);dq2(i);dq3(i);-dq4(i);dq5(i);dq6(i);dq7(i)];
j=j+1;
end

figure
for i=1:7
subplot(2,4,i)
plot(tout,T(i:7:end,1),'b'),hold on
xlabel('Tiempo (seg)')
ylabel('Torque (Nm)')
title(['Torque del motor ',num2str(i)])
%xlim([4 14])
end

figure
for i=1:7
subplot(2,4,i)
plot(tout,QQ(i:7:end,1).*180/pi,'r'),hold on
xlabel('Tiempo (seg)')
ylabel('Posici n (  )')
title(['Posici n del eje',num2str(i)])
%xlim([4 14])
end

figure
for i=1:7
subplot(2,4,i)
plot(tout,DQQ(i:7:end,1).*60*160/2/pi,'r'),hold on
xlabel('Tiempo (seg)')
ylabel('Velocidad (rpm)')
title(['Velocidad del eje ',num2str(i)])
%xlim([4 14])
end
```